Лекция №8
Фреймоврки

Гибадулин Олег

образование

Организационная часть

- Отметиться - важно
- Оставить отзыв (после занятия)

Структура лекции

- Что такое модуль
- Когда нужна модульность
- Как создавать модули
  - Библиотеки
  - Фреймворки
  - Swift Package
- Как распространять
  - CocoaPods
  - Carthage
  - SPM
- Как разделить приложение на модули

Модуль

Модуль - фрагмент повторно используемого кода, который можно использовать в нескольких приложениях

Модуляризация

Модуляризация — это процесс разбиения кодовой базы на отдельные модули

Почему стоит делить на модули

- Повторное использование кода
- Открытый доступ
- Инкапсуляция
- Разделение ответственности
- Ускорение времени сборки
- Масштабирование разработки

Почему не стоит делить на модули

- Сложность
- Накладные расходы
- Увеличение времени запуска

Как создать модуль

- Библиотеки
- Фреймворки
- Swift package

Вводные данные

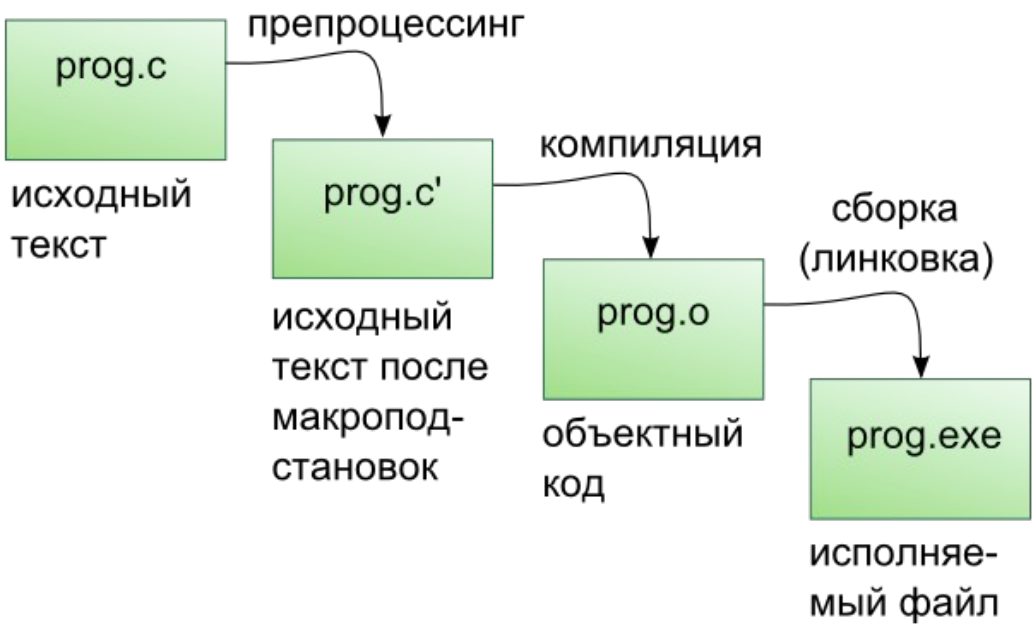
Исходный файл - файл исходного кода

Исполняемый файл - бинарный файл приложения (executable file) exe-шник

Объектный или object файл - файл содержащий машинный код

Машинный код - продукт компилятора

Вводные данные. Компиляция и линковка



Библиотека

Библиотека - это архив объектных файлов

Если просто, библиотека - это кучка скомпилированного кода, объединенная в один файл

Т.к. библиотека это последовательность инструкций на языке машинного кода, то это накладывает некоторые ограничения на их создание и распространение:

- Библиотека должна быть собрана для той же архитектуры процессора, что и клиент
- Библиотека не может включать файлы ресурсов

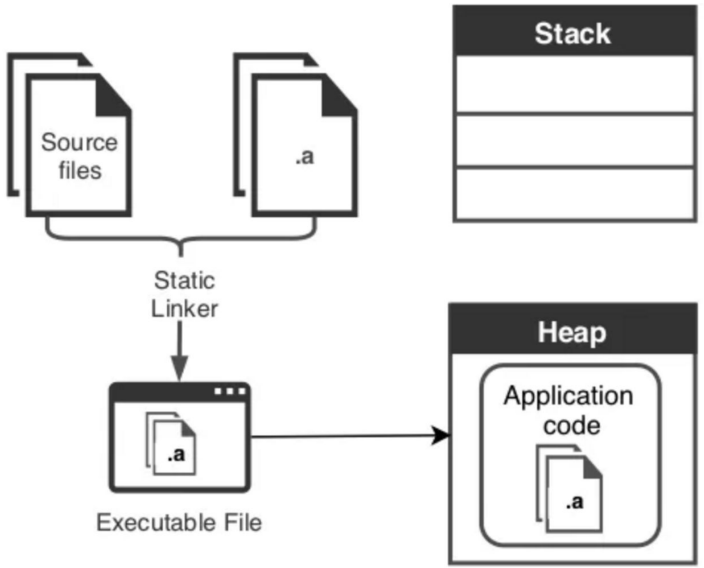
Разделяют статическую и динамическую библиотеки

Статическая библиотека

*.a

Компоновщик линкует скомпилированный исходный код приложения вместе с объектными файлами библиотеки.

В результате получается исполняемый файл, который полностью загружается в память во время выполнения.



Связывание статической библиотеки и использования памяти

Статическая библиотека. Плюсы и минусы

Преимущества

- Можно быть уверенным, что к библиотекам есть доступ и они правильных версий.
- Будут скопированы только те части либы, которые реально используются.

Недостатки

- Размер исполняемого файла приложения увеличивается
- Нужно больше времени для сборки исполняемого файла приложения
- Библиотека всегда загружена в память. Увеличивается время запуска приложения
- Сложно обновить у конечного пользователя

Статическая библиотека. Демо

Модификаторы доступа

- open
- public
- internal
- fileprivate
- private

Модификаторы доступа

public == final open

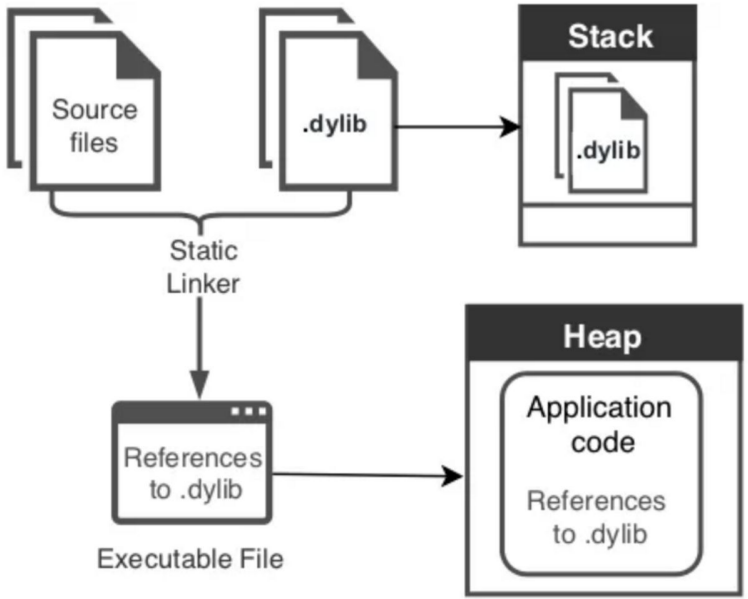
Динамическая библиотека

*.dylib

Не копируются в исполняемый файл.

Хранится и версионируется отдельно от приложения.

Динамически связываются при загрузке или во время работы приложения.



Связывание динамической библиотеки и использования памяти

Динамическая библиотека. Плюсы и минусы

Преимущества

- Размер исполняемого файла не увеличивается
- Легко обновить у конечного пользователя
- Может быть выгружена из памяти

Недостатки

- Загружается полностью, тк не известно какие функции будут использованы
- Увеличивает старт приложения
- Может быть недоступна или версия может не совпадать
- Может привести к Dependency hell

Динамическая библиотека. Демо

Динамические библиотеки (*.dylib) не поддерживаются на iOS

Фреймворк

Это иерархический каталог с различными типами файлов, включая другие библиотеки и фреймворки, заголовочные файлы и ресурсы

В отличие от библиотек:

- Может включать в себя другие библиотеки
- Может включать в себя такие ресурсы, как изображения, документацию, localization files, документацию
- В память загружается только одна копия фреймворка, независимо от количества приложений использующих её

Фреймворк

Помимо стандартных фреймворков существует:

- Umbrella framework
- Fat или universal framework
- XCFramework

You’re most likely to encounter submodules in large system frameworks like AppKit and Accelerate. These **umbrella frameworks** are no longer considered a best-practice, but they served an important role during Apple’s transition to Cocoa in the early 00’s.

Фреймворк. Демо

Swift Package

Swift Package содержит исходные файлы и файл манифеста (Package.swift)

В манифесте описывается конфигурация для пакета Swift

Создается и распространяется с помощью SPM

Swift Package

Swift Package состоит из 3 частей: **зависимостей**, **целей** и **продуктов**

```
let package = Package(
    name: "AnotherOnePackage",
    products: [
         // Products define the executables and libraries a package
             produces, and make them visible to other packages.
        .library(
            name: "AnotherOnePackage",
            targets: ["AnotherOnePackage"]),
    ],
    dependencies: [
        // Dependencies declare other packages that this package
            depends on.
        // .package(url: /* package url */, from: "1.0.0"),
    ],
    targets: [
        // Targets are the basic building blocks of a package. A target
            can define a module or a test suite.
        // Targets can depend on other targets in this package, and on
            products in packages this package depends on.
        .target(
            name: "AnotherOnePackage",
            dependencies: [],
            resources: [.process("Resources")])
    ]
)
```

Swift Package. Плюс и минусы

Преимущества

- Зависимости, управляемые Xcode
- Версии, управляемые Xcode
- Распространение в виде исходного кода:
  - позволяет проверять код и входить в него во время отладки
  - нет необходимости в поддержке множества архитектур
- Может содержать изображения, локализацию, XCFrameworks и другие типы файлов

Недостатки

- Распространение в исходном виде не подойдет для поставщиков фреймворков, которые не хотят делиться исходным кодом

Swift Package. Демо

Что выбрать

Связывание слишком большого количества **статических библиотек** в приложение приводит к большим исполняемым файлам приложения, замедленному времени запуска, большому объему памяти. Также отсутствуют ресурсы.

Фреймворки дают вам гораздо больше гибкости, чем статические библиотеки, они могут содержать ресурсы. Но каждый встроенный фреймворк, добавляемый в проект, также увеличивает время запуска.

Если вы можете открыто публиковать свои исходные файлы, **Swift packages** могут быть правильным выбором для вас. Xcode позаботится обо всех зависимостях, управлении версиями, платформах.

Как распространять

- Без менеджера зависимостей
- CocoaPods
- Carthage
- SPM

Cocoapods

Cocoapods - популярный менеджер зависимостей iOS

Необходимо описать от каких модулей зависит ваше приложение и Cocoapods сделаем все за вас

2 способа распространения:

- Исходный код
- Скомпилированные объекты: статические и динамические фреймворки, XCFrameworks.

Carthage

Наименее популярен

В отличие от Cocoapods не требует файла спецификации

В качестве альтернативы используются файлы проекта в Xcode

SPM

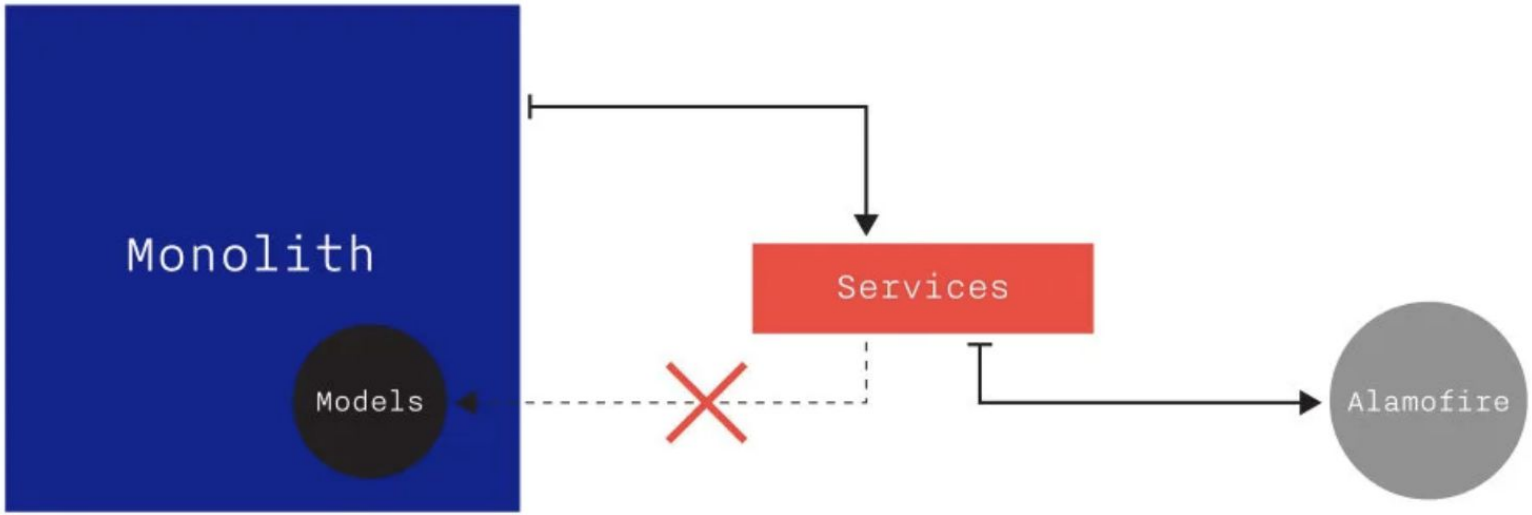
Входит в состав Swift и поддерживается сообществом Swift

Полагается на файл спецификации пакета (как в Cocoapods), который сообщает конечному потребителю, как установить и связать с пакетом.

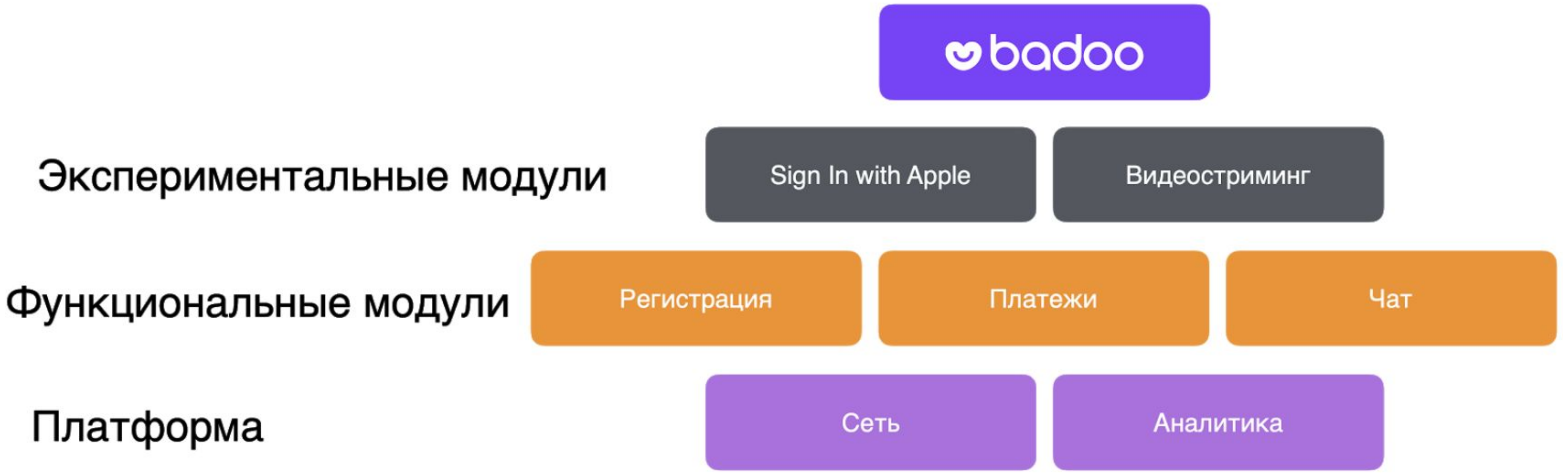
2 способа распространения:

- Исходный код
- Скомпилированные объекты: XCFrameworks.

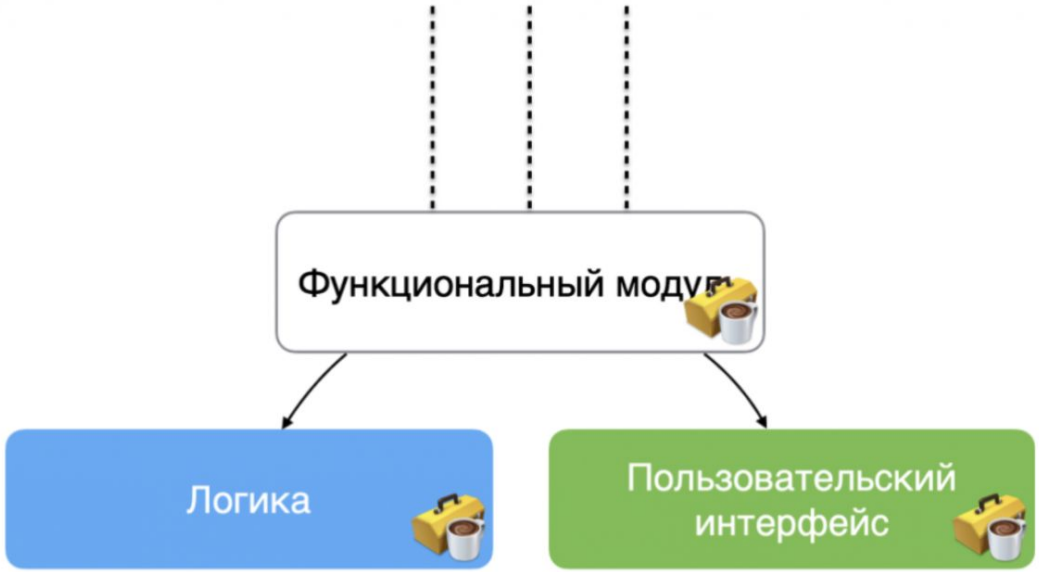
Как разделить приложение на модули



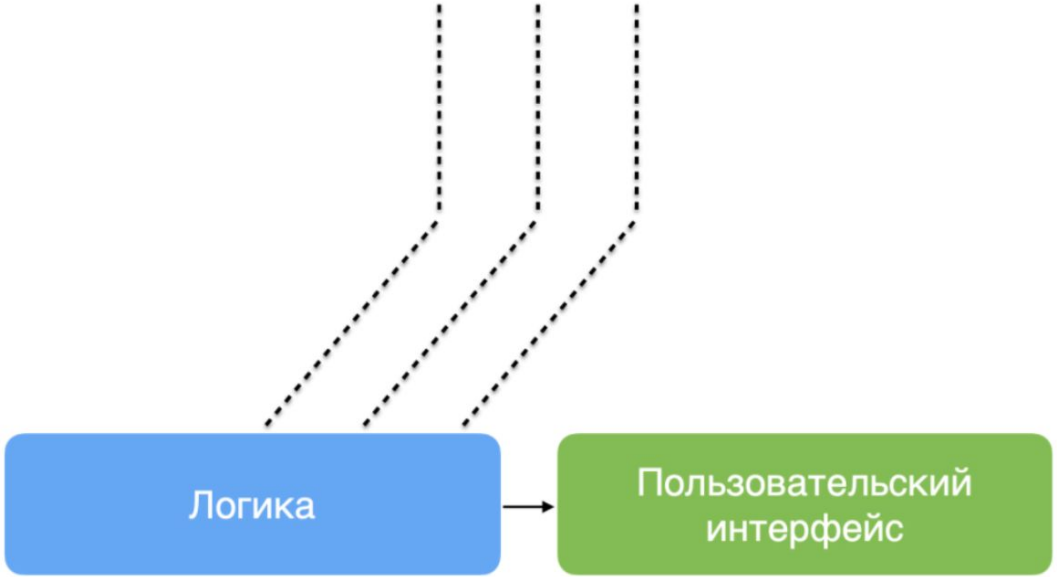
Как разделить приложение на модули



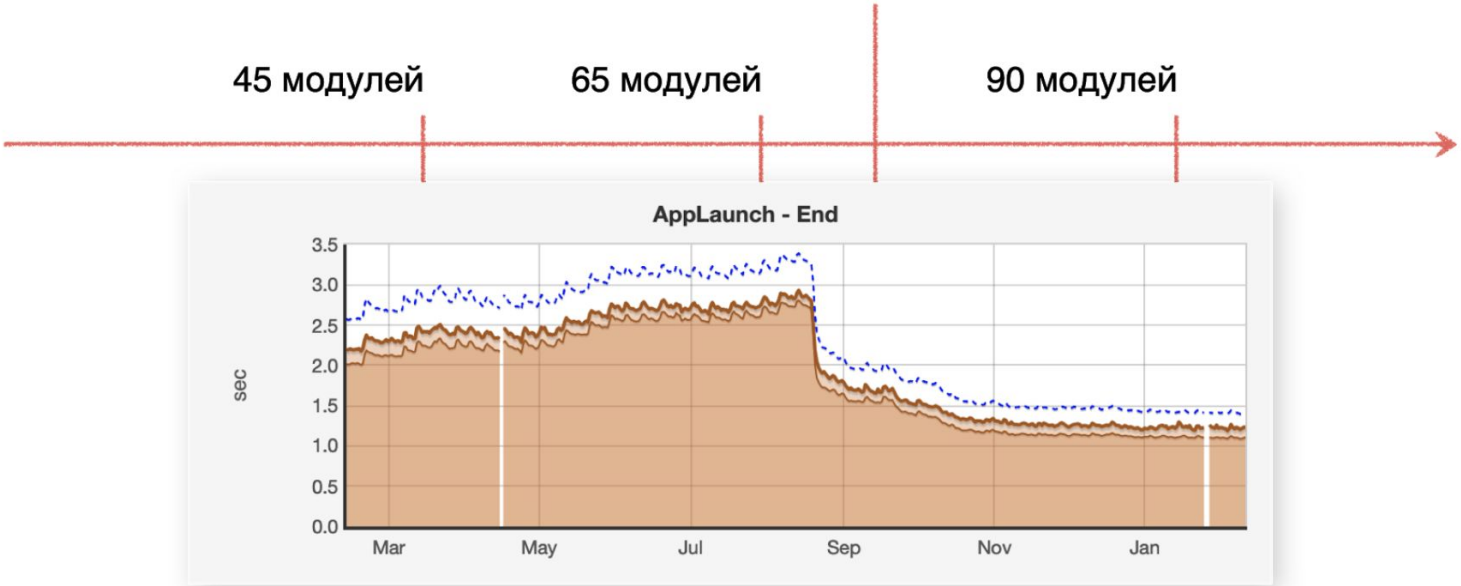
Как разделить приложение на модули



Как разделить приложение на модули



Как разделить приложение на модули



Литература

- Modular iOS Guide https://anuragajwani.medium.com/modular-ios-guide-60810f5a7f97
- Libraries, frameworks, swift packages… What's the difference? https://medium.com/@zippicoder/libraries-frameworks-swift-packages-whats-the-difference-764f371444cd
- Static Library vs Dynamic Library in iOS https://pratheeshbennet.medium.com/static-library-vs-dynamic-library-in-ios-55478ed53a03
- Модуляризация iOS-приложения: зачем и как мы разбиваем Badoo на модули - https://habr.com/ru/company/badoo/blog/531162/
- Модуляризация iOS-приложения Badoo: борьба с последствиями - https://habr.com/ru/company/badoo/blog/538270/

Литература. Статическая библиотека

- Static Library in iOS - https://medium.com/swift-india/static-library-in-ios-d133123678d1
- Создание статической библиотеки Static Library с использованием Cocoapods - https://habr.com/ru/post/586562/

Литература. Фреймворк

- Создание фреймворка Framework с использованием Cocoapods - https://habr.com/ru/post/586756/#products
- Creating a Framework for iOS - https://www.kodeco.com/17753301-creating-a-framework-for-ios#toc-anchor-009

Литература. Swift Package

- Building a Library with Swift Package Manager - https://dev.to/appwrite/swift-101-building-a-library-with-swift-package-manager-46h
- How to modularize existing iOS projects using Swift Package - https://sarunw.com/posts/how-to-modularize-existing-ios-projects-using-swift-package/
- How to Add Resources in Swift Package Manager - https://betterprogramming.pub/how-to-add-resources-in-swift-package-manager-c437d44ec593

Вопросы

Спасибо за внимание!

образование