Менеджеры зависимостей Cocoapods, Carthage, SPM

\* Исходный код — текст программы, который написал разработчик.

\* Исполняемый код — код, который получился в результате компиляции исходного кода.

Определение

Зависимость — использование стороннего кода в нашей программе.

Зависимости позволяют воспользоваться реализованными кем-то хорошими решениями (методами или целой архитектурой и т.п.)

Решения хранят в себе библиотеки и фреймворки.

Для установки и обновления версий библиотек используются менеджеры зависимостей.

Основные термины, используемые при работе с менеджерами зависимостей

**Semantic Versioning (SemVer)**

Стандарт-описания работы с версиями пакетов/библиотек/приложений — как их указывать и накладывать на них ограничения.

По этому стандарту № версии — это строка из чисел, разделённых точкой, и имеющая формат A.B.C, где

A — мажорная версия. Увеличивается, если новая версия включает в себя так называемые breaking changes, т. е. обратно несовместимые изменения.

B — минорная версия. Увеличивается при добавлении новых фич или депрекации(помечании/удалении) старых.

C — патч-версия. Увеличивается при исправлении багов.

**Модули**

Модуль — отдельный блок исполняемого кода или функциональности, которые можно распространять разными способами и переиспользовать.

Виды модулей:

* Библиотека. Представляет собой набор готовых методов.
* Фреймворк - папка, содержащая библиотеку и вспомогательные файлы. Может определять архитектуру приложения, разбивку на классы и объекты, их взаимодействие и поток управления.
* XCFramework. Набор фреймворков под разные архитектуры.
* Swift-пакет. Фреймворк, который используется для работы со Swift Package Manager.

Статическая библиотека - набор объектных файлов, скомпилированных и связанных в один файл, который включается непосредственно в исполняемый файл вашего приложения во время сборки.

Динамическая библиотека - файл, который загружается в память во время выполнения программы, а не включается напрямую в исполняемый файл приложения.

**CocoaPods**

Архитектура и подход к управлению зависимостями: Использует централизованный репозиторий для управления зависимостями и установки библиотек.

Способ интеграции в проект: После определения зависимостей в файле Podfile, они устанавливаются через команду pod install, а затем CocoaPods генерирует на основе их .xcworkspace, который будет содержать два проекта — ваш проект и проект с подами.

Важны два термина — подфайл и спека.

Подфайл описывает все зависимости приложения, а спека — библиотеку (её название и набор её зависимостей) для Cocoapods.

**Carthage**

Архитектура и подход к управлению зависимостями: Работает на основе децентрализованной модели, где зависимости загружаются из исходных репозиториев и собираются локально.

Способ интеграции в проект: Зависимости определяются в файле Cartfile, а затем загружаются и собираются с помощью команды carthage update, после чего вы вручную интегрируете скомпилированные фреймворки в проект.

**Swift Package Manager (SPM)**

Архитектура и подход к управлению зависимостями: Предлагает интегрированную поддержку в среде разработки Xcode и работает с Package.swift файлом, который определяет зависимости и их версии.

Способ интеграции в проект: Интегрируется непосредственно в Xcode, и зависимости автоматически загружаются и интегрируются в проект при сборке.