FRAMEWORK ORIENTED PROGRAMMING

Pedro Piñera @pepibumur - IOS DEVELOPER AT SOUNDCLOUD

NSBUDAPEST

SZIASZTOK! 👋

Pedro

IOS DEVELOPER AT SOUNDCLOUD

@PEPIBUMUR

TWITTER/FACEBOOK/~~YOUTUBE~~

WWW.PPINERA.ES

CONTEXT

Before 2008

OSX - 1 TARGET 📦

CONTEXT

2008

🍎 LAUNCHES IPHONE SOFTWARE DEVELOPMENT KIT 📱

(Developers move to iOS. New platform, frameworks,... New exciting area)

CONTEXT

After 2008

📱 - 1 TARGET 📦

💻 - 1 TARGET 📦

CONTEXT

2011

COCOAPODS RELEASED

Dependency Resolving + Integration + Community 🎉

CONTEXT

After 2011

📱 - 1 TARGET 📦

💻 - 1 TARGET 📦

X TARGETS (EXTERNAL) 📦📦

CONTEXT

2015

CONTEXT

2015

📱 - 1 TARGET

💻 - 1 TARGET

📺 - 1 TARGET

⌚ - 1 TARGET

HOW TO REUSE CODE?

(ACROSS PLATFORMS)

Frameworks 📦

SWIFT ❤
DYNAMIC FRAMEWORKS

EMBEDDED RESOURCES

(IMAGES, FONTS, ...)

DYNAMICALLY LINKED

(NO DUPLICATED SYMBOLS)

SWIFT CODE

FRAMEWORK ORIENTED PROGRAMMING

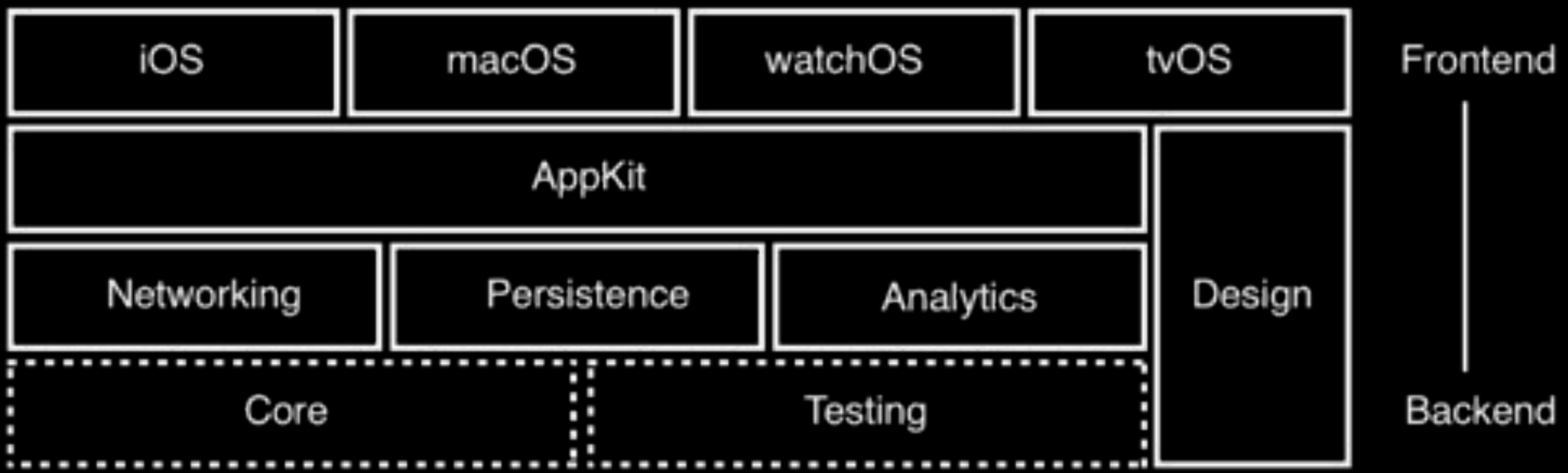
Reusable & platform independent code

GITHUB

BEST PRACTICES
PRINCIPLES
ADVICES
EXAMPLES

FRAMEWORKS STACK

SoundCloud Approach

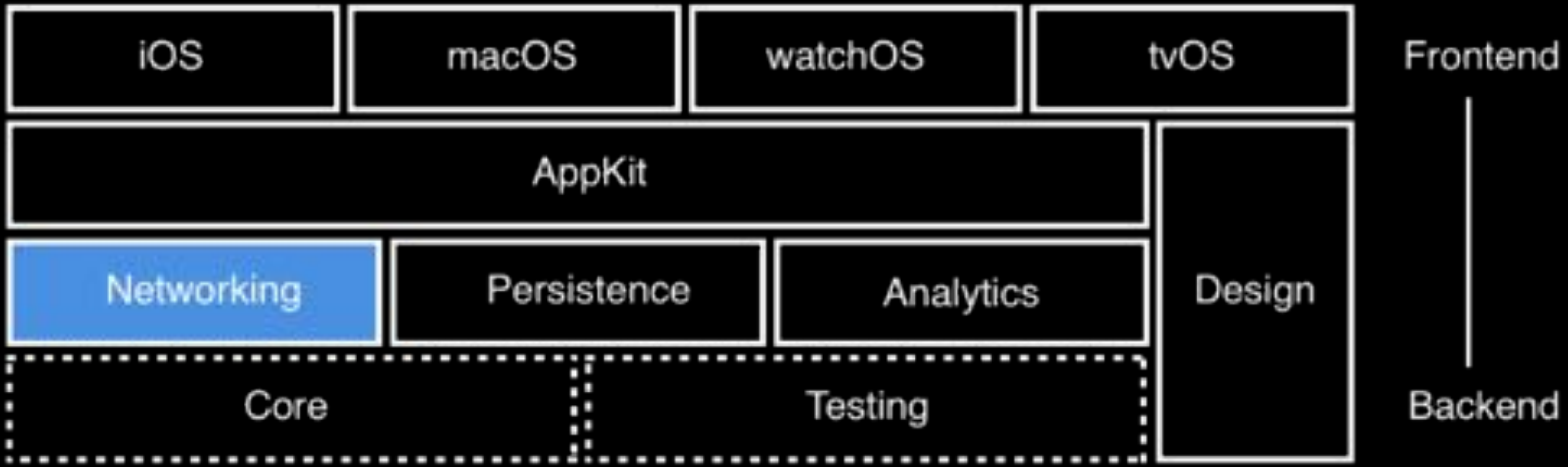


PRINCIPLES

Some theory 📖

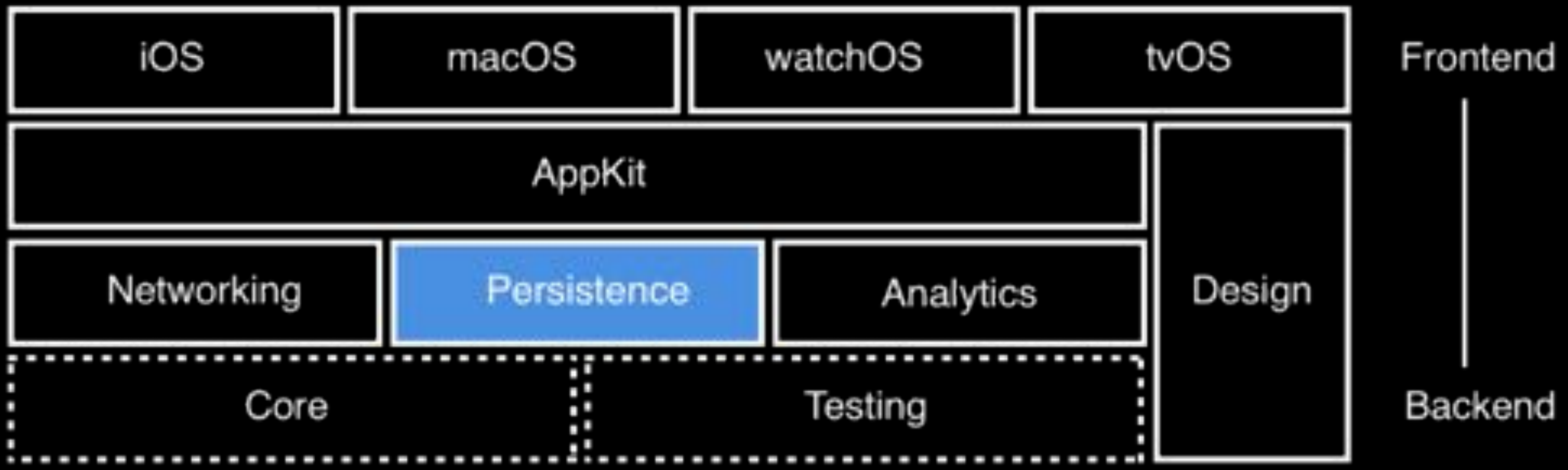
1. SINGLE RESPONSIBILITY

SOLID INSPIRED



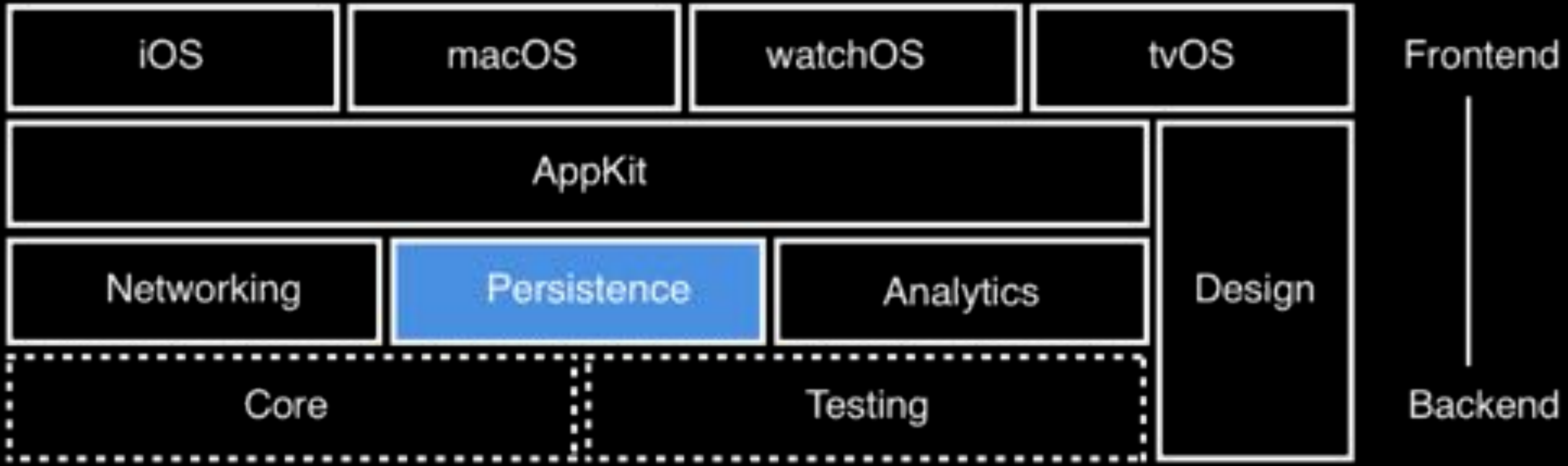
1. SINGLE RESPONSIBILITY

SOLID INSPIRED



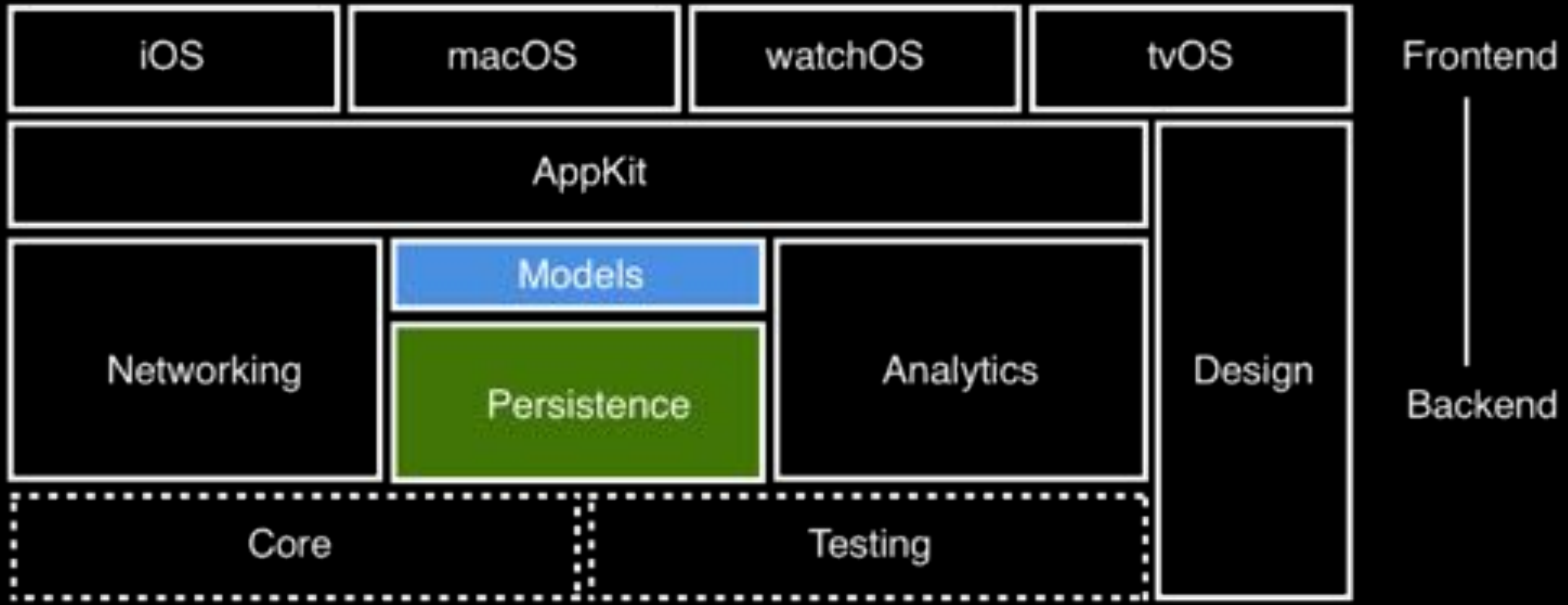
1. SINGLE RESPONSIBILITY

START FROM A HIGH LEVEL



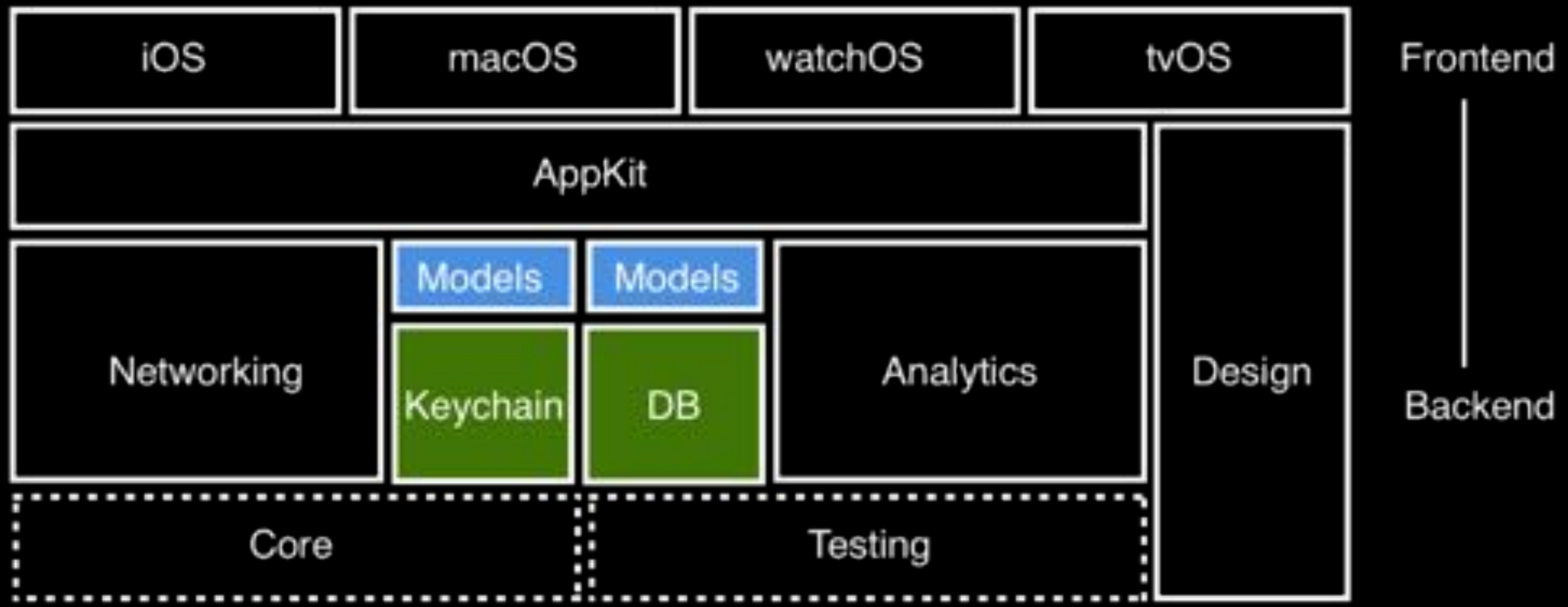
1. SINGLE RESPONSIBILITY

SLICE THEM PROGRESSIVELY



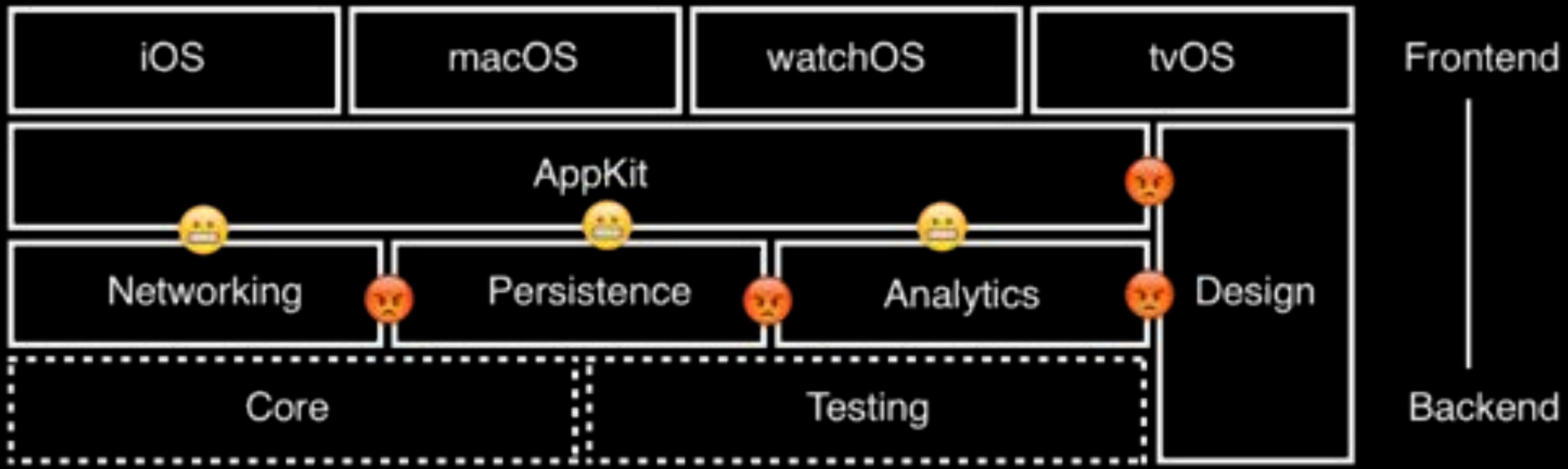
1. SINGLE RESPONSIBILITY

SLICE THEM PROGRESSIVELY



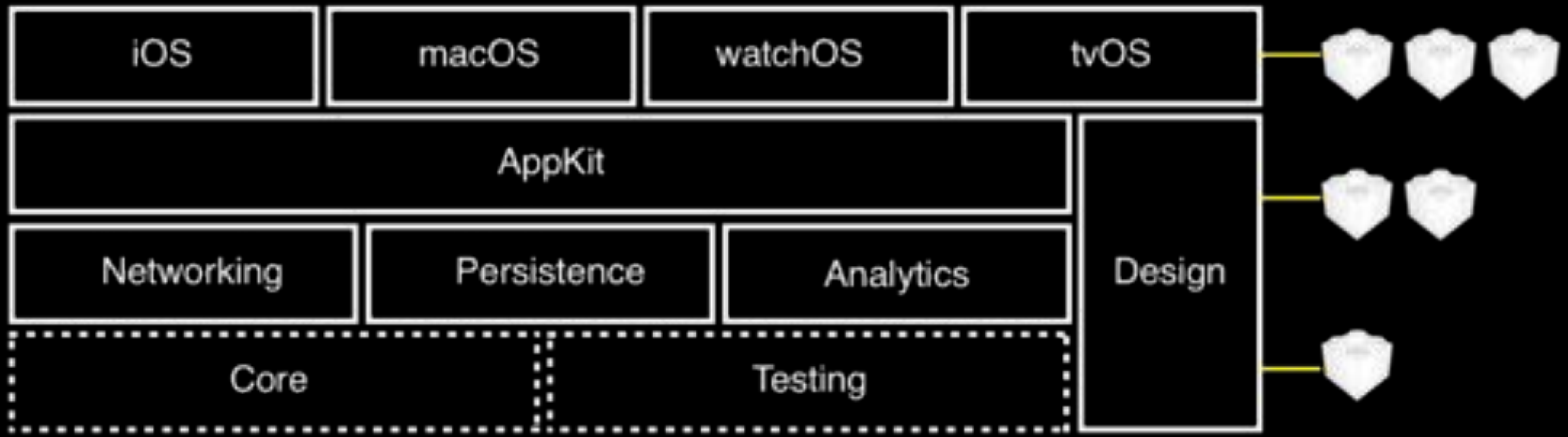
2. VERTICAL DEPENDENCIES

(OVER HORIZONTAL)

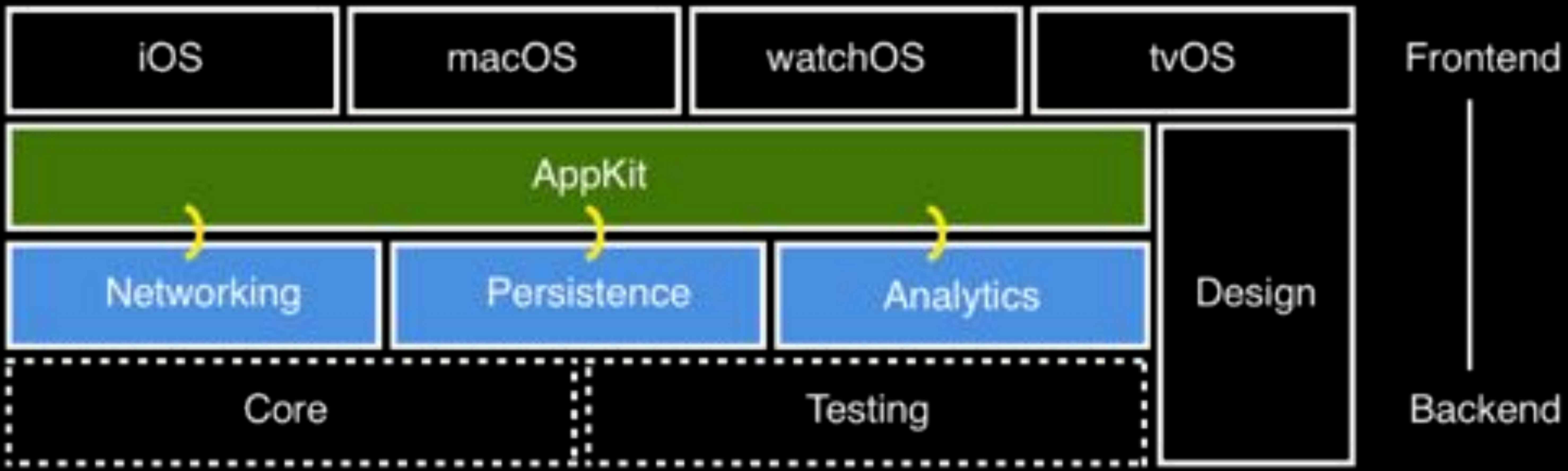


3. LOWER IN THE STACK

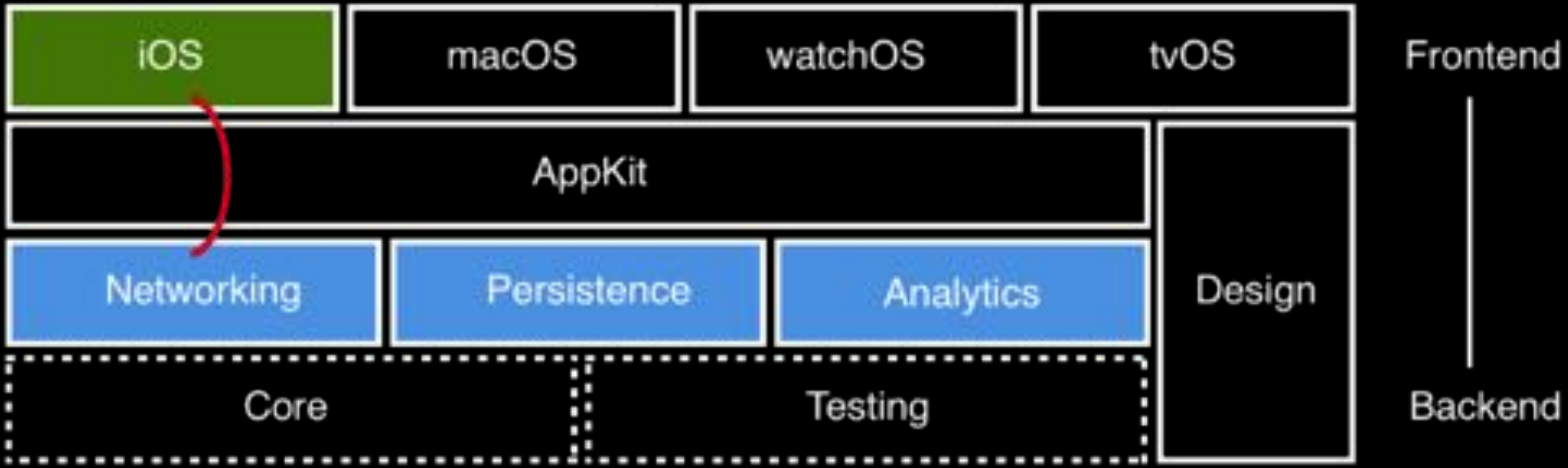
FEWER EXTERNAL DEPENDENCIES



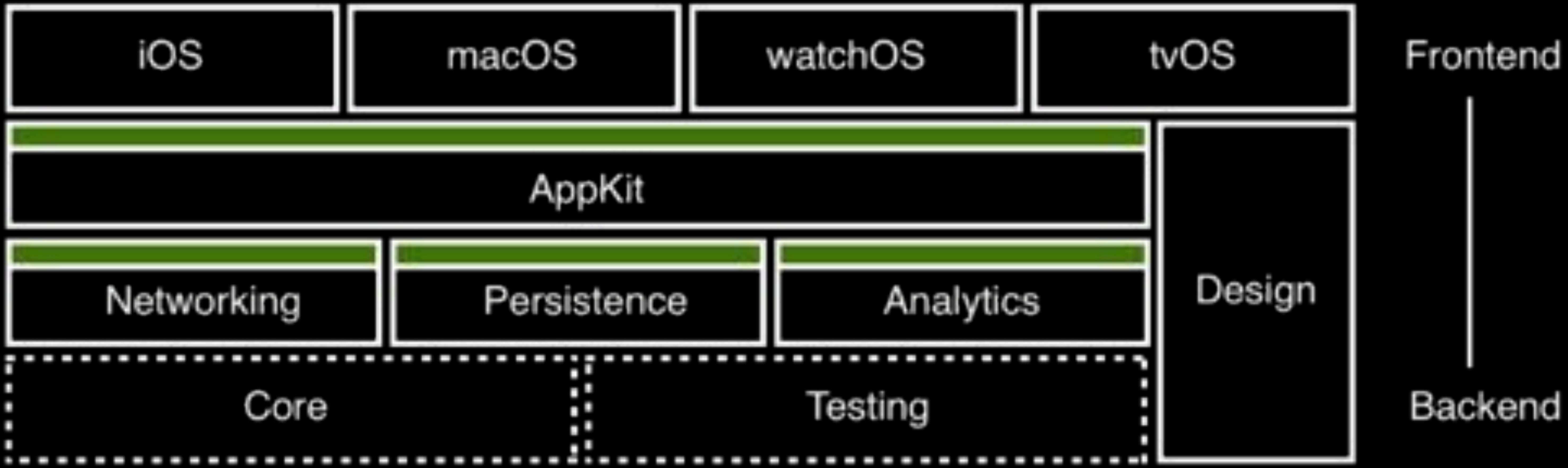
4. ONE STEP DEPENDENCIES



4. ONE STEP DEPENDENCIES

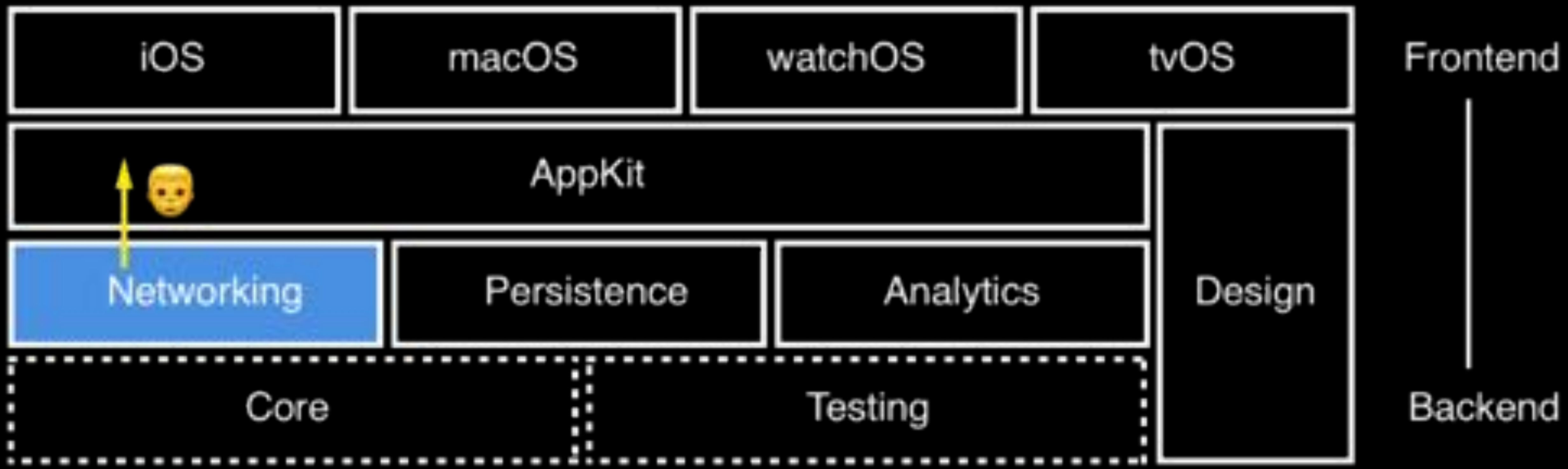


5. INTERNAL BY DEFAULT



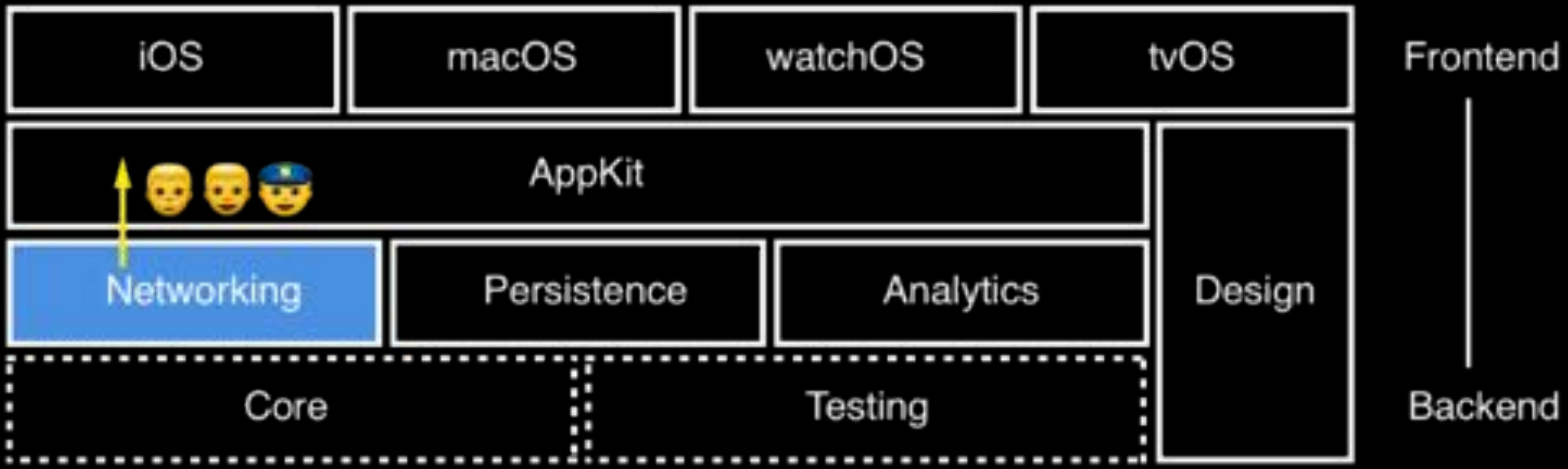
6. FINAL

SOLID INSPIRED (OPEN/CLOSED)



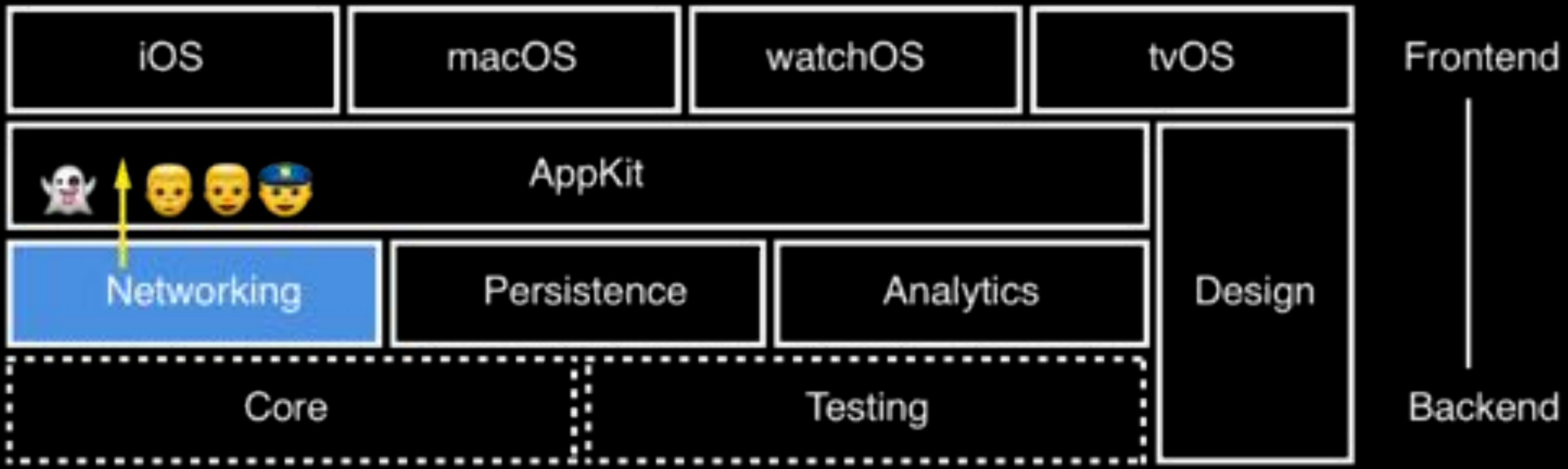
6. FINAL

SOLID INSPIRED (OPEN/CLOSED)



6. FINAL

SOLID INSPIRED (OPEN/CLOSED)



6. FINAL

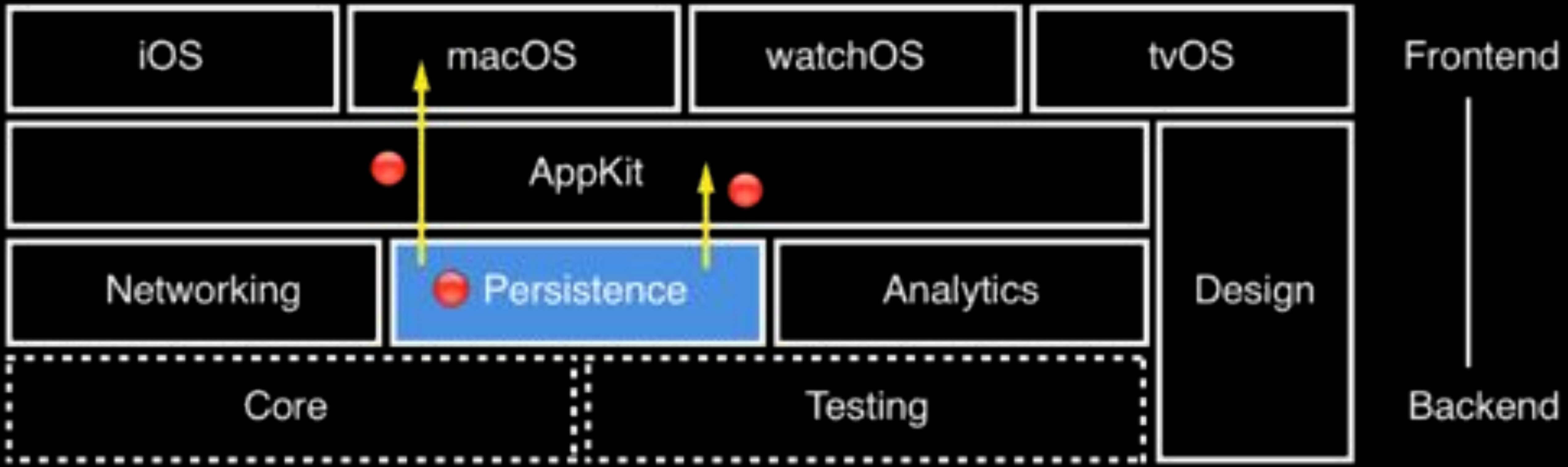
SOLID INSPIRED (OPEN/CLOSED)

```
final class Person {
  let name: String
}

class Alien: Person { // Compiler complains
}
```

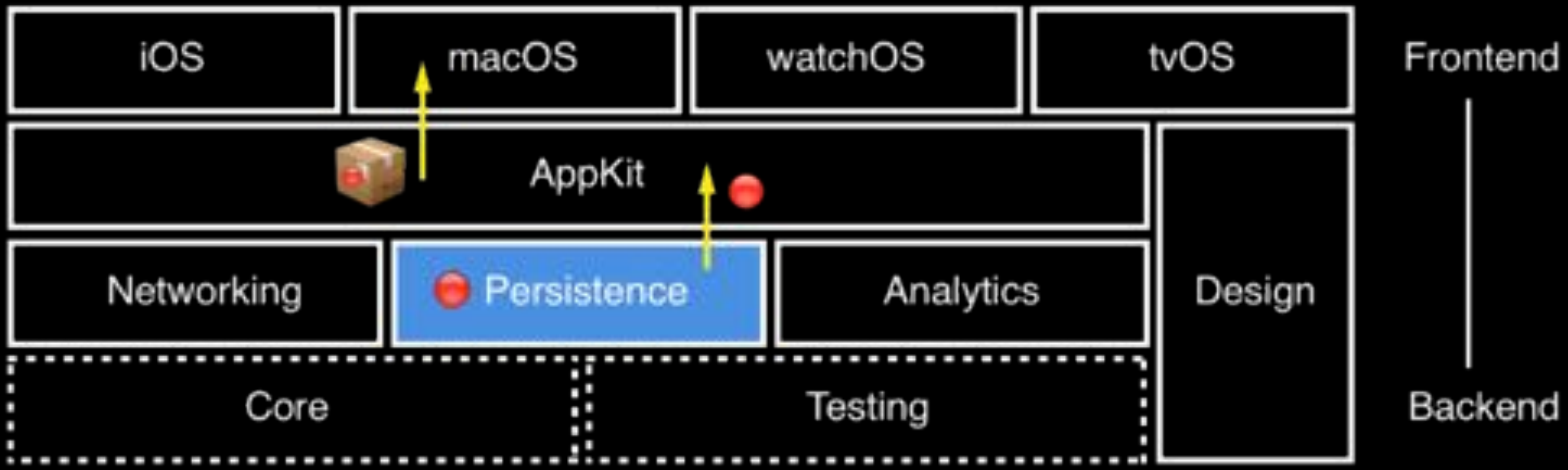
7. FRAMEWORK MODELS

DON'T SHARE LOWER FRAMEWORKS MODELS UPWARDS



7. FRAMEWORK MODELS

DON'T SHARE LOWER FRAMEWORKS MODELS UPWARDS



7. FRAMEWORK MODELS

DON'T SHARE LOWER FRAMEWORKS MODELS UPWARDS

```
// Persistence
class Author: NSManagedObjectModel {
  let name: String
}
class Track: NSManagedObjectModel {
  let author: Author
}

// ListenersKit
struct StreamTrackEntity {
  let name: String
  let authorName: String
}
```

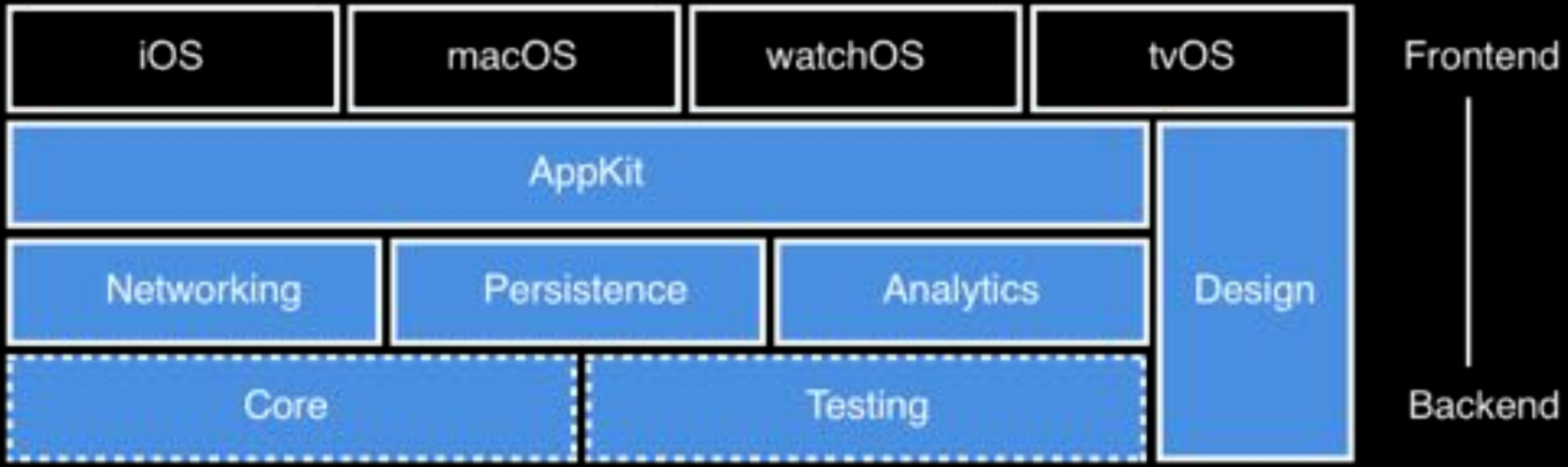
7. FRAMEWORK MODELS

DON'T SHARE LOWER FRAMEWORKS MODELS UPWARDS

```swift
struct StreamTrackEntityAdapter {
  func adapt(track: Track) -> StreamTrackEntity {
    return StreamTrackEntity(name: track.name, authorName: track.author.name)
  }
}
```

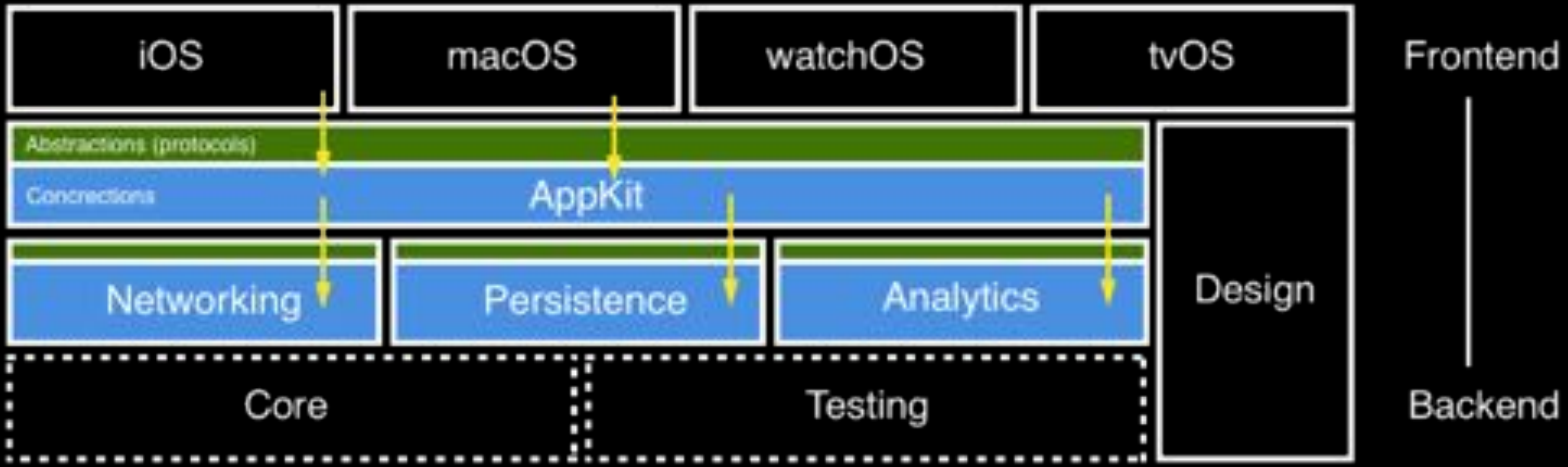
8. PLATFORM ABSTRACTION

SOLID INSPIRED (DI)



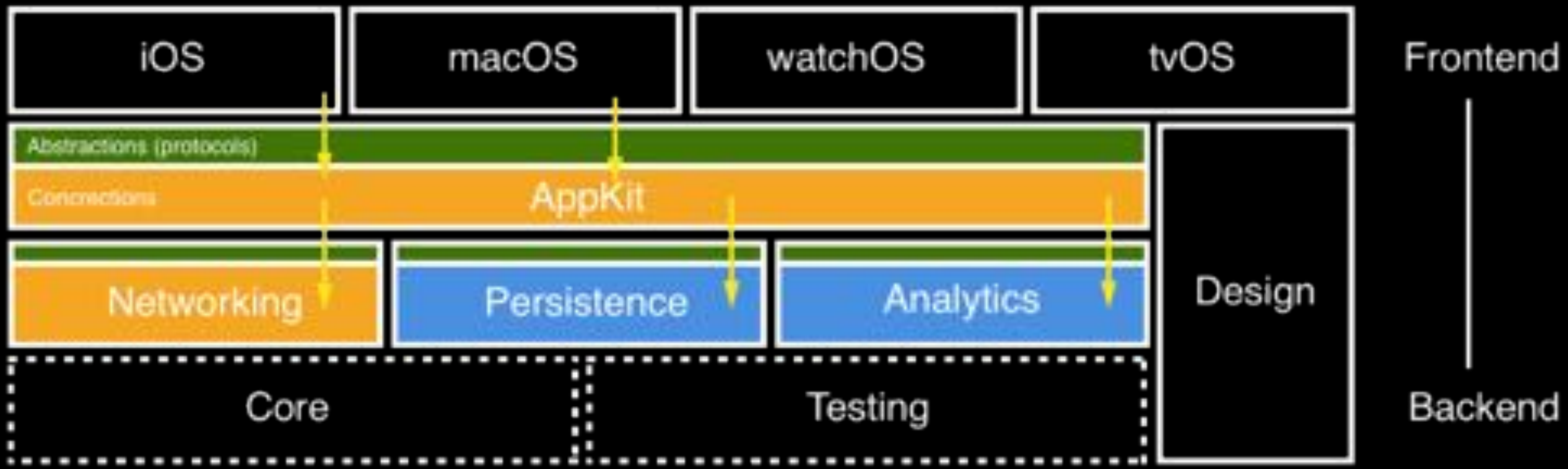
9. PROTOCOL ORIENTED INTERFACES

SOLID INSPIRED (DI)



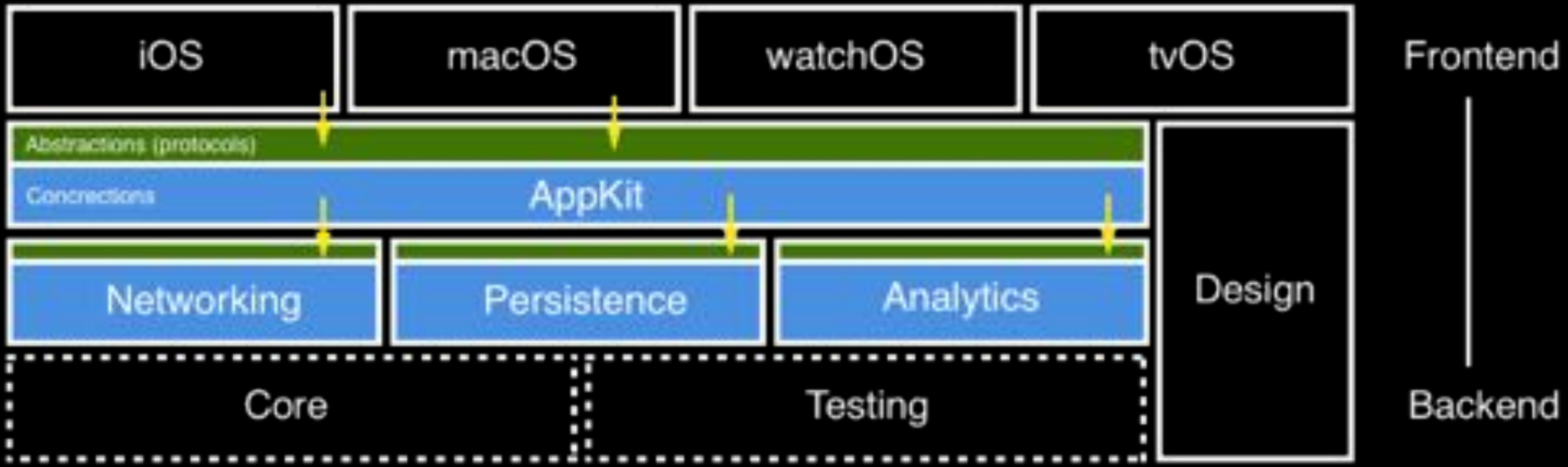
9. PROTOCOL ORIENTED INTERFACES

SOLID INSPIRED (DI)



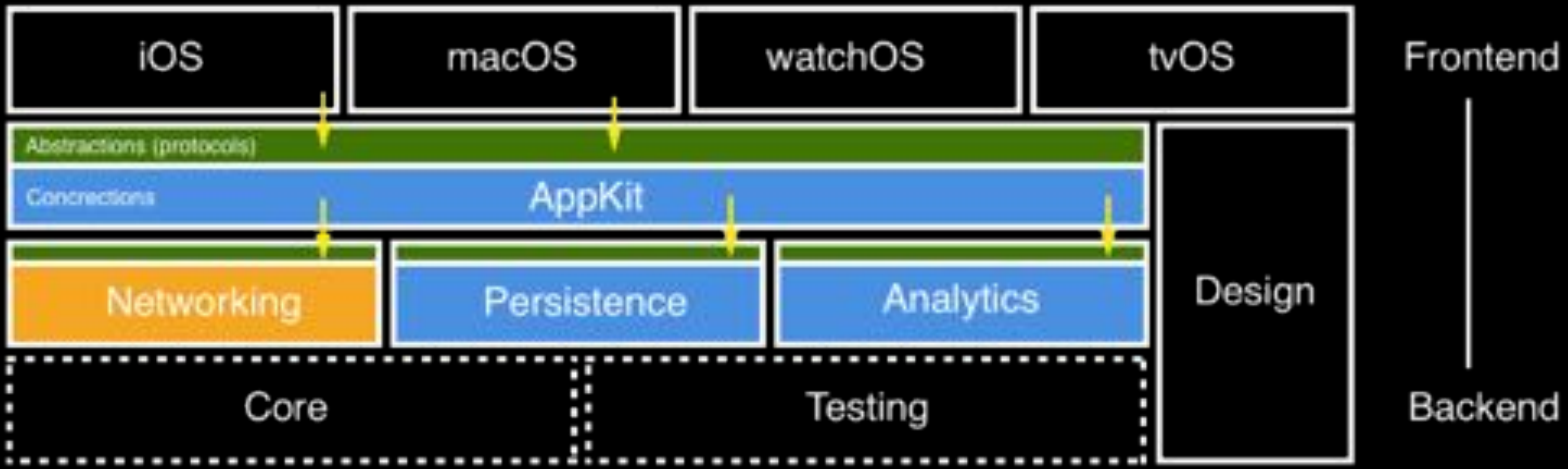
9. PROTOCOL ORIENTED INTERFACES

SOLID INSPIRED (DI)



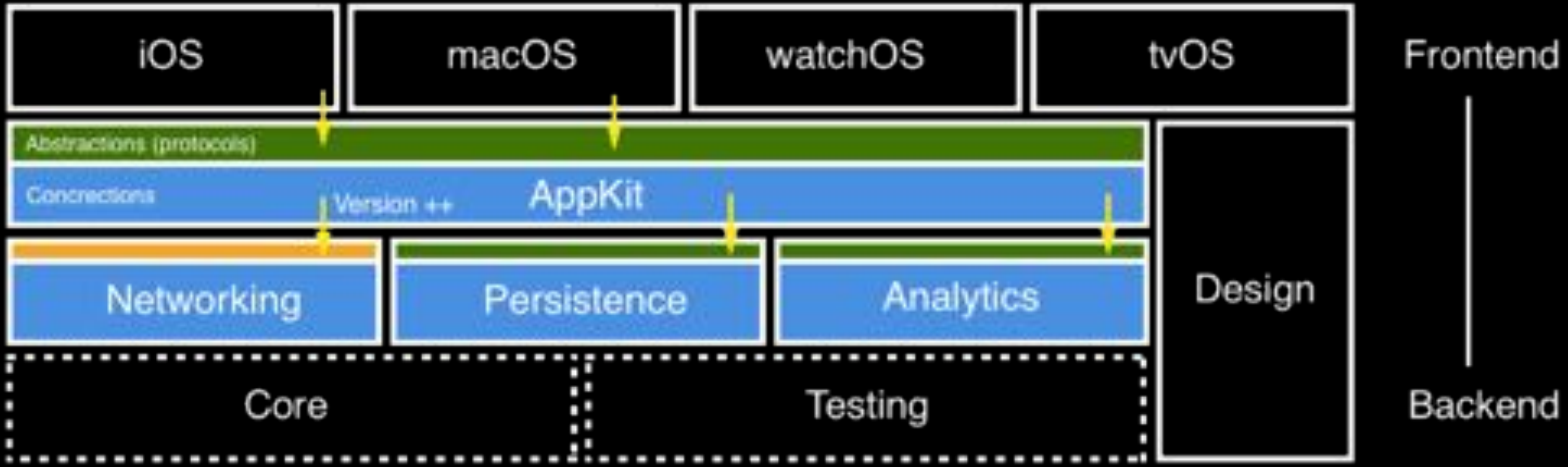
9. PROTOCOL ORIENTED INTERFACES

SOLID INSPIRED (DI)



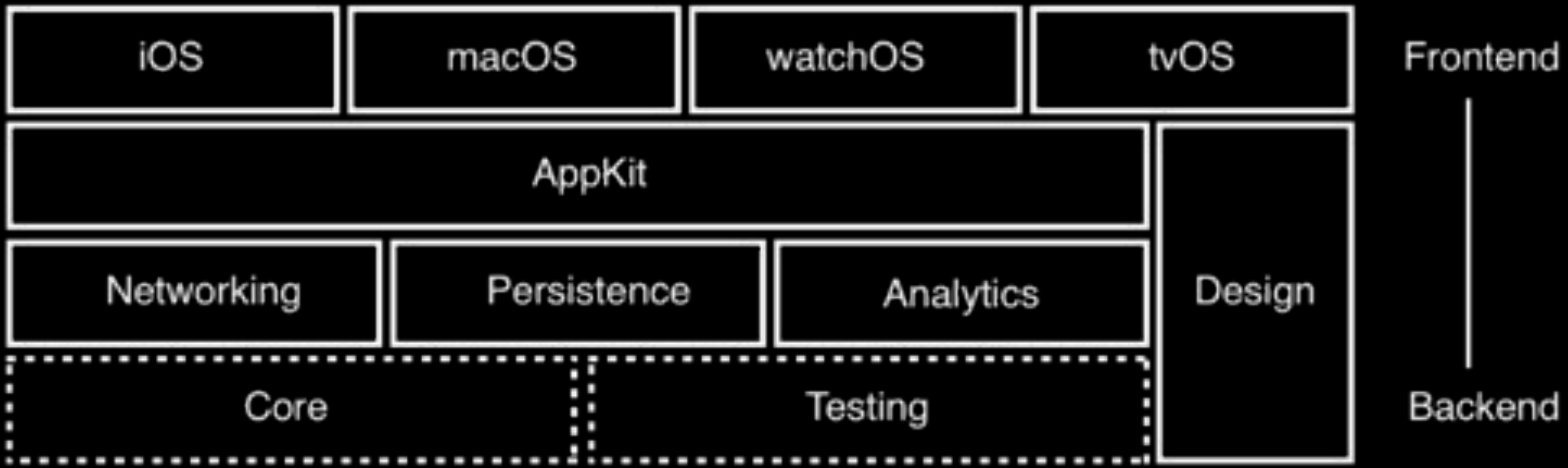
9. PROTOCOL ORIENTED INTERFACES

SOLID INSPIRED (DI)



10. CORE/TESTING

(AKA YOUR PROJECT FOUNDATION FRAMEWORKS)



10. CORE/TESTING

(AKA YOUR PROJECT FOUNDATION FRAMEWORKS)

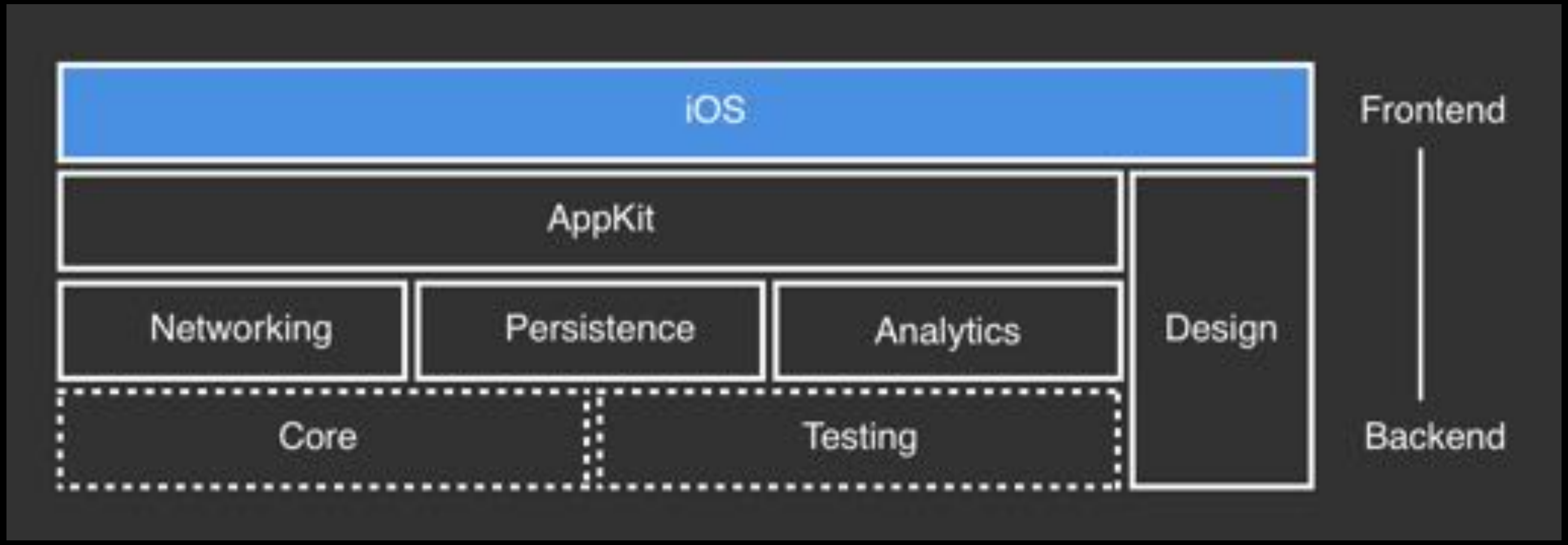
- Extensions
- Logging
- Analytics
- Architectural components (e.g. Reactive)

ADVANTAGES

MULTIPLATFORM APPS

Only working on the UI

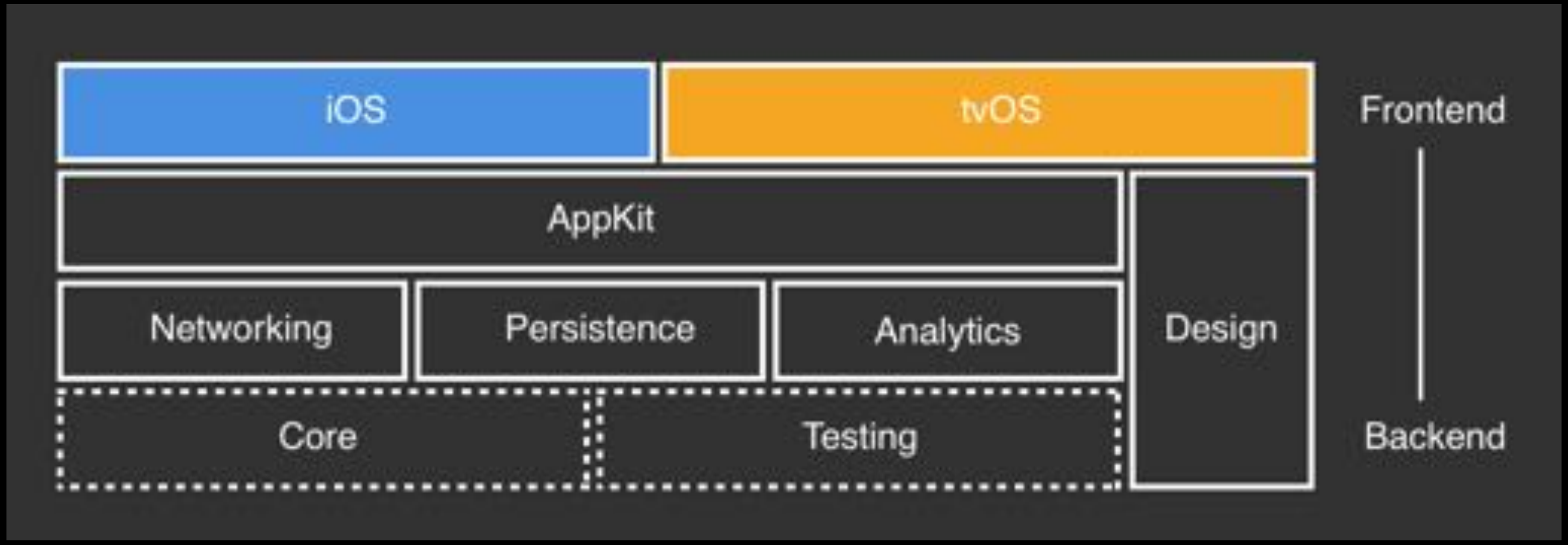
⌚ 📱 📺 💻



MULTIPLATFORM APPS

Only working on the UI

⌚ 📱 📺 💻



EXPERIMENTATION

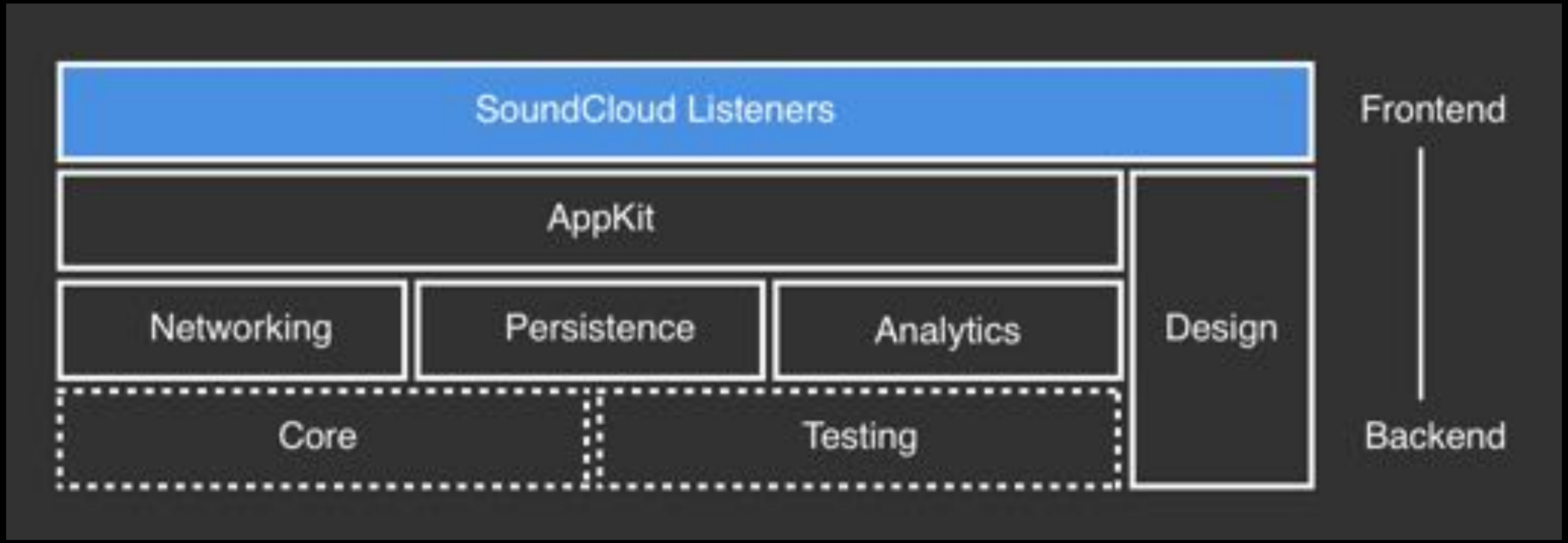
- Prototyping
- Playgrounds

```
import MyAppKit
requestFactory.request(path: "/myPath/").subscribeNext { response in
  // yai!
}
```

NEW PRODUCTS

With similar core needs

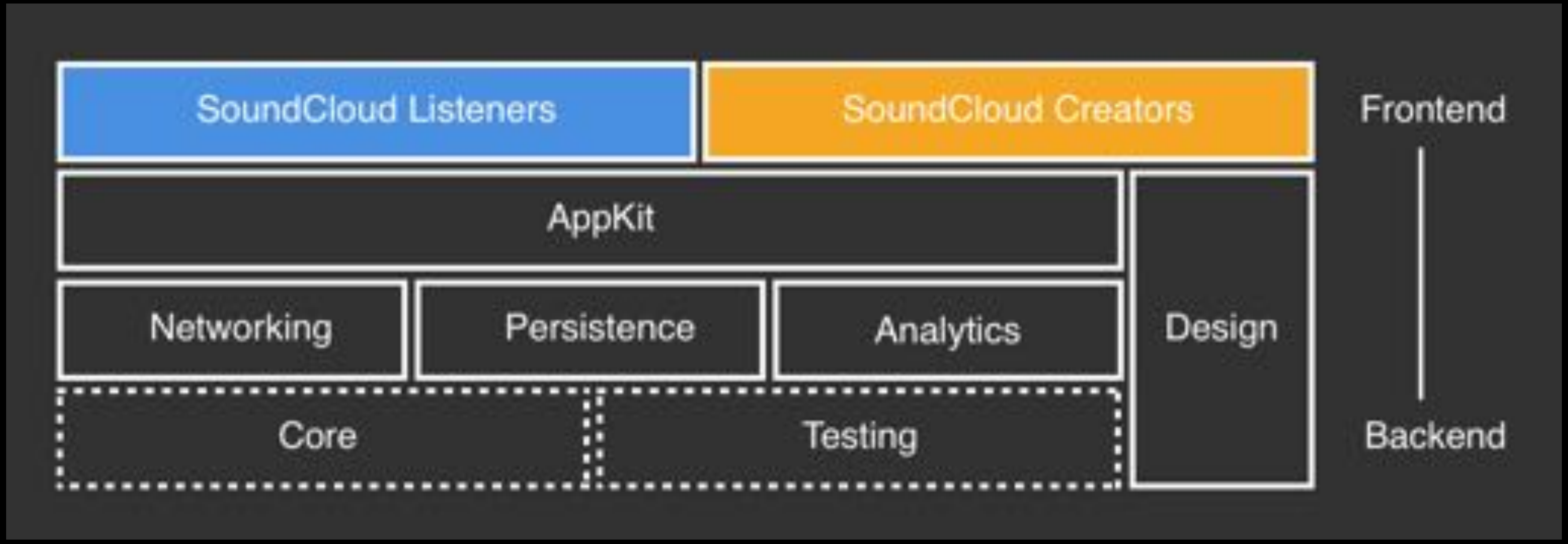
BECAUSE YOU WANT TO REUSE CODE, RIGHT?



NEW PRODUCTS

With similar core needs

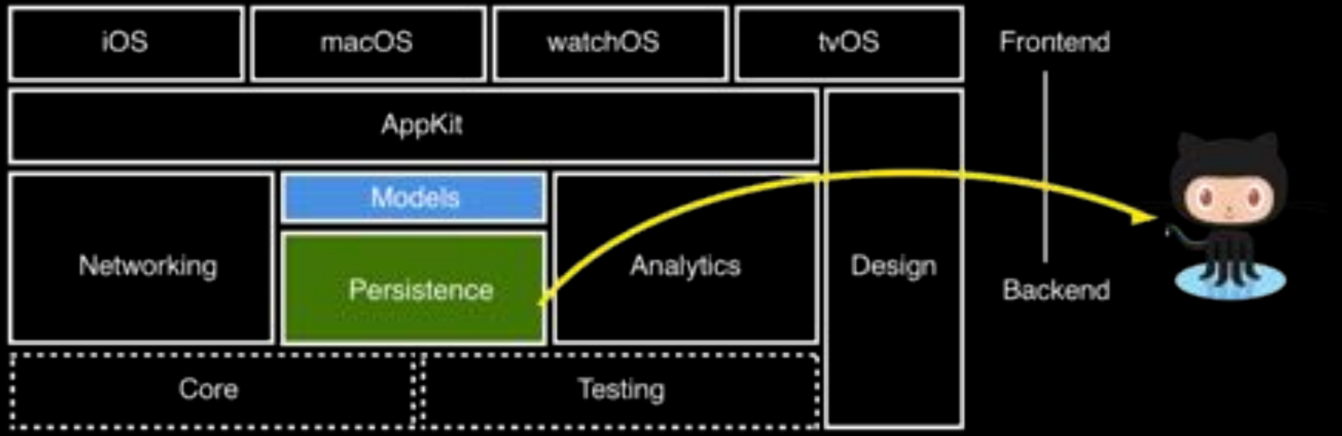
BECAUSE YOU WANT TO REUSE CODE, RIGHT?



OPEN SOURCE

And benefit from the community

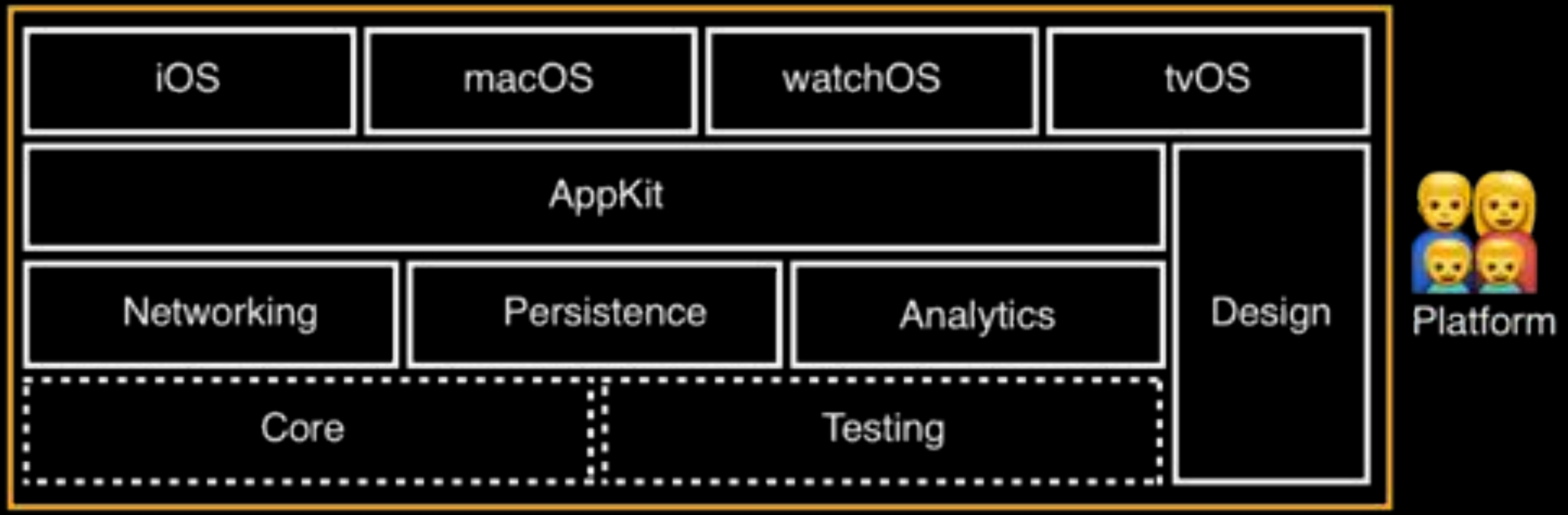
BUILD PIECES OF CODE THAT YOU'D BE PROUD OF OPEN SOURCING



SPECIALIZED TEAMS

From UI lovers to Core Data experts

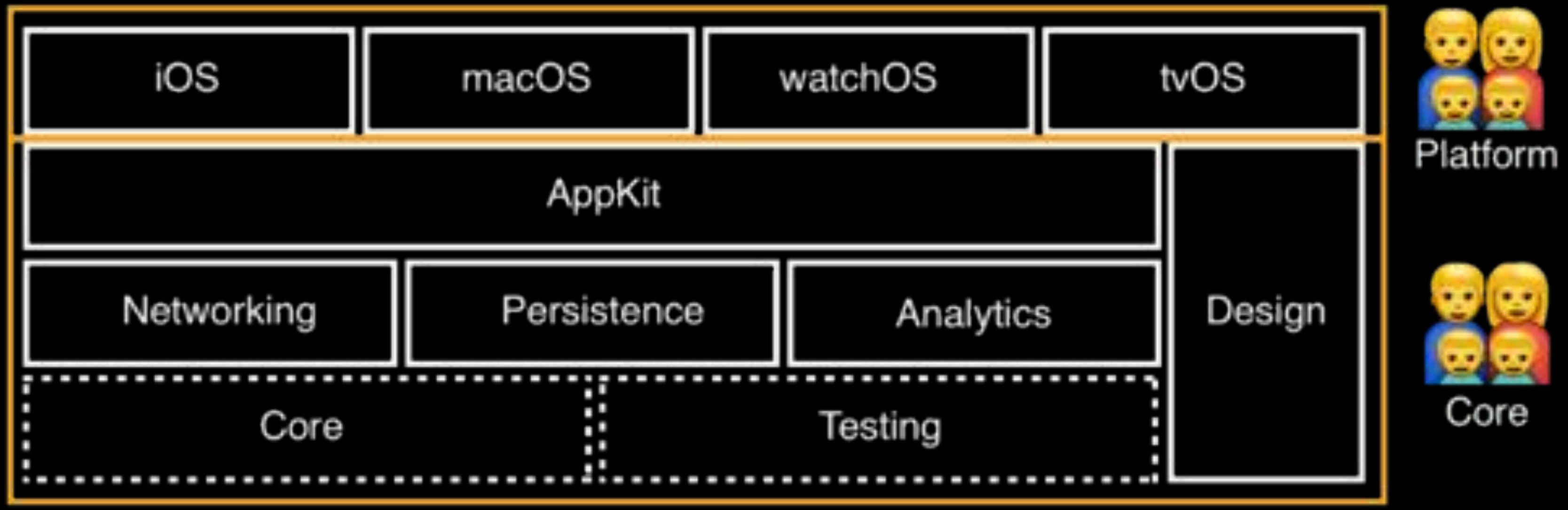
(CLEARLY DEFINED TEAM BOUNDARIES)



SPECIALIZED TEAMS

From UI lovers to Core Data experts

(CLEARLY DEFINED TEAM BOUNDARIES)

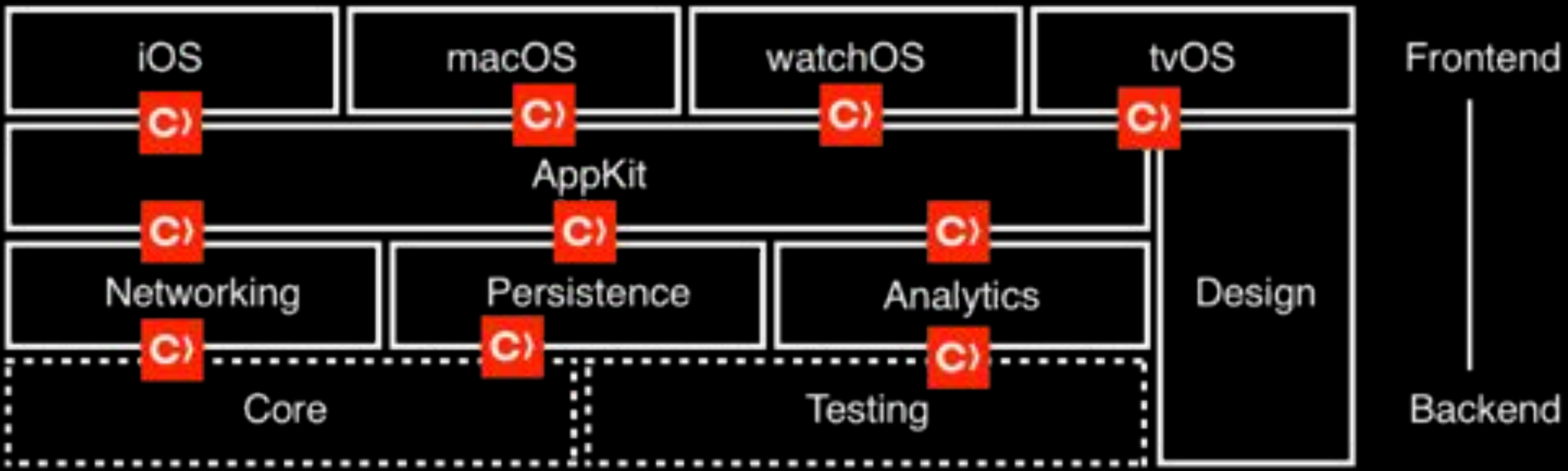


HOW TO? 🤔

There are multiple options

(I'LL SHOW YOU SOME)

CocoaPods


iOS
macOS
watchOS
tvOS
Frontend
AppKit
Networking
Persistence
Analytics
Design
Core
Testing
Backend

CocoaPods

- ✅ Easy setup (each Framework `.podspec`)
- ✅ Same setup for local/external dependencies
- ❌ It sucks if you don't version
- ❌ Fully frameworks approach (load time)

Manual

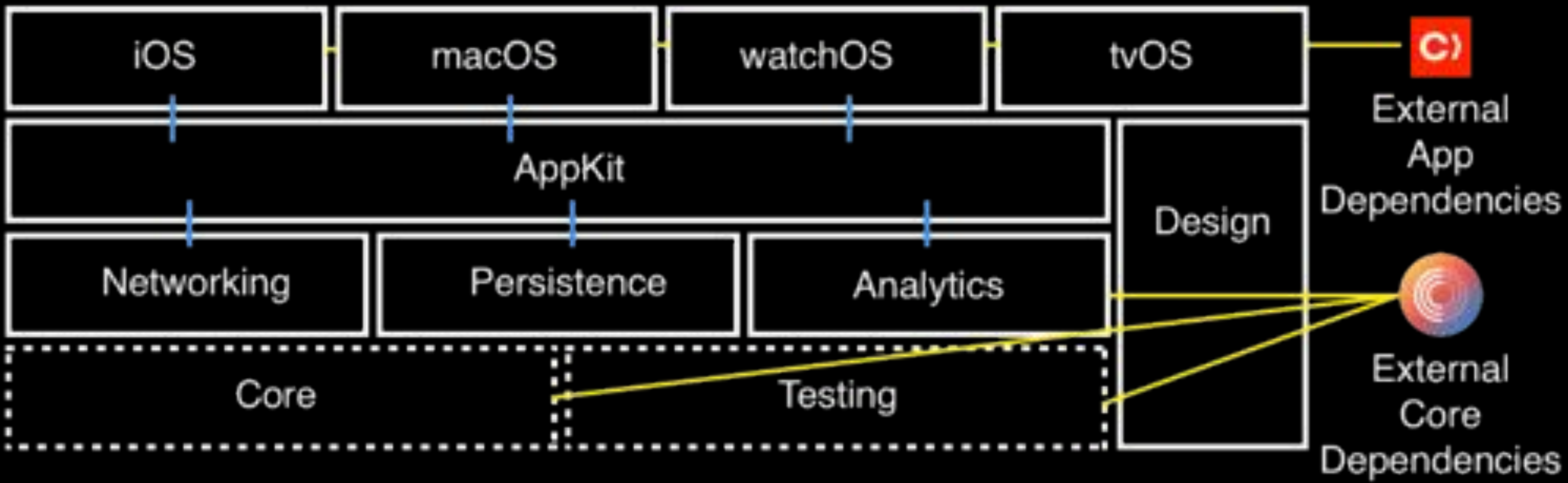
- ✅ More control over the workspace
- ✅ Custom setup (you design it)
- ❌ Cumbersome setup (Build Settings)

External dependencies can be checked out with Carthage/Git Submodules

XCCONFIG

AND MAKING YOUR FRAMEWORK MULTIPLATFORM

Hybrid


iOS
macOS
watchOS
tvOS
AppKit
Design
Networking
Persistence
Analytics
Core
Testing
External App Dependencies
External Core Dependencies

Hybrid

- ✅ CocoaPods resolves/integrates app dependencies
- ✅ Carthage resolves frameworks dependencies
- ✅ Custom stack setup

OPEN QUESTIONS

VERSIONING?

GIT REPO PER FRAMEWORK?

1. Keep it in the same repository (fast iterations)
2. Move it once it consolidates (sporadic changes)
3. Then version it! (snapshots in time)

EXTERNAL DEPENDENCIES?

HOW TO FETCH THEM?

- If CocoaPods for local: Use it also for external
- If manual setup: Use Carthage or Git Submodules

STATIC OR DYNAMIC?

📦

- Objective-C & not shared - *Static*
- Objective-C && shared - *Dynamic*
- Swift - *Dynamic*

HOW MANY DYNAMIC FRAMEWORKS?

THE MORE, THE WORSE LOADING TIME

WILL 🍎 IMPROVE IT? 🤔

- No more than 6 - (WWDC2016:406)
- Group dependencies in Framework (Manual setup)

```
Testing.framework
    Quick.{swift,h,m}
    Nimble.{swift,h,m}
    OHHTTPStubs.{swift,h,m}
Core.framework
    RxSwift{.swift}
```

MIGRATE FROM EXISTING PROJECT?

- Start with *Core/Testing*
- Move *Foundation* components down.
- Continue building layers progressively.

You'll figure out how coupled your code is 😂

DOWNSIDES 😔

LACK OF DOCUMENTATION

(TARGETS CONFIGURATION)

Tip: Use CocoaPods and copy the configuration

STORYBOARDS/XIBS IN FRAMEWORKS

Sucks 😥

TIP: KEEP THEM IN THE APPLICATION TARGET

FRAMEWORKS CODE RECOGNITION

Sucks even more 😭

SOME EXTERNAL DEPENDENCIES

ARE DISTRIBUTED AS PLATFORM BINARIES

```
## XCConfig
LD_RUNPATH_SEARCH_PATHS[sdk=macosx*] = $(inherited) Fabric/OSX
LD_RUNPATH_SEARCH_PATHS[sdk=appletv*] = $(inherited) Fabric/tvOS
LD_RUNPATH_SEARCH_PATHS[sdk=iphone*] = $(inherited) Fabric/iOS
```

SOME EVEN DON'T PROVIDE

BINARIES FOR ALL THE PLATFORMS

PROXY THEM USING MACROS

MACROS!

```
#if !os(watchOS)
  import Fabric
#end

def log(message: String) {
  #if !os(watchOS)
    // Log using Fabric
  #end
}
```

APIS
MIGHT DIFFER BETWEEN PLATFORMS

- NSFetchedResultsController not for macOS
- NSIndexPath for watchOS has no row/section

PROXY THEM

ALSO USING MACROS!

CONCLUSIONS

Very time-saver

FOR MULTI-PLATFORM PROJECTS

AIMS LESS COUPLED CODE

(defined boundaries)

📦

SETUP REQUIRES SOME

Xcode Build Settings knowledge

(UNLESS YOU USE COCOAPODS) 😬

MINIMIZE DEPENDENCIES

6 DEPENDENCIES

(KISS)

USE YOUR COMMONSENSE

WHEN DESIGNING YOUR STACK

USE YOUR COMMONSENSE

Don't be a Javascript developer

📦📦📦📦📦📦📦📦📦📦📦

AND REMEMBER

THE STACK DEPENDS ON YOUR NEEDS

IS IT A COMPANY OR A FREELANCE PROJECT?

IS IT A COMPANY OR A FREELANCE PROJECT?

IS IT A NEW PROJECT?

IS IT A COMPANY OR A FREELANCE PROJECT?
IS IT A NEW PROJECT?
AM I USING ANY DEPENDENCIES TOOL?

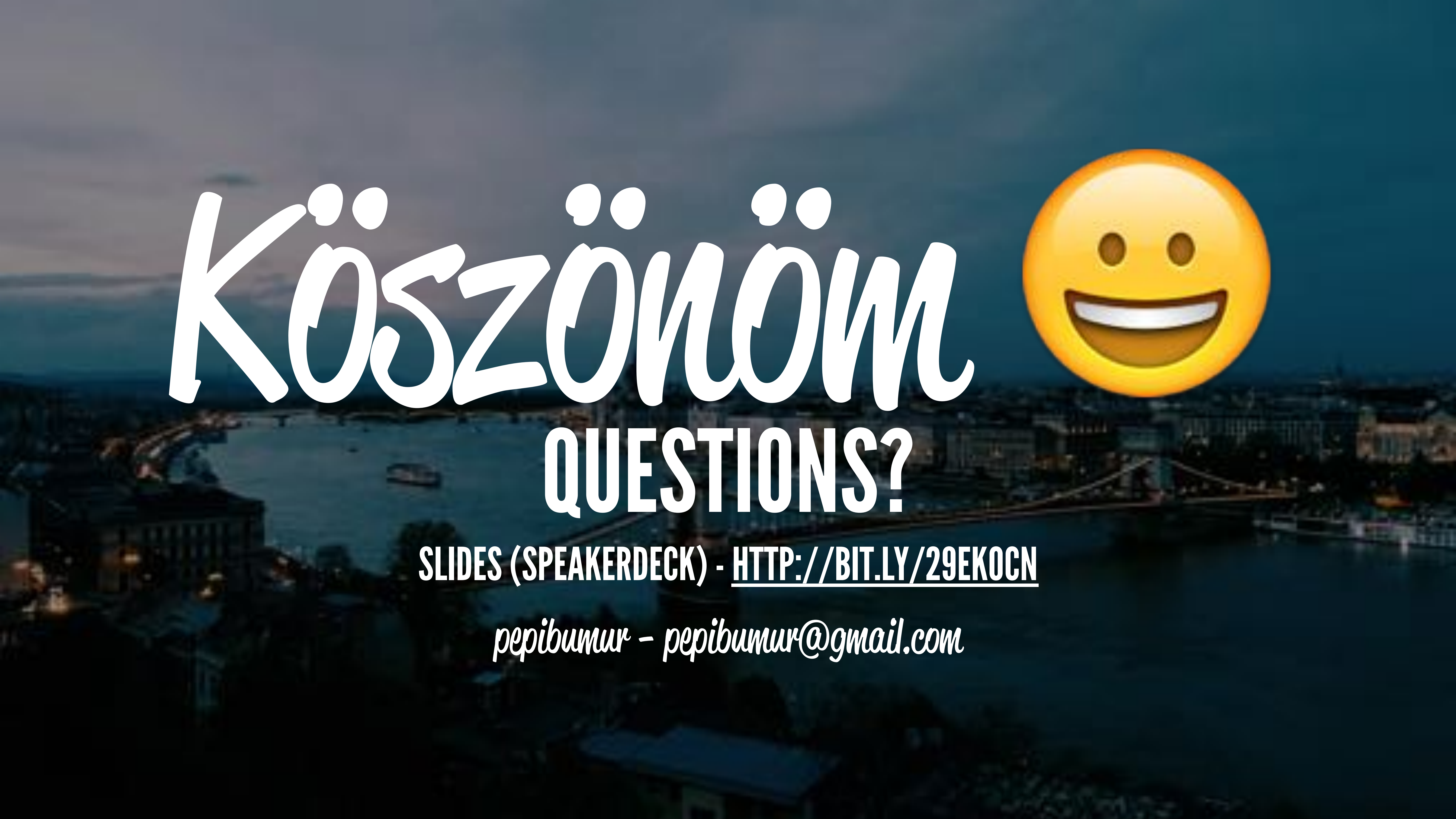
IS IT A COMPANY OR A FREELANCE PROJECT?
IS IT A NEW PROJECT?
AM I USING ANY DEPENDENCIES TOOL?
HOW MANY PEOPLE IN THE TEAM?

REFERENCES

- Library Oriented Programming: Justin Spahr-Summers
- The Unofficial Guide to xcconfig files
- WWDC: Optimizing App Startup Time
- Static & Dynamic libraries
- pepibumur/framework-oriented-programming

CREDITS

📷 FROM UNSPLASH



Köszönöm 😀

QUESTIONS?

SLIDES (SPEAKERDECK) - HTTP://BIT.LY/29EKOCN

pepibumur - pepibumur@gmail.com