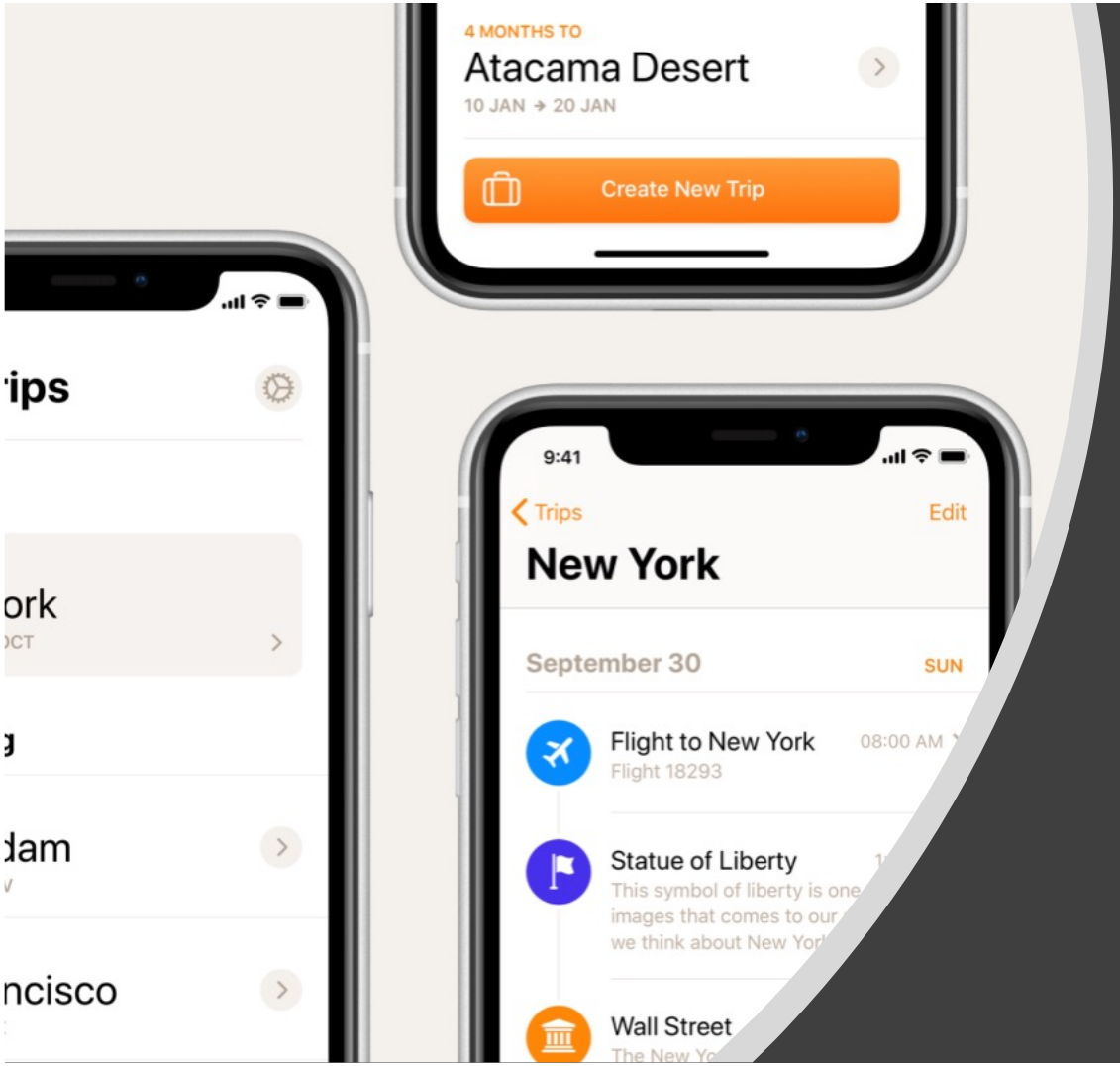


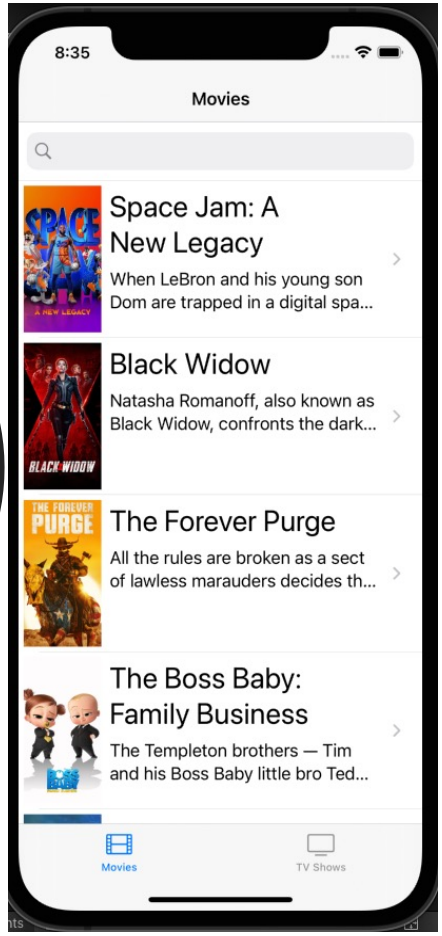
Developing an iOS application

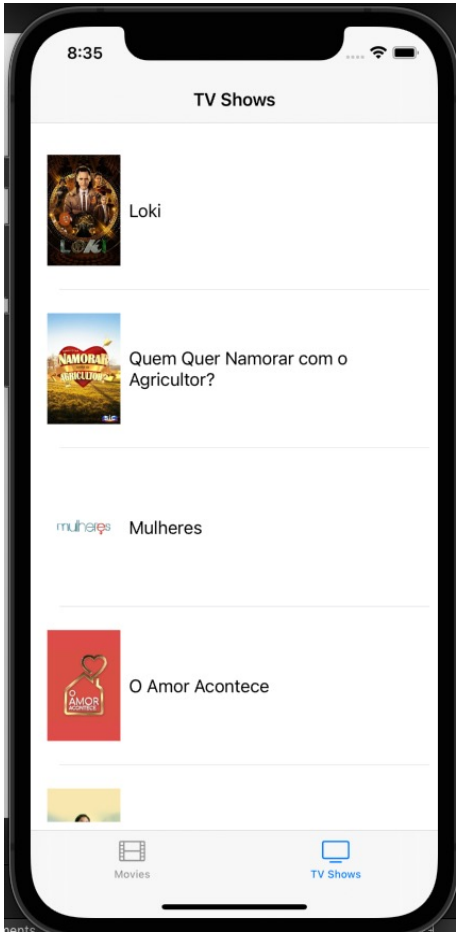
What do I need to know?

In my experience

- Get the idea of the App (creating mockups, storyboard).
- Overview of App Store Review Guidelines - link
- Define the architecture of the code – MVC, MVP, MVVM, VIPER
- Avoid Common App Rejections - link
- Design Guidelines of Apple – link
- Make code!

Mockup of our app

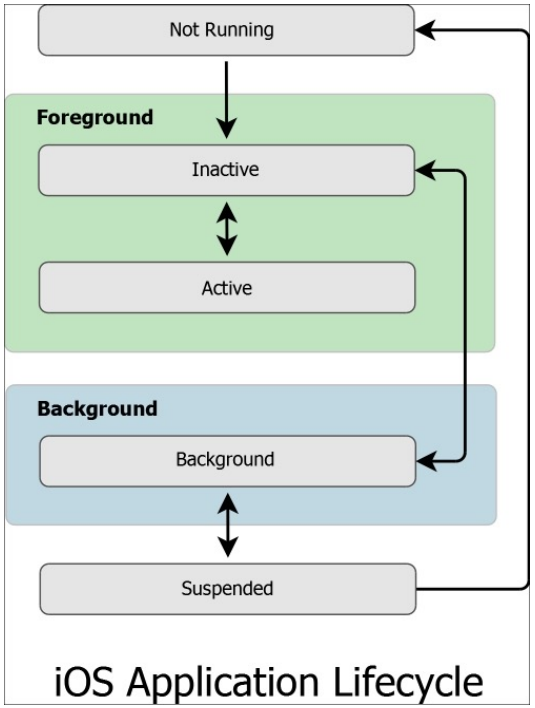


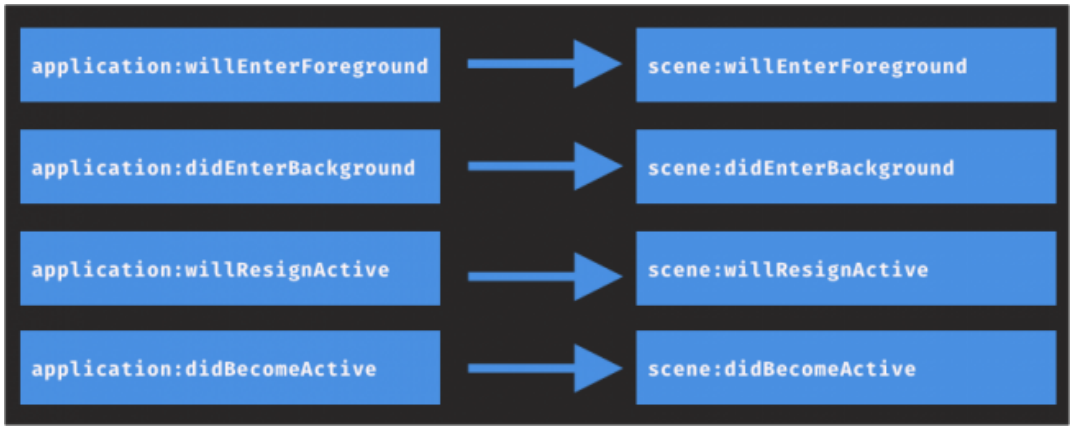


Application basics
Life cycle

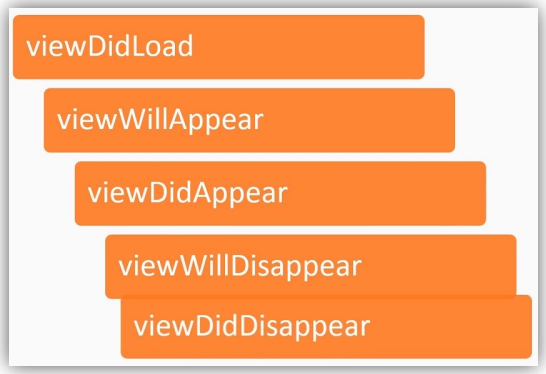
Earlier iOS 13 – AppDelegate

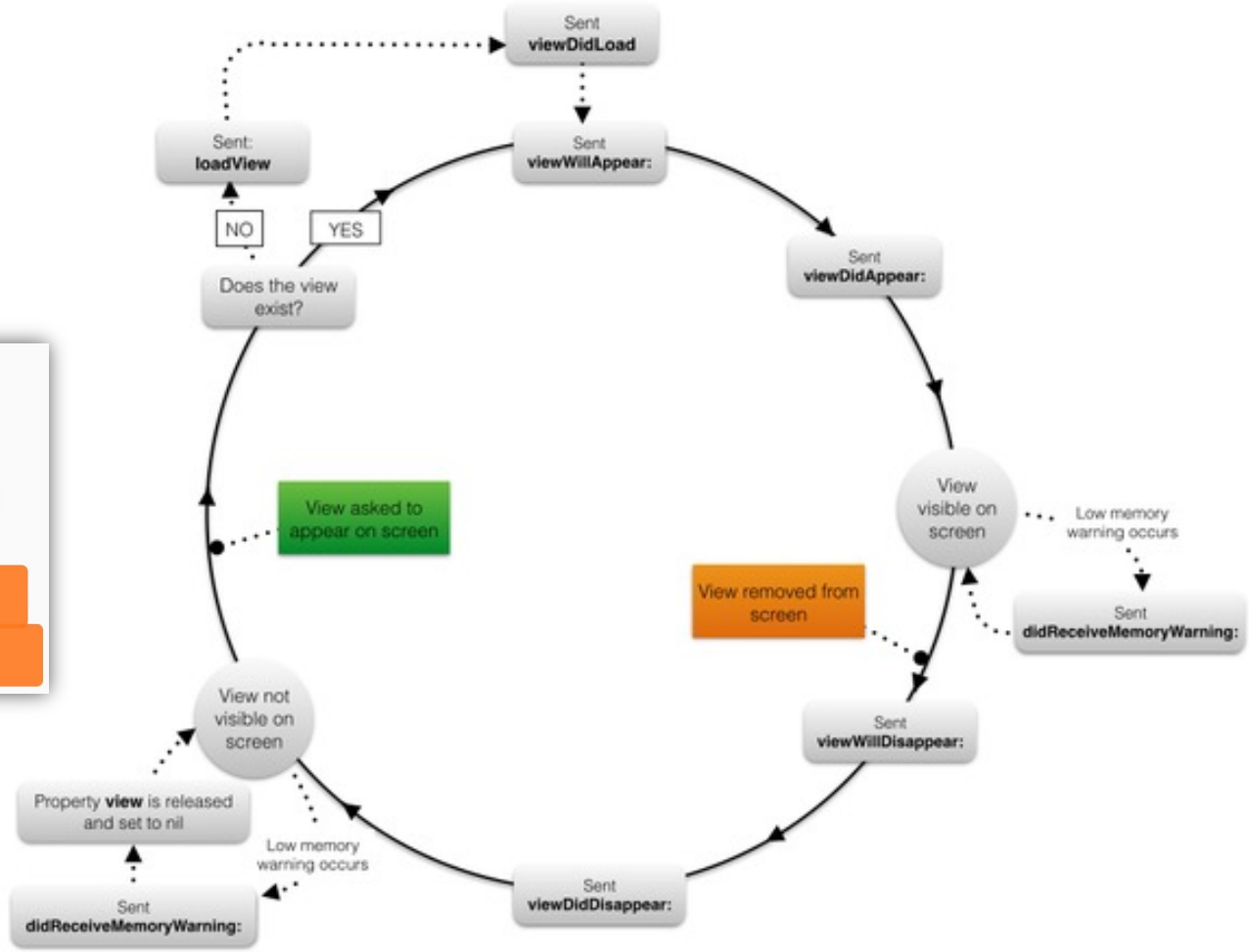
From iOS 13 – SceneDelegate





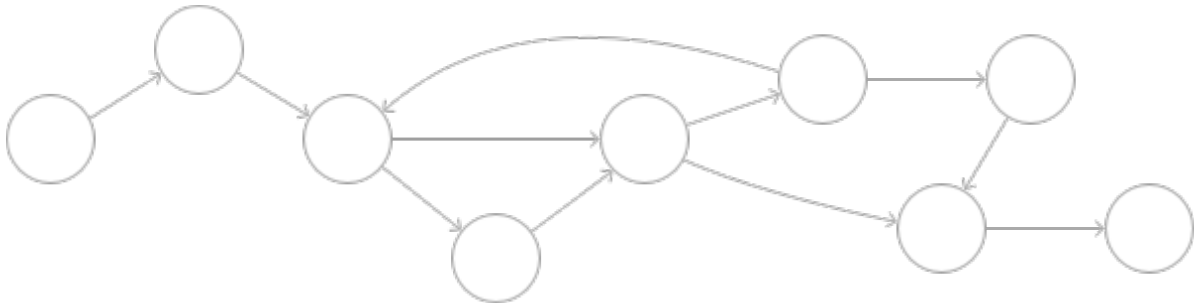
ViewController Life cycle



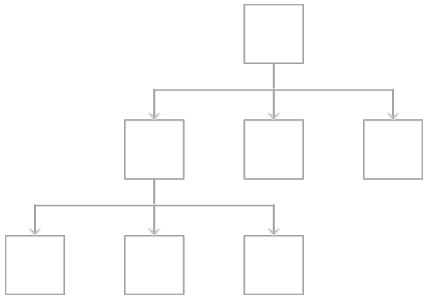


Navigation – iOS Apps

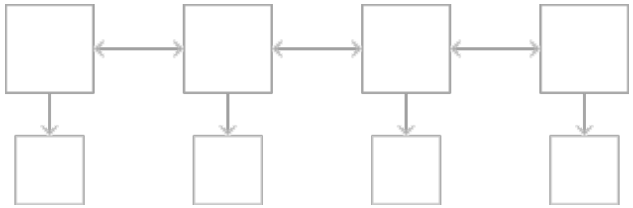
Content-Driven or Experience-Driven Navigation



Hierarchical Navigation

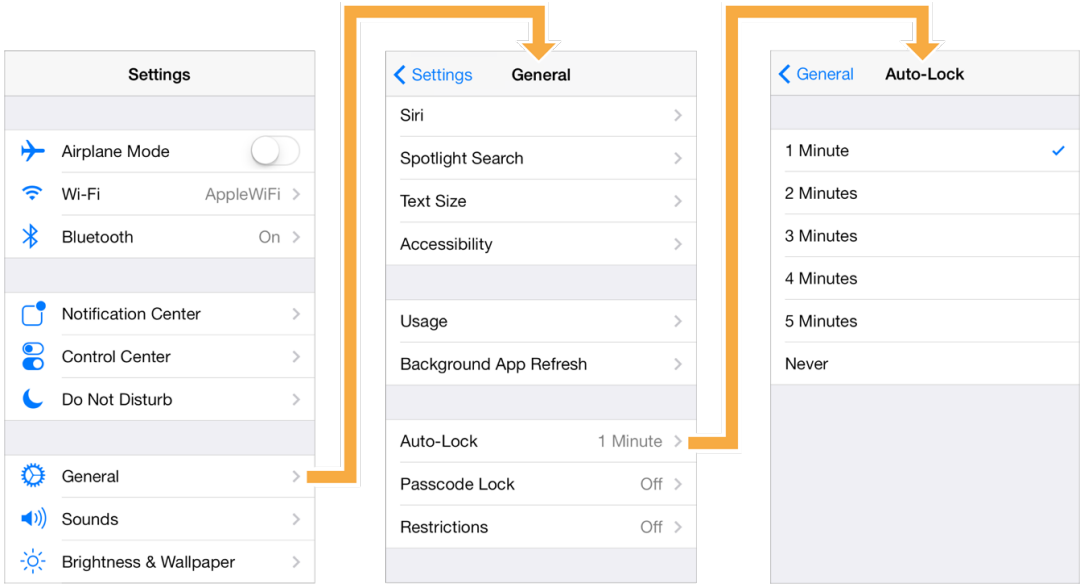


Flat Navigation



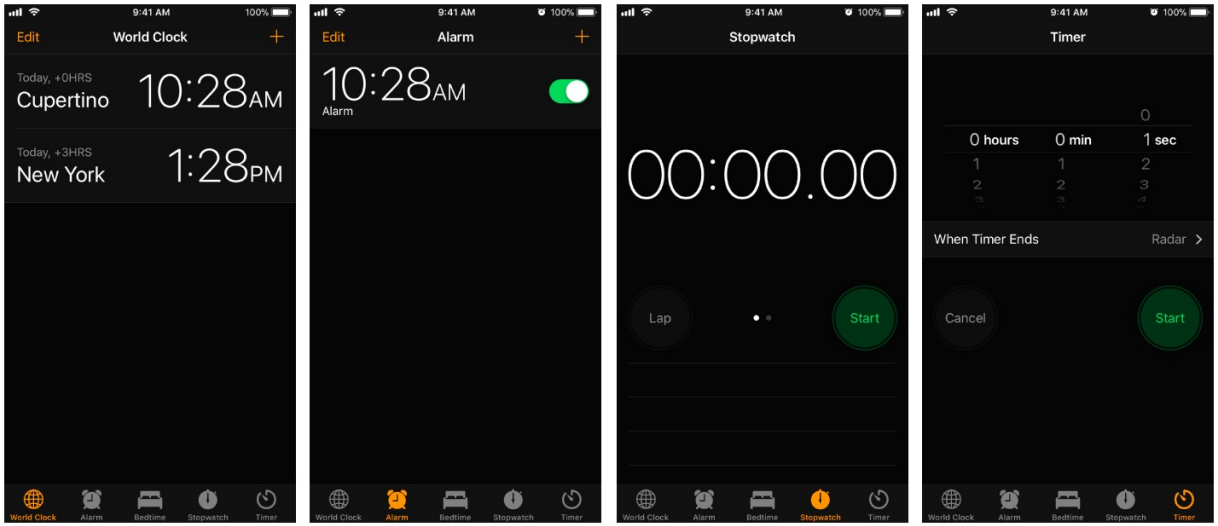
Navigation – Controllers

UINavigationController



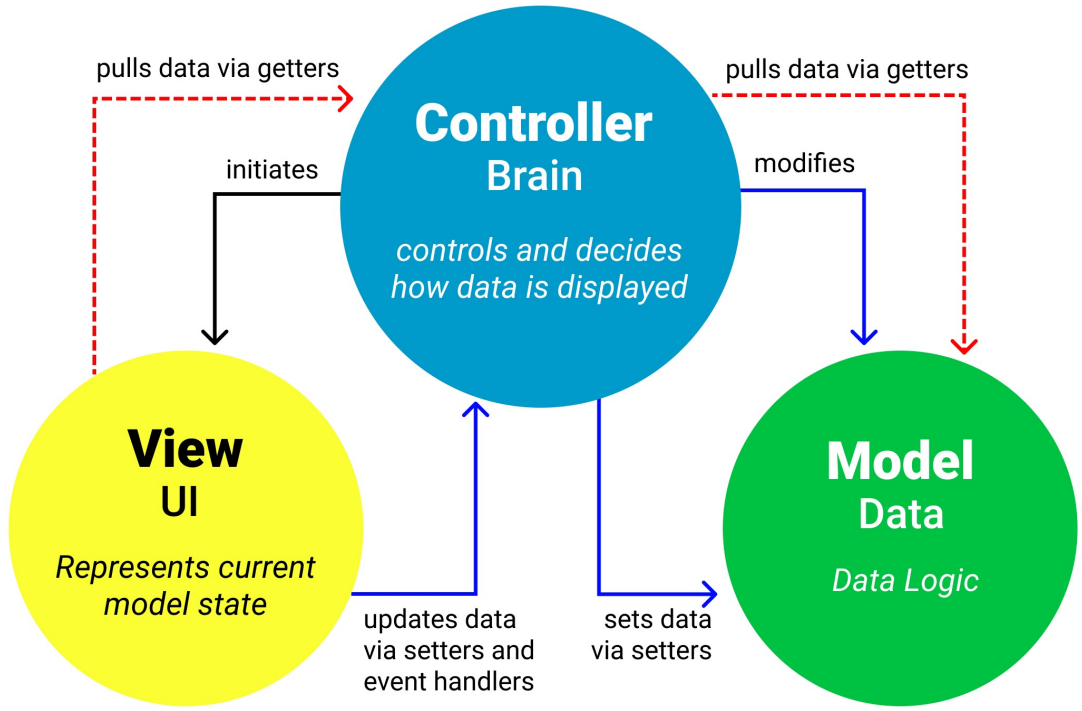
Navigation – Controllers

UITabBarController



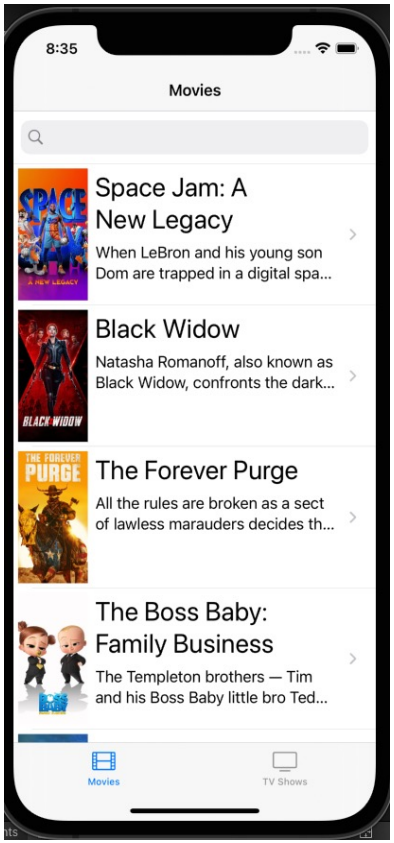
What does architecture we use?

MVC Architecture Pattern



What do natives components we recognize?

UIKit



UINavigationController

UISearchBar

UITableView, UITableViewCell

UIImageView, UILabel

UITabBarController

Where does datasource come?

THE MOVIE DB

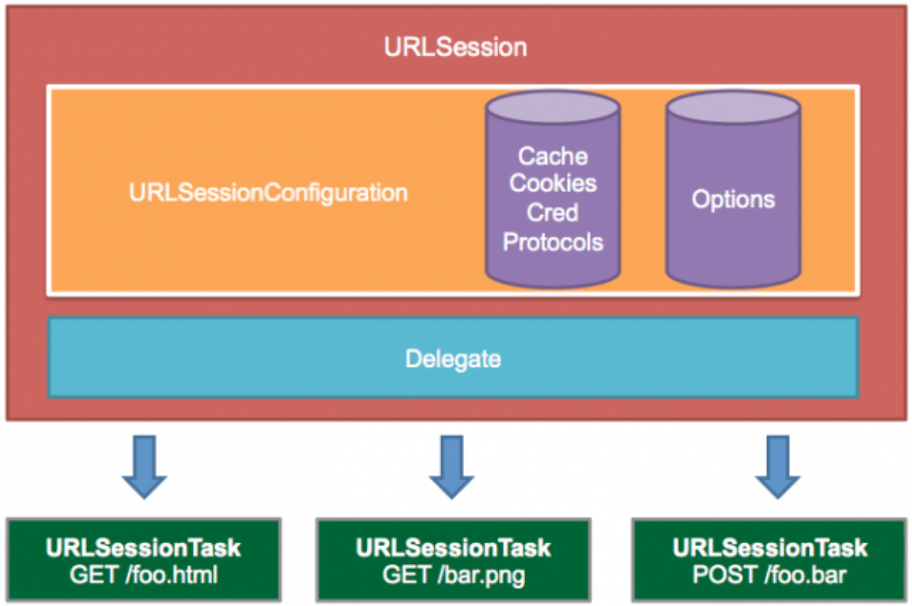
API / REST

JSON format / Model


```
{
    "adult": false,
    "backdrop_path": "/8s4h9friP6Ci3adRGahHARVd76E.jpg",
    "genre_ids": [
        16,
        35,
        10751,
        878
    ],
    "id": 379686,
    "original_language": "en",
    "original_title": "Space Jam: A New Legacy",
    "overview": "When LeBron and his young son Dom are tr
        leading Bugs, Lola Bunny and the whole gang of no
        champions on the court. It's Tunes versus Goons i
    "popularity": 9054.455,
    "poster_path": "/5bFK5d3mVTAvBCXi5NPWH0tYjKl.jpg",
    "release_date": "2021-07-08",
    "title": "Space Jam: A New Legacy",
    "video": false,
    "vote_average": 7.8,
    "vote_count": 1185
},
```

Networking layer

URLSession is a class and a suite of classes for handling HTTP- and HTTPS-based requests.



There are three types of concrete session tasks:

1. URLSessionDataTask: Use this task for GET requests to retrieve data from servers to memory.

2. URLSessionUploadTask: Use this task to upload a file from disk to a web service via a POST or PUT method.

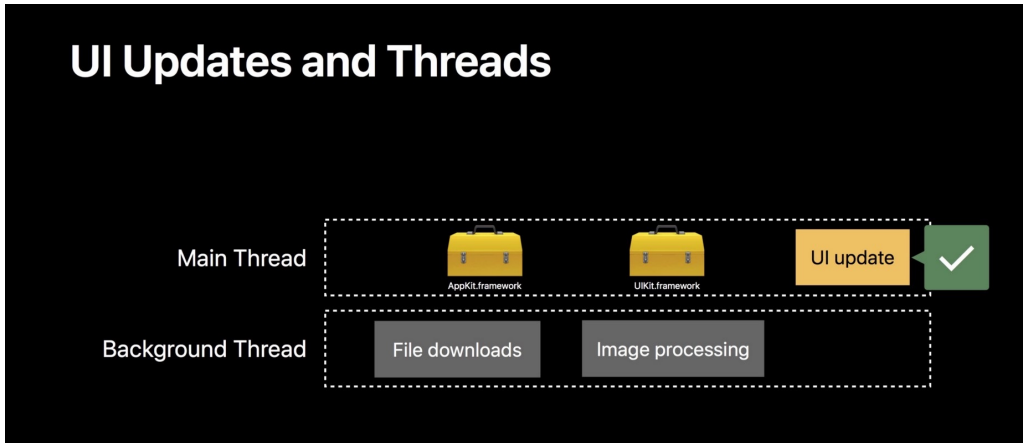
3. URLSessionDownloadTask: Use this task to download a file from a remote service to a temporary file location.

Result type

- Swift's Result type is implemented as an enum that has two cases: success and failure.
- Both are implemented using generics so they can have an associated value of your choosing, **but failure must be something that conforms to Swift's Error type**.
- If you want, you can use a specific error type of your making, such as **NetworkError** or **AuthenticationError**, allowing us to have typed throws for the first time in Swift, but this isn't required.

```
enum Result<Success, Failure> where Failure : Error
```

Threads on iOS



Grand Central Dispatcher
GCD

Execute code concurrently on multicore hardware by submitting work to dispatch queues managed by the system.

```
DispatchQueue.global(qos: .background).async {

    let image = downloadImageFromServer()
    DispatchQueue.main.async {
        self.imageView.image = image
    }
}
```

Writing constraints by using Layout Anchors

First of all, we need to set the:

translatesAutoresizingMaskIntoConstraints = false

Define your constraint

```
let constraints = [
    view.centerXAnchor.constraint(equalTo: superview.centerXAnchor),
    view.centerYAnchor.constraint(equalTo: superview.centerYAnchor),
    view.widthAnchor.constraint(equalToConstant: 100),
    view.heightAnchor.constraint(equalTo: view.widthAnchor)
]
NSLayoutConstraint.activate(constraints)
```

UIView

comes with a collection of anchor properties that allow you to set up relations between views.

Each Anchor it returns subclasses from **NSLayoutAnchor** which comes with a few common methods to set relationships.

These include equal to, greater than, and less than or equal to relationships.

```
extension UIView {

    /* Constraint creation conveniences. See NSLayoutAnchor.h for details.
     */
    open var leadingAnchor: NSLayoutXAxisAnchor { get }

    open var trailingAnchor: NSLayoutXAxisAnchor { get }

    open var leftAnchor: NSLayoutXAxisAnchor { get }

    open var rightAnchor: NSLayoutXAxisAnchor { get }

    open var topAnchor: NSLayoutYAxisAnchor { get }

    open var bottomAnchor: NSLayoutYAxisAnchor { get }

    open var widthAnchor: NSLayoutDimension { get }

    open var heightAnchor: NSLayoutDimension { get }

    open var centerXAnchor: NSLayoutXAxisAnchor { get }

    open var centerYAnchor: NSLayoutYAxisAnchor { get }

    open var firstBaselineAnchor: NSLayoutYAxisAnchor { get }

    open var lastBaselineAnchor: NSLayoutYAxisAnchor { get }
}
```