ARCHITECTURE PATTERN

What is it?

How many do we have?

SOFTWARE ARCHITECTURE DESCRIBES THE DESIGN AND COLLECTION OF COMPONENTS INTO SYSTEMS THAT MAKE UP THE BUILDING BLOCKS OF SOFTWARE.
SOFTWARE ARCHITECTURE EXPLAINS THE STRUCTURAL COMPOSITION OF THE SOFTWARE PROGRAM AND THE INTERACTIONS BETWEEN THE ELEMENTS.

source: redhat

What is an architectural pattern?

WHICH ARE ARCHITECTURES IN IOS?

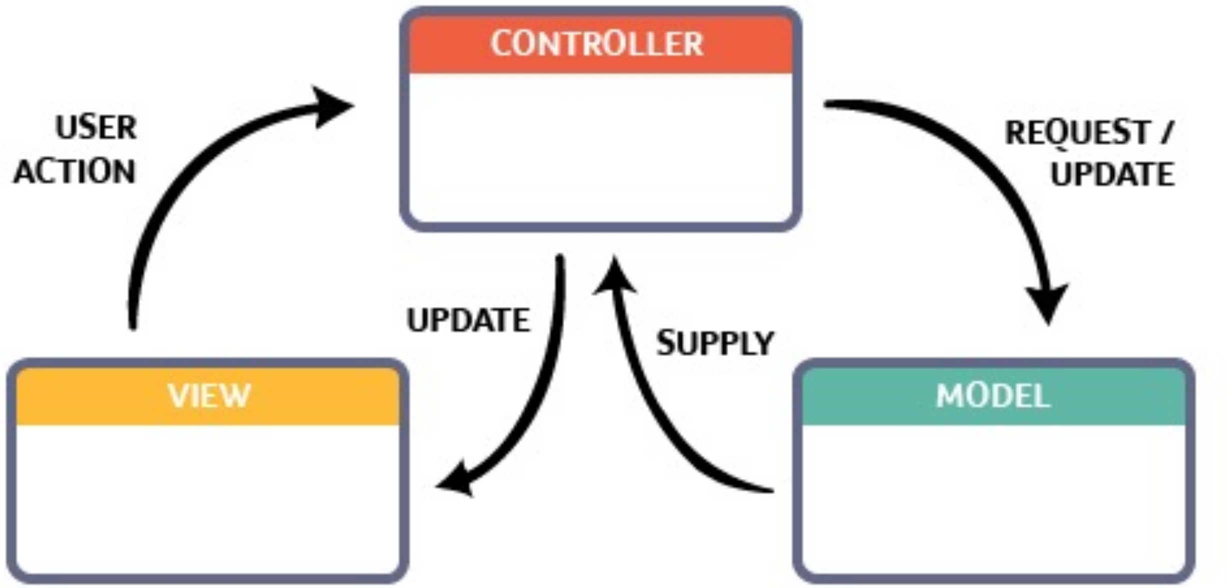
There are many!

- Model View Controller – MVC
- Model View Presenter - MVP
- Model View View Model - MVVM
- View Interactor Presenter Entity Routing – VIPER

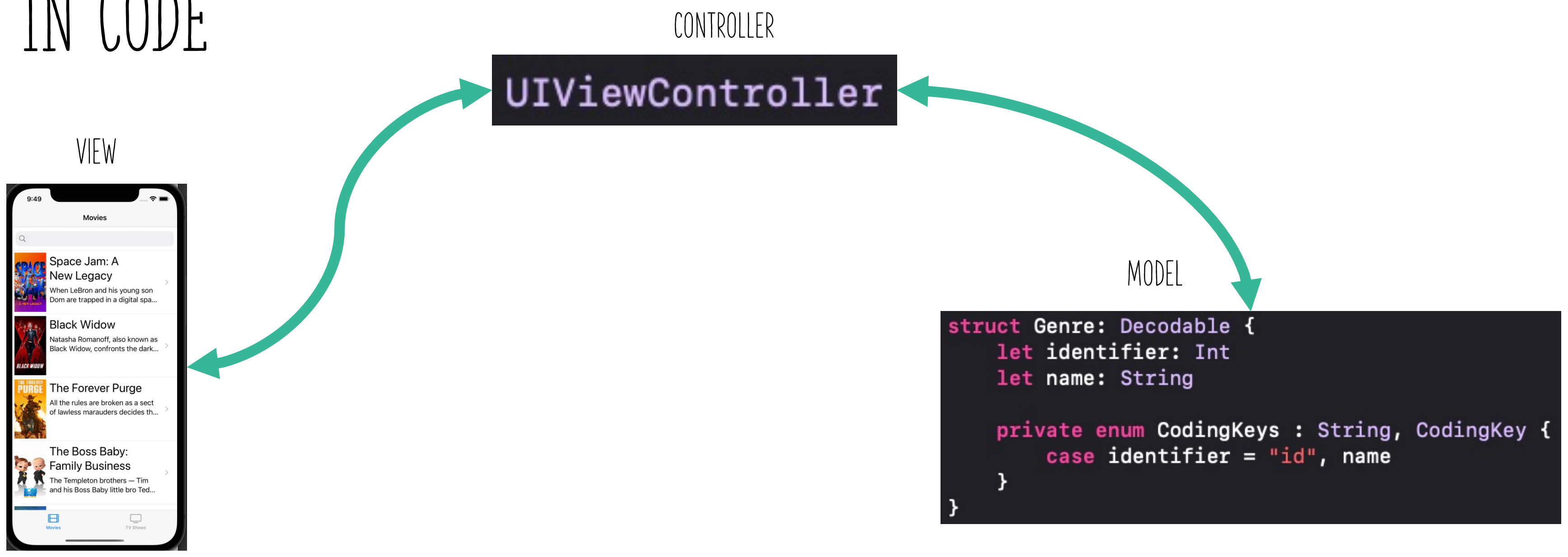
WHEN DO ARCHITECTURES START?

Since iOS's introduction in March 2008, almost 13 years ago, the core iOS MVC architecture hasn't materially changed.

core iOS MVC



MVC IN CODE



9:49

Movies

Space Jam: A New Legacy
When LeBron and his young son Dom are trapped in a digital spa...

Black Widow
Natasha Romanoff, also known as Black Widow, confronts the dark...

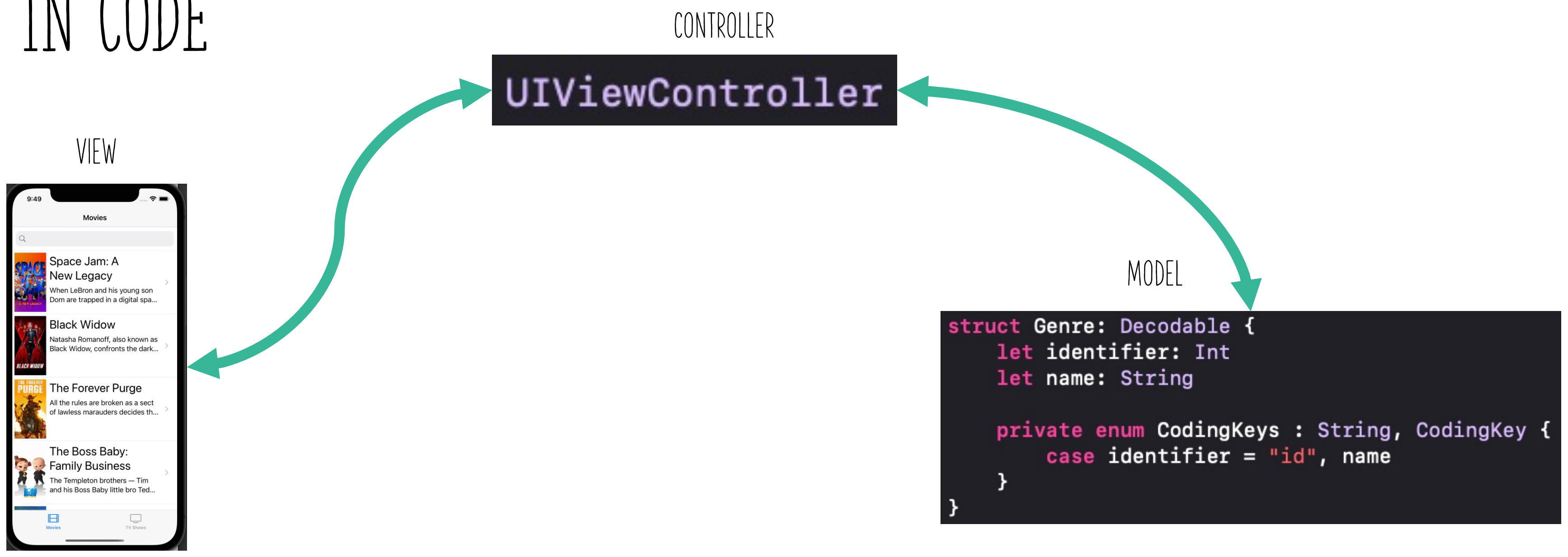
The Forever Purge
All the rules are broken as a sect of lawless marauders decides th...

The Boss Baby: Family Business
The Templeton brothers — Tim and his Boss Baby little bro Ted...

```swift
struct Genre: Decodable {
    let identifier: Int
    let name: String

    private enum CodingKeys : String, CodingKey {
        case identifier = "id", name
    }
}
```

MVC IN CODE



9:49
Movies

Space Jam: A New Legacy
When LeBron and his young son Dom are trapped in a digital spa…

Black Widow
Natasha Romanoff, also known as Black Widow, confronts the dark…

The Forever Purge
All the rules are broken as a sect of lawless marauders decides th…

The Boss Baby: Family Business
The Templeton brothers — Tim and his Boss Baby little bro Ted…

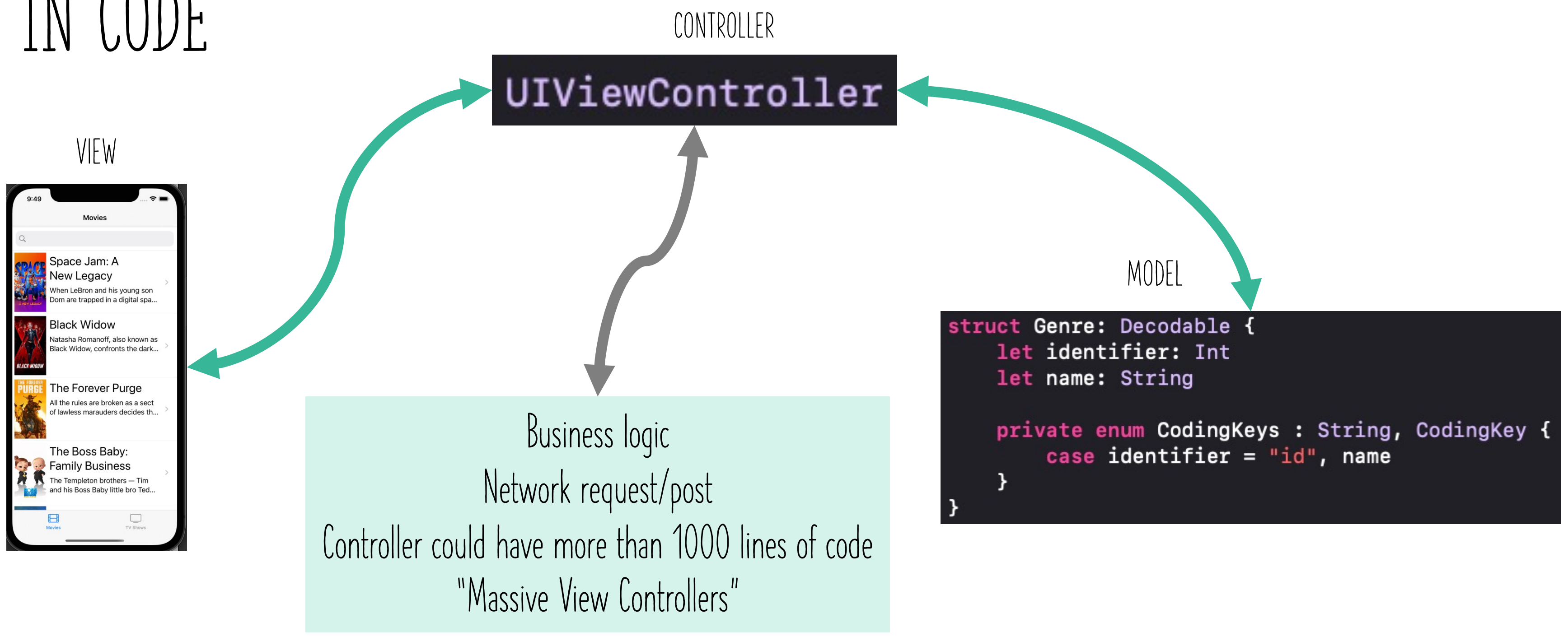
Movies
TV Shows

```
struct Genre: Decodable {
    let identifier: Int
    let name: String

    private enum CodingKeys : String, CodingKey {
        case identifier = "id", name
    }
}
```

Over the years, as applications have become more complex, the limits and shortcomings of MVC and UIKit have prompted developers to develop and use alternative architectures

MVC IN CODE



Space Jam: A New Legacy
When LeBron and his young son Dom are trapped in a digital spa...

Black Widow
Natasha Romanoff, also known as Black Widow, confronts the dark...

The Forever Purge
All the rules are broken as a sect of lawless marauders decides th...

The Boss Baby: Family Business
The Templeton brothers — Tim and his Boss Baby little bro Ted...

```swift
struct Genre: Decodable {
    let identifier: Int
    let name: String

    private enum CodingKeys : String, CodingKey {
        case identifier = "id", name
    }
}
```

Business logic

Network request/post

Controller could have more than 1000 lines of code

"Massive View Controllers"

Over the years, as applications have become more complex, the limits and shortcomings of MVC and UIKit have prompted developers to develop and use alternative architectures

WE CAN SEE THAT THERE ARE TWO LAYERS THAT WE CAN'T TOUCH

VIEW

MODEL

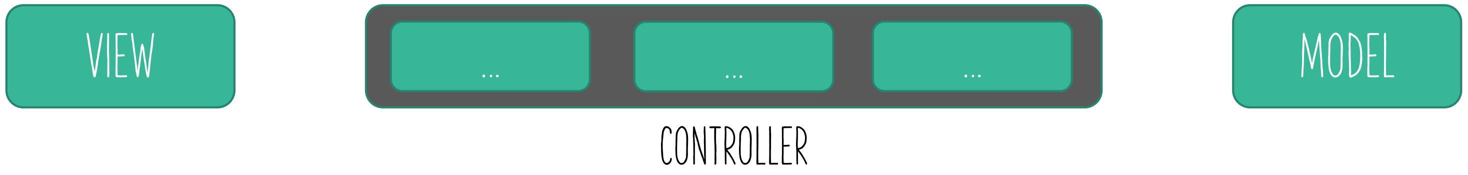
WE CAN SEE THAT THERE ARE TWO LAYERS THAT WE CAN'T TOUCH

VIEW

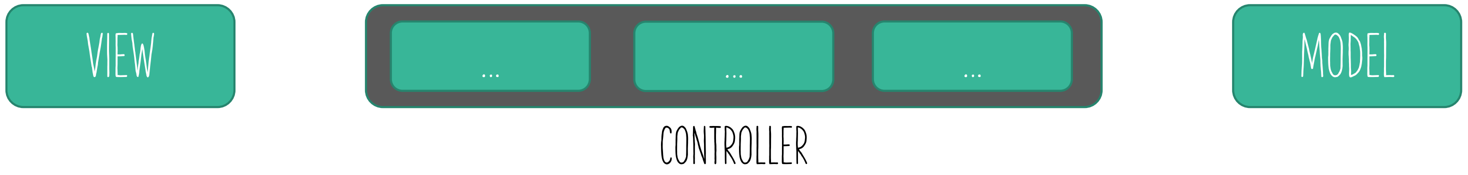
CONTROLLER

MODEL

WE CAN SEE THAT THERE ARE TWO LAYERS THAT WE CAN'T TOUCH



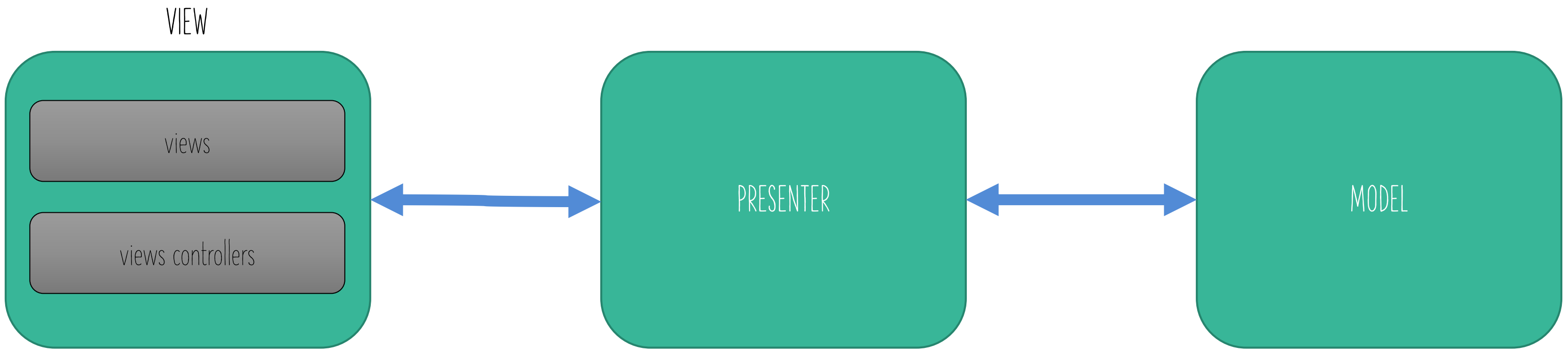
WE CAN SEE THAT THERE ARE TWO LAYERS THAT WE CAN'T TOUCH



- Model View Presenter - MVP
- Model View View Model - MVVM
- View Interactor Presenter Entity Routing – VIPER

MODEL VIEW PRESENTER

We add an extra layer "PRESENTER"

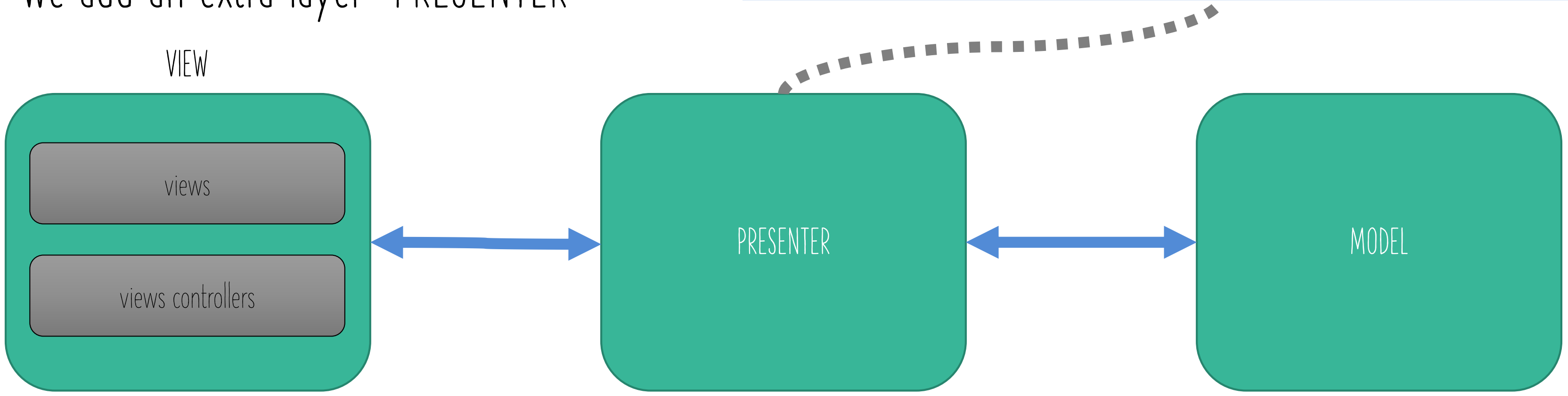


•You can control the view from the presenter layer.
•When something happens in the view layer, for example when the user initiates an action, it is communicated to the model through the Presenter.
•When the model is changed, for example when new data is made available and we need to update the UI, the Presenter updates the View.

MODEL VIEW PRESENTER

We add an extra layer "PRESENTER"

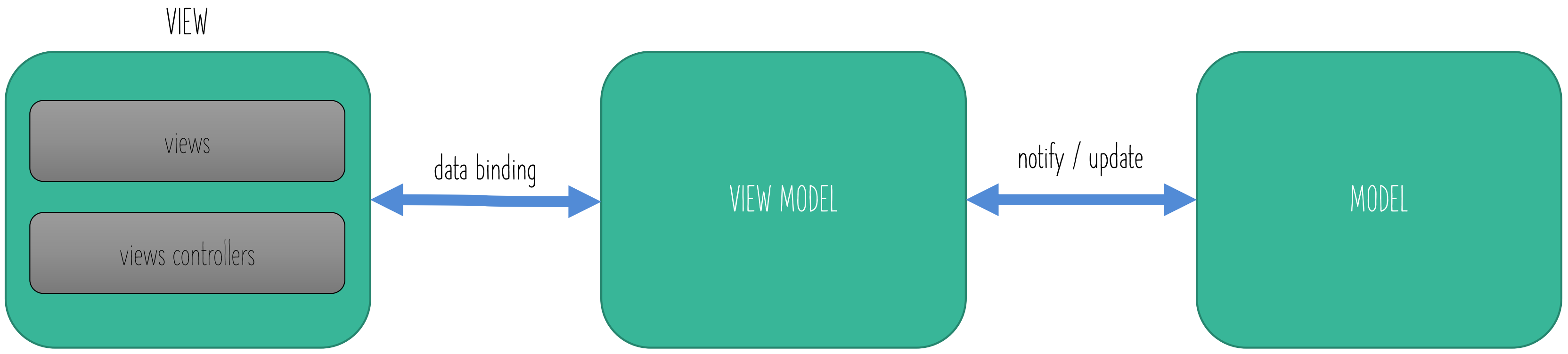
•Responsible for handling the events coming from the view and triggering the appropriate events with the Model.
•Connects the View with the Model, but without any logic added to the View.
•Has a 1:1 mapping to a View.



•You can control the view from the presenter layer.
•When something happens in the view layer, for example when the user initiates an action, it is communicated to the model through the Presenter.
•When the model is changed, for example when new data is made available and we need to update the UI, the Presenter updates the View.

MODEL VIEW VIEW MODEL

We change presenter to "VIEW MODEL"

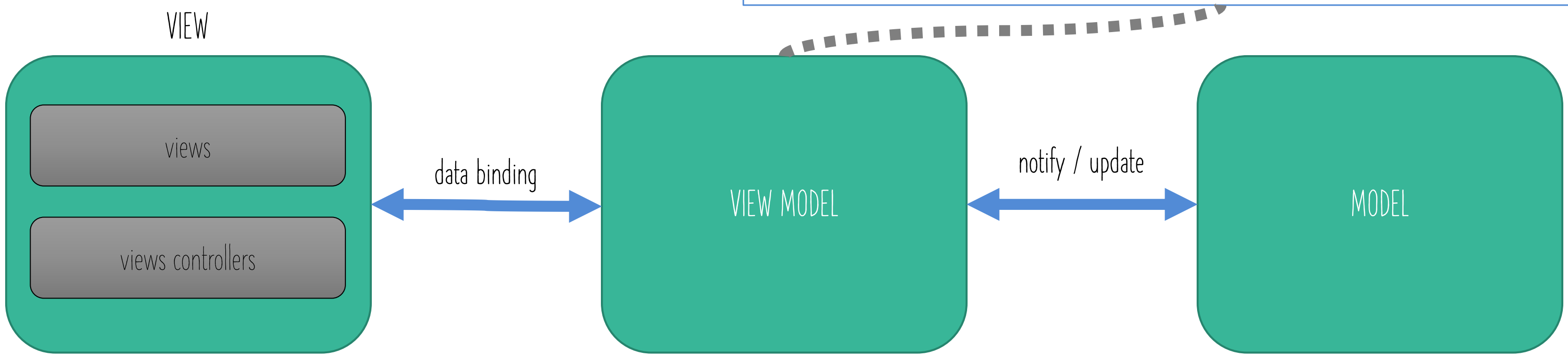


- •The presenter Layer is replaced by the ViewModel Layer and the communication between the View and the ViewModel done through Data Binding approach
- •The View (UI) responds to user input by passing input data (defined by the Model) to the ViewModel. In turn, the ViewModel evaluates the input data and responds with an appropriate UI presentation according business logic workflow.

MODEL VIEW VIEW MODEL

We change presenter to "VIEW MODEL"

•Data Binding paradigms used is Reactive Programming, RxSwift, **Combine.**
•The ViewModel should represent the View's current state at any time, for example if we have two UITextFields in the View the ViewModel should have two strings properties which represents those views elements

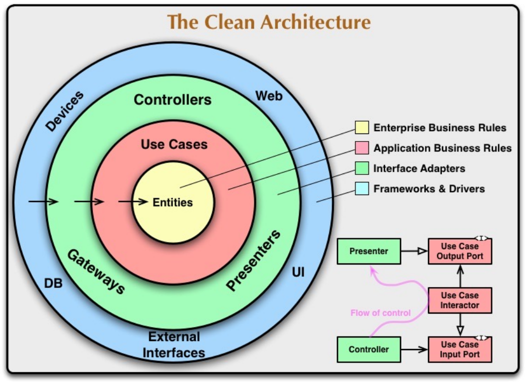


•The presenter Layer is replaced by the ViewModel Layer and the communication between the View and the ViewModel done through Data Binding approach
•The View (UI) responds to user input by passing input data (defined by the Model) to the ViewModel. In turn, the ViewModel evaluates the input data and responds with an appropriate UI presentation according business logic workflow.

VIPER

VIPER has been used to build many large projects.

VIPER is an application of Clean Architecture to iOS apps.



Clean Architecture divides an app's logical structure into distinct layers of responsibility.

Application Design Based on Use Cases

Use cases are also known as acceptance criteria, or behaviors, and describe what an app is meant to do, for example, a list needs to be sortable by date, type, or name. That's a use case.

A use case is the layer of an application that is responsible for business logic.

Use cases should be independent from the user interface implementation of them

VIPER

View : displays what it is told to by the Presenter and relays user input back to the Presenter.

Interactor : contains the business logic as specified by a use case.

Presenter : contains view logic for preparing content for display (as received from the Interactor) and for reacting to user inputs (by requesting new data from the Interactor).

Entity : contains basic model objects used by the Interactor.

Routing : contains navigation logic for describing which screens are shown in which order.

VIPER

Application Design Based on Use Cases

View, Interactor, Presenter, Entity, Routing

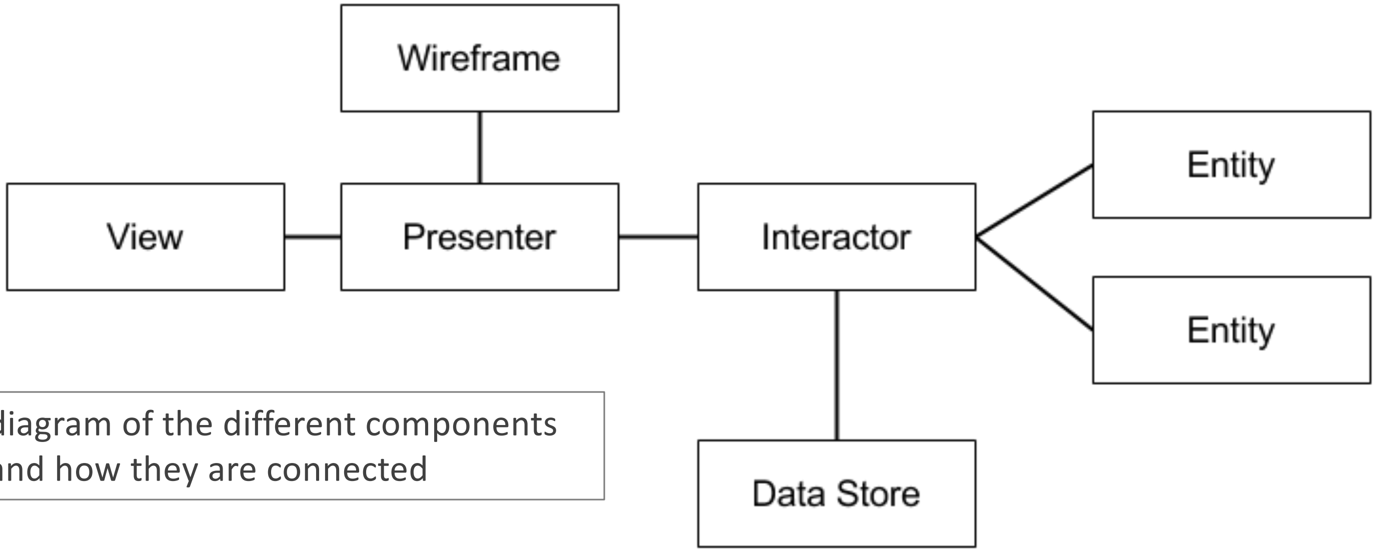


diagram of the different components and how they are connected

CONCLUSION

You can choose any architectures but always considered these characteristics: separation of Concerns, unidirectional data flow, "model" immutability.

Always you must have in mind the SOLID principles when you use any architecture.

CONCLUSION

You can choose any architectures but always considered these characteristics: separation of Concerns, unidirectional data flow, "model" immutability.

Always you must have in mind the SOLID principles when you use any architecture.

MY THOUGHTS

FUTURE

MVVM
MODEL VIEW VIEWMODEL

SwiftUI

Combine

```swift
print("THANK YOU!")
```

SOURCES

https://medium.com/backticks-tildes/the-s-o-l-i-d-principles-in-pictures-b34ce2f1e898

https://medium.com/@damonallison/book-review-app-architecture-ios-application-patterns-in-swift-39b5753ebae7

https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

https://www.radude89.com/blog/mvp.html

https://www.objc.io/issues/13-architecture/viper/

https://github.com/objcio/issue-13-viper

https://github.com/kitasuke/SwiftUI-MVVM

https://github.com/yokurin/Swift-VIPER-iOS

https://github.com/infinum/iOS-VIPER-Xcode-Templates

Questions?