VIPER

iOS Architecture

Pablo Lerma Martínez
@pablo_lerma

Why do we need an architecture?

UIViewController


App analysis
Interface flow
Responsive interfaces
Animations
Local persistence
Gestures
Server API
Business logic
© 2009 Daniel Austin Hoherd

Massive View Controller

```
@interface MassiveViewController () <UITableViewDelegate,
UITableViewDataSource, UIAlertViewDelegate,
UITextFieldDelegate,  CustomButtonDelegate,
CustomViewDelegate,  OtherCustomViewDelegate,
AnotherRandomDelegate, YouGetThePointDontYouDelegate>
```

∞ imports

∞ LOC

Massive View Controller

Consequences?

Consequences

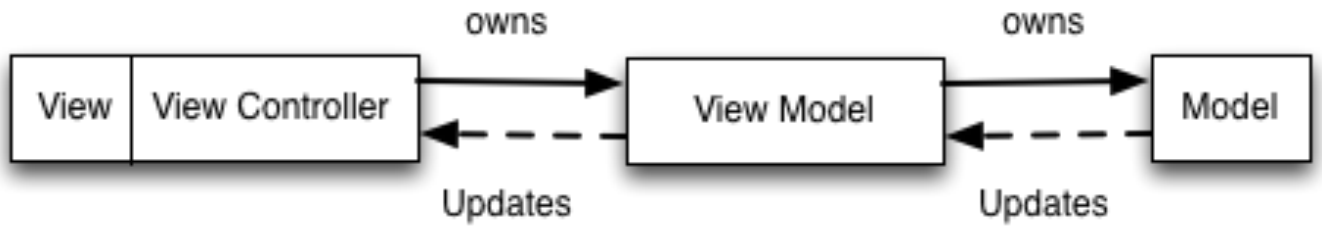
- Multi-delegation
- Over-responsibility
- Monster files
- Impossible to test
- Components too coupled
- Hard to understand /review classes

What are our options?

MVVM

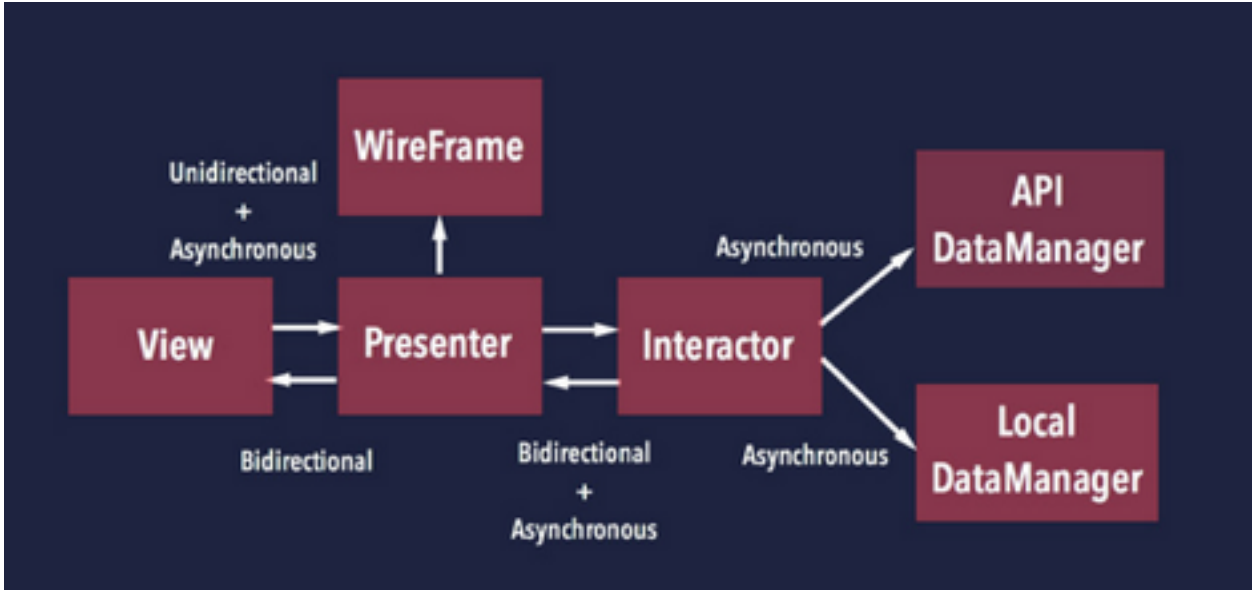
Model-View-ViewModel

MVVM


owns
owns
View
View Controller
View Model
Model
Updates
Updates

VIPER

- V(iew) I(nteractor) P(resenter) E(ntity) R(outer)



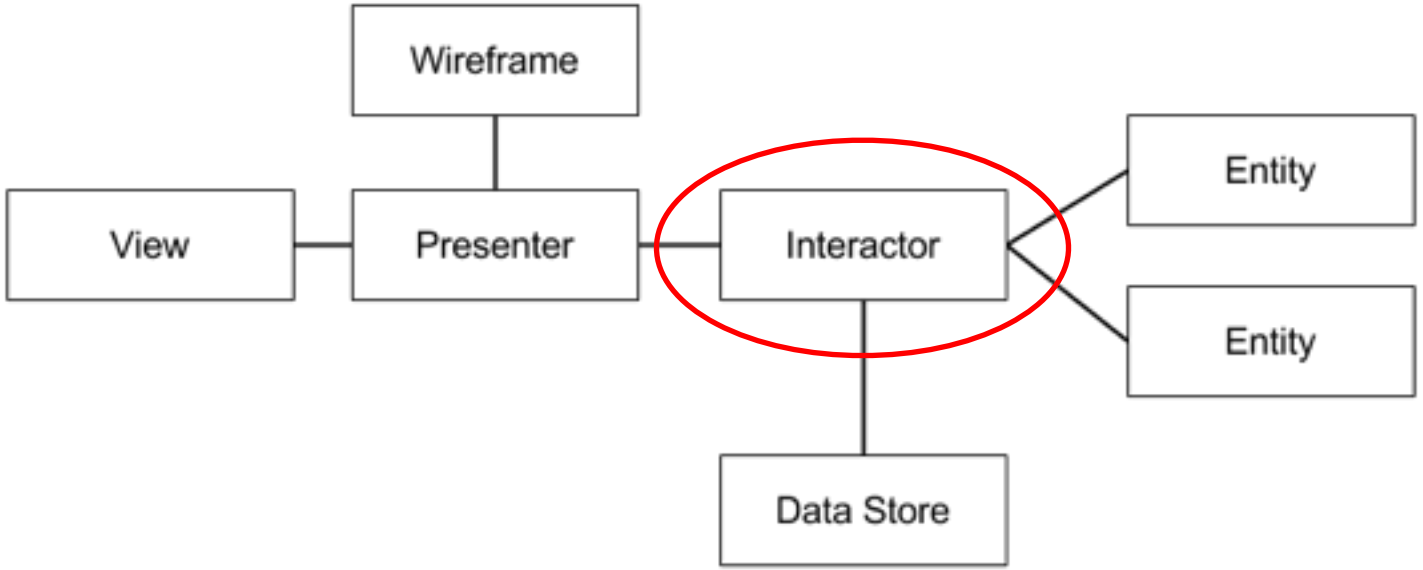
Single responsibility principle

"A class or module should have one,

and only one, reason to change"

- Robert C. Martin

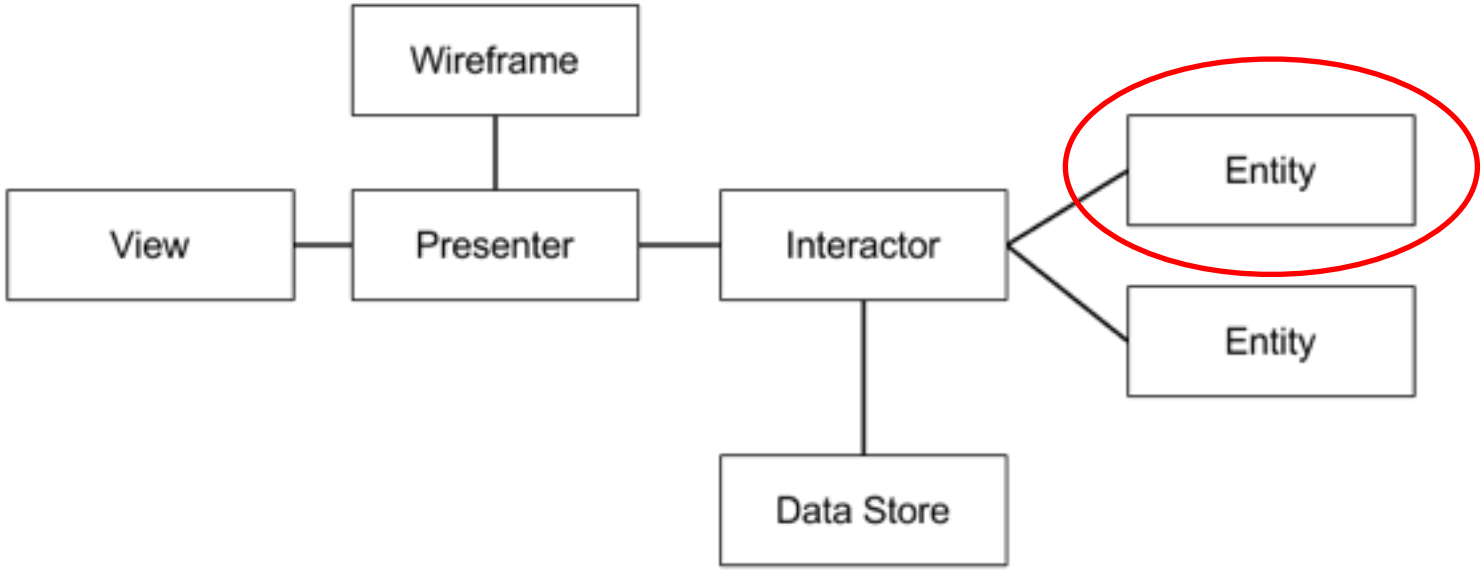
Interactor


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

Interactor

- Represents a single use case in the app
- It contains the business logic
- Fetches Entities from the Data Store
- Pass and receive models from the Presenter
- UI independent
- PONSO
  - TDD
- Same Interactor could be used in an iOS / OS X app

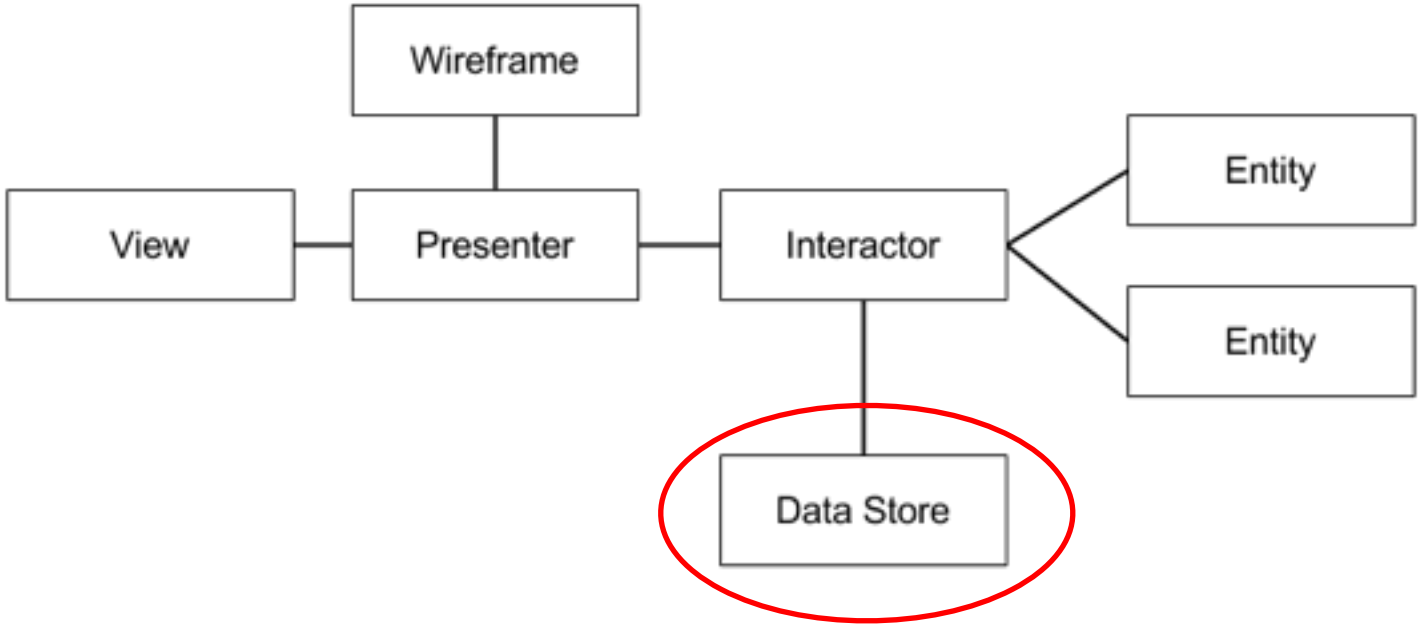
Entity


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

Entity

- Contains basic model objects used by the Interactor
- Created by the Data Store
- They are not managed. Entities do not know how to persist themselves
- Tend to be PONSOs

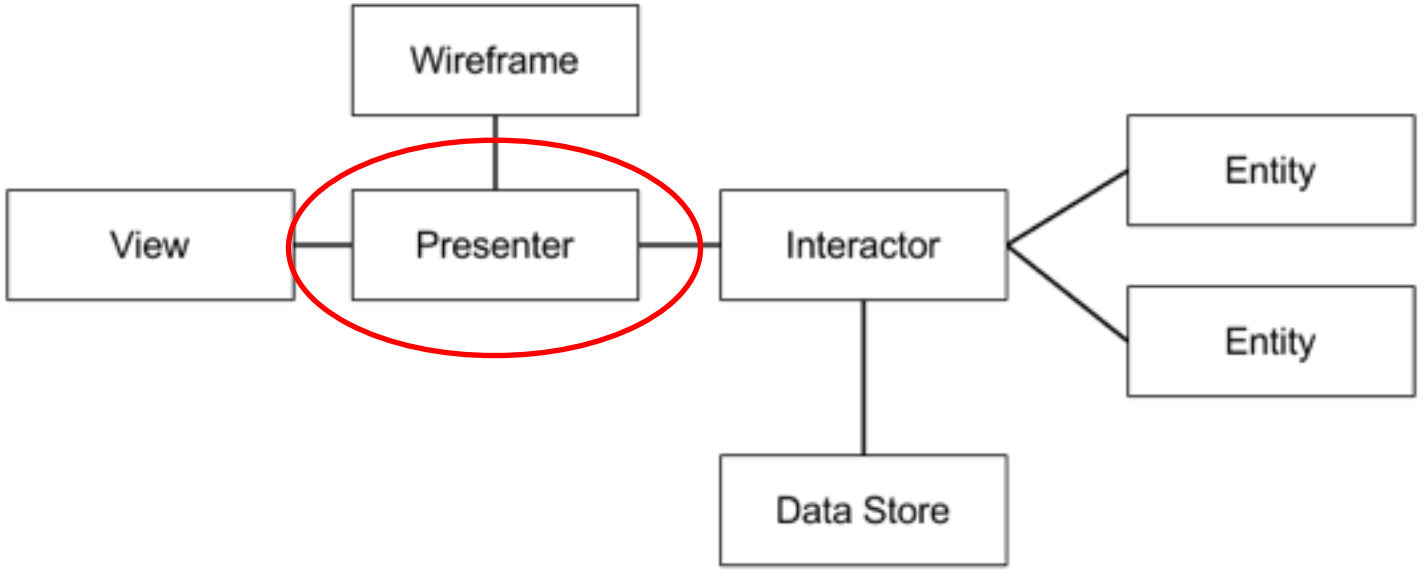
Data Store


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

Data Store

- Responsive for presenting Entities to the Interactor
- All persistence decisions are made here
- Can be replaced by a test double (mock)
- Can be independently tested

Presenter


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

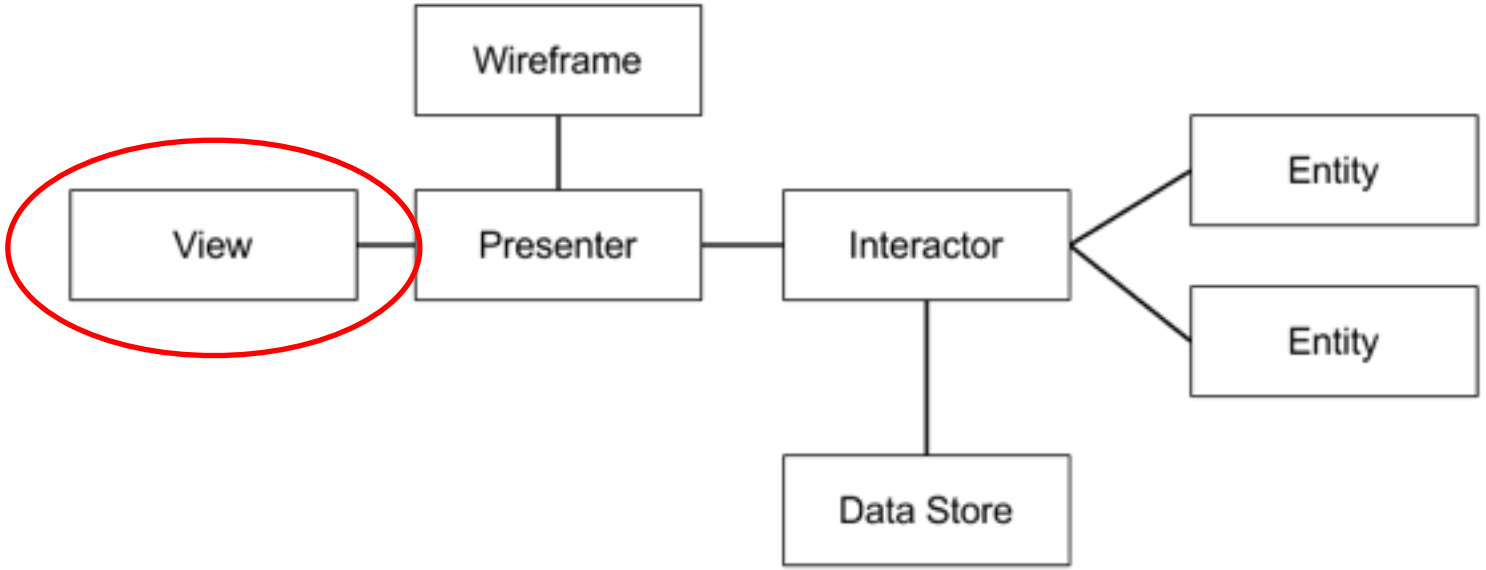
Presenter

- Contains view logic for preparing content for display
- React to user inputs
- Sends requests to the Interactor
  - with data entered in the View
  - for data to be presented in the View
- Sends requests to the Wireframe for UI transitions

Presenter

- Entities are never passed from the Interactor to the Presenter, only simple data structures
- PONSOs

View


Wireframe
View
Presenter
Interactor
Entity
Entity
Data Store

View

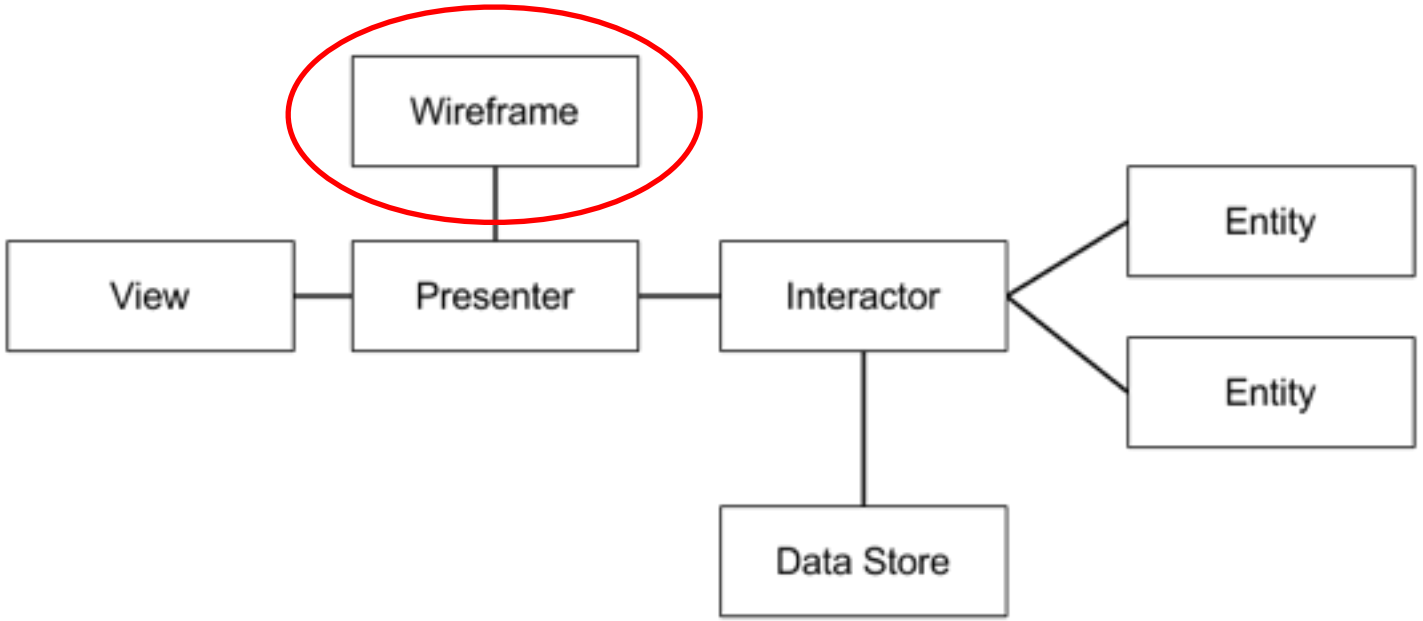
- UIKit components (UILabel, UITableView, …)
- Presents a public interface that can be used to drive the UI
  - Views are completely passive
  - Implemented as public methods
- Defines an event interface triggered by IBActions
  - Implemented as @protocols methods
- The VC implements these two interfaces

View

```
@protocol LoginView <NSObject>
- (void)setUserName:(NSString*)userName;
- (void)setPassword:(NSString*)password;
- (void)setLoginEnabled:(BOOL)enabled;
@end

@interface LoginViewController : UIViewController <LoginView>
...
@end
```

Routing / Wireframe



Routing / Wireframe

- Initializes the VIPER module
- Owns the UIWindow, UINavigationController and all UIViewControllers
- Presents an action interface for the Presenter
- Responsive for transition animations

VIPER isn't a **panacea**

What will we need to know?

- Patterns
- Protocols
- Delegates
- Data sources
- KVO
- Dependency Injection

General Tips

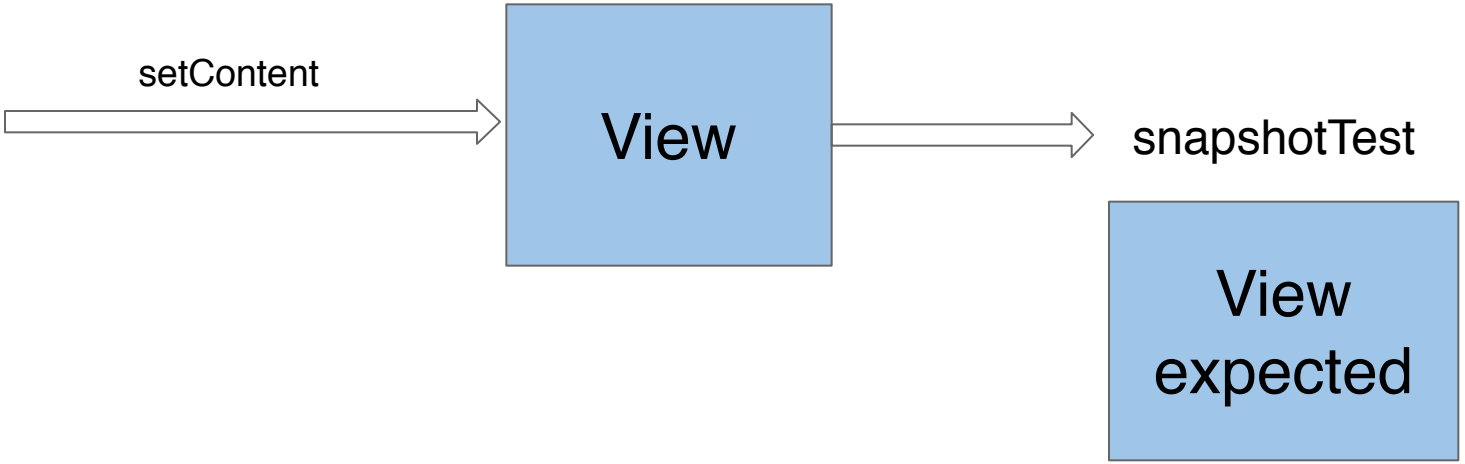
- Think before code
- Sketch UML
  - Specially the protocols
- Code Reviews
- Unit test

General Tips

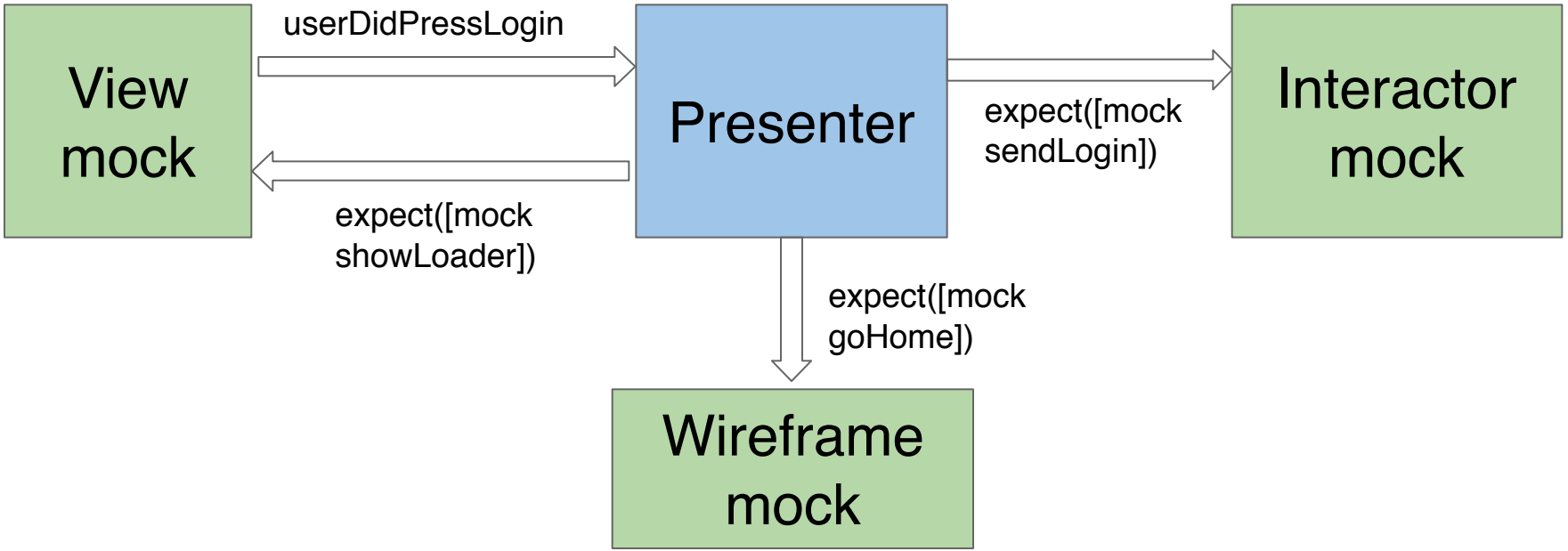
- Use some Viper Generator
- Write all the module protocols in one file
- Be self-critical
- Team members have to push together

Testing

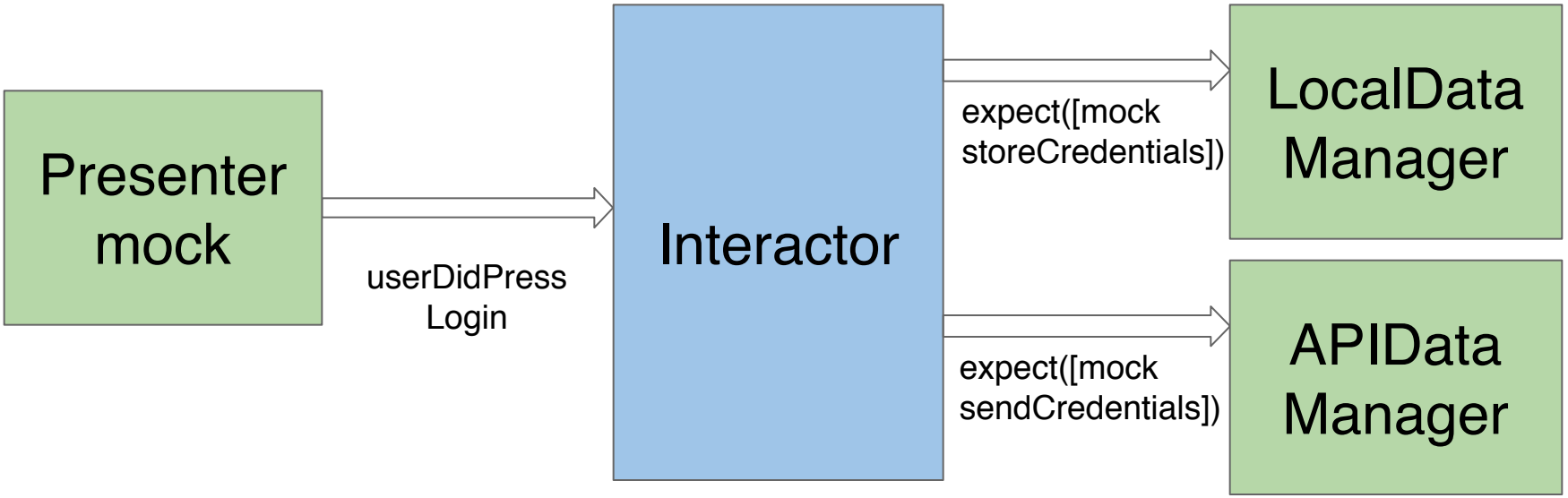
Testing the View



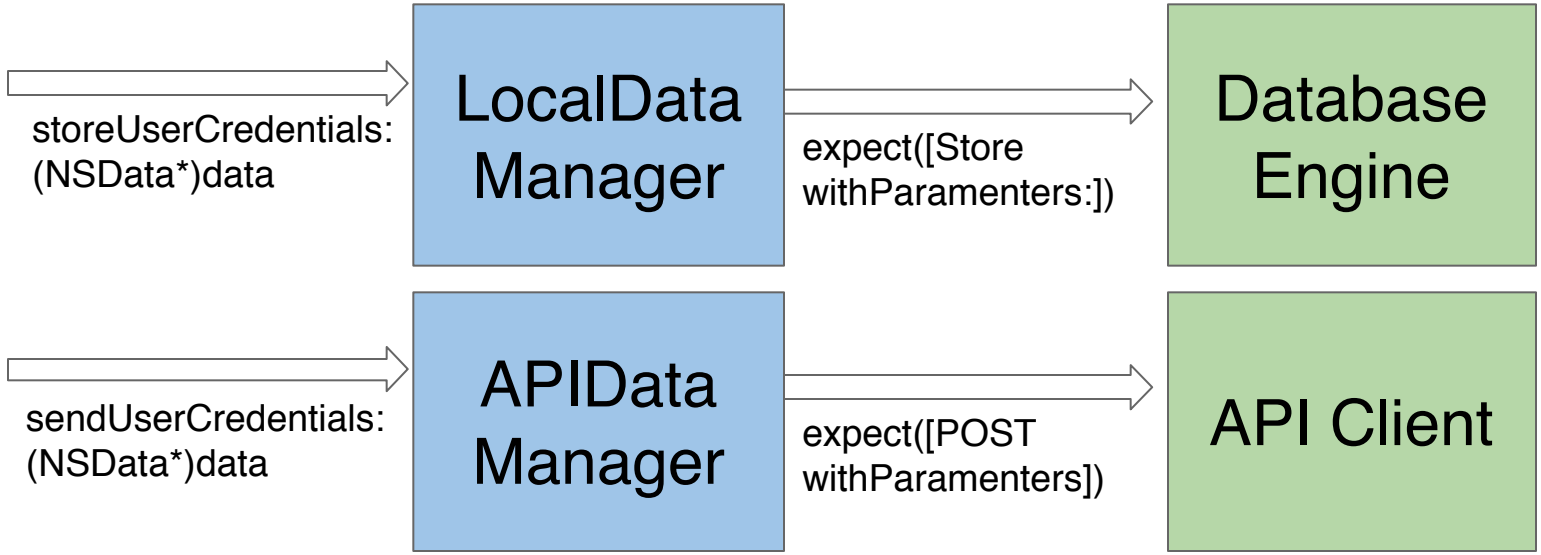
Testing the Presenter



Testing the Interactor



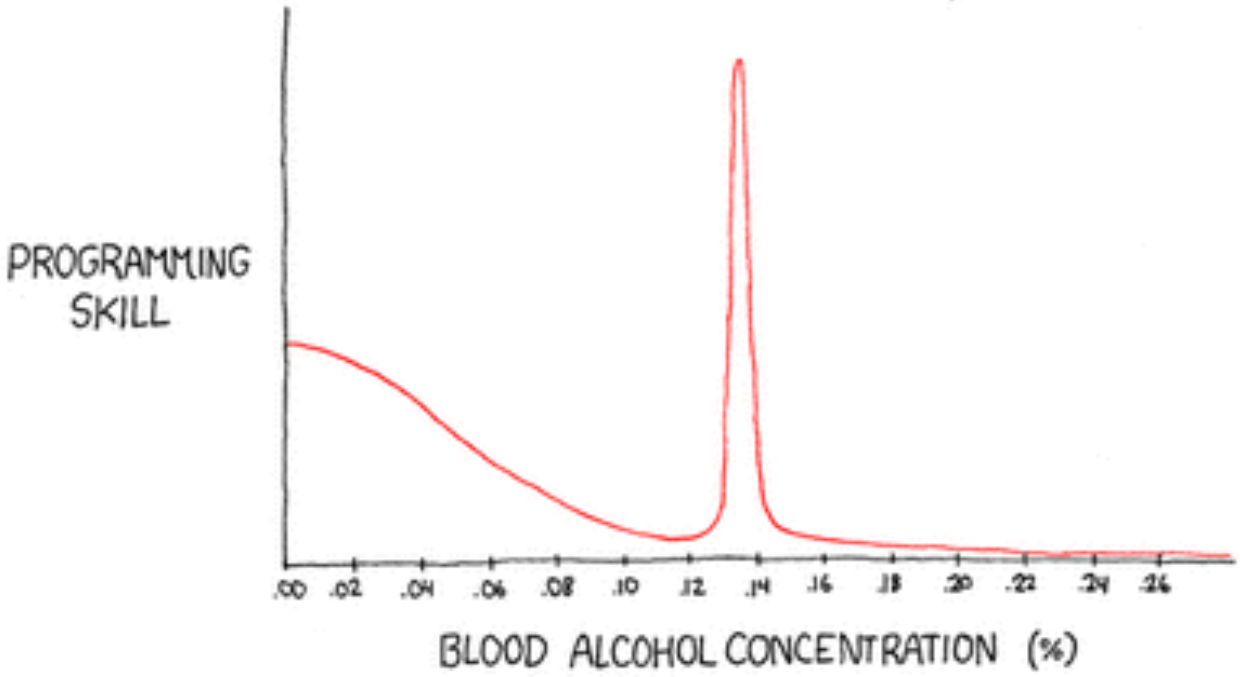
Testing the DataManager



One last tip

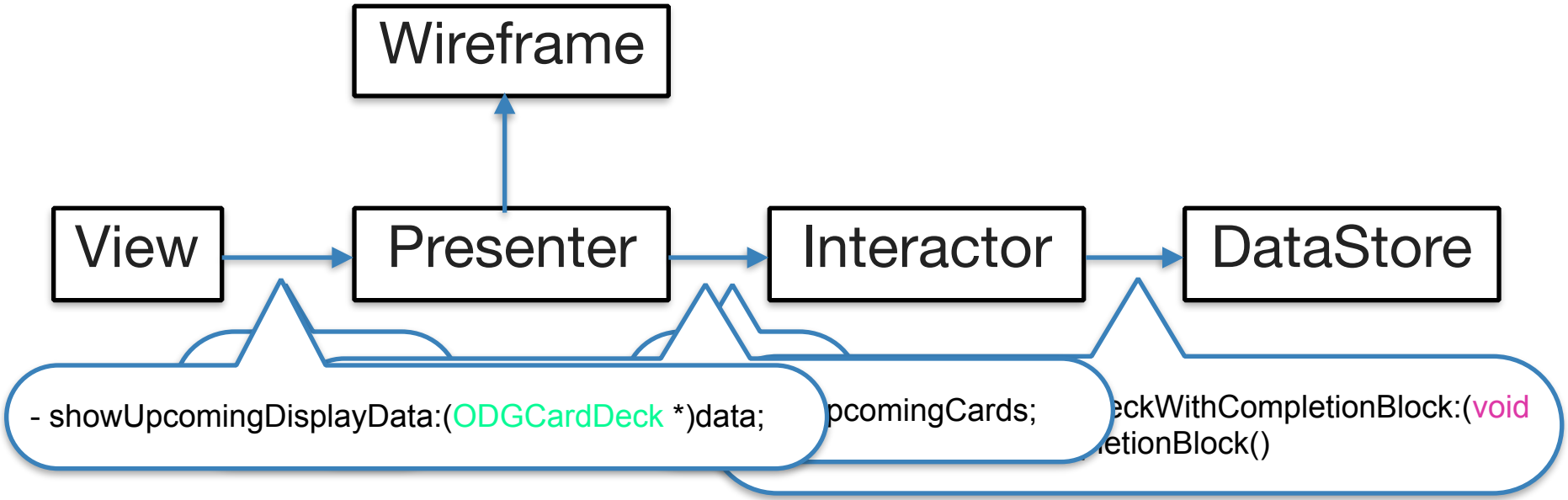
PATIENCE

Ballmer Peak

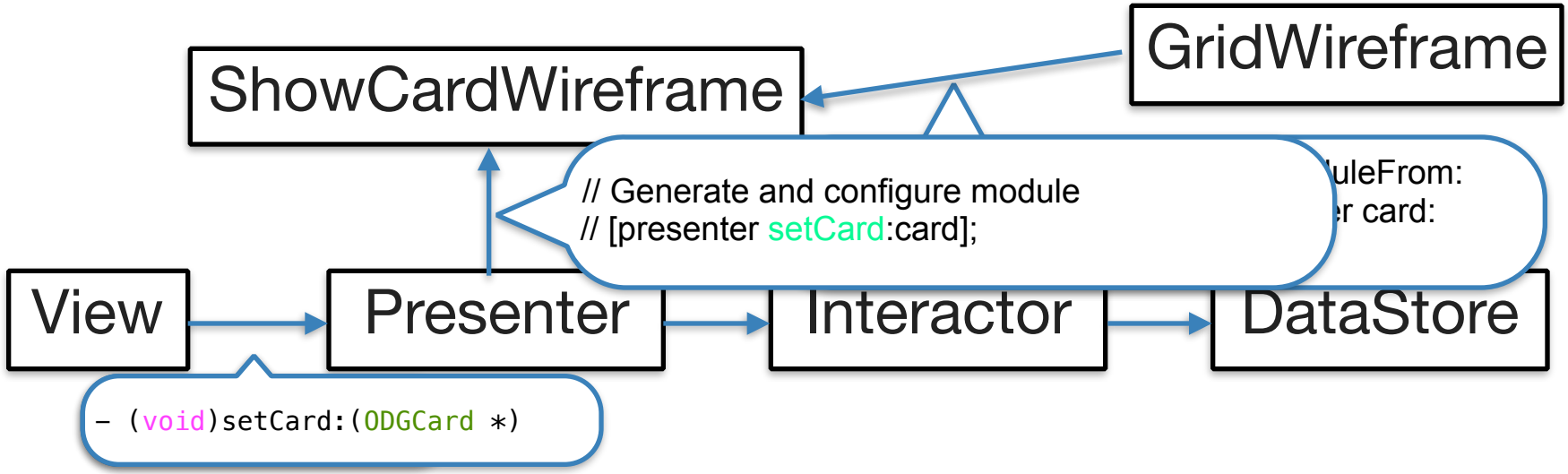


Demo

Grid - Initialization



ShowCard


GridWireframe
ShowCardWireframe
// Generate and configure module
// [presenter setCard:card];
View
Presenter
Interactor
DataStore
- (void)setCard:(ODGCard *)

Questions?

Examples

- ScrumViper
- Viper To-do
- Viper counter
- Viper module generator

References

- http://www.objc.io/issue-13/viper.html
- https://speakerdeck.com/pepibumur/viper-looking-for-a-perfect-architecture
- http://www.slideshare.net/kprofic/from-mvc-to-viper?qid=04fec0d2-0359-4e54-81b9-c2db9bf23516&v=qf1&b=&from_search=1
- http://www.objc.io/issue-1/lighter-view-controllers.html
- http://ppinera.es/2014/11/16/viper-looking-for-the-perfect-architecture.html
- http://mutualmobile.github.io/blog/2013/12/04/viper-introduction/
- https://github.com/pepibumur/viper-module-generator
- https://github.com/mutualmobile/Counter
- http://typhoonframework.org/

Thank you!