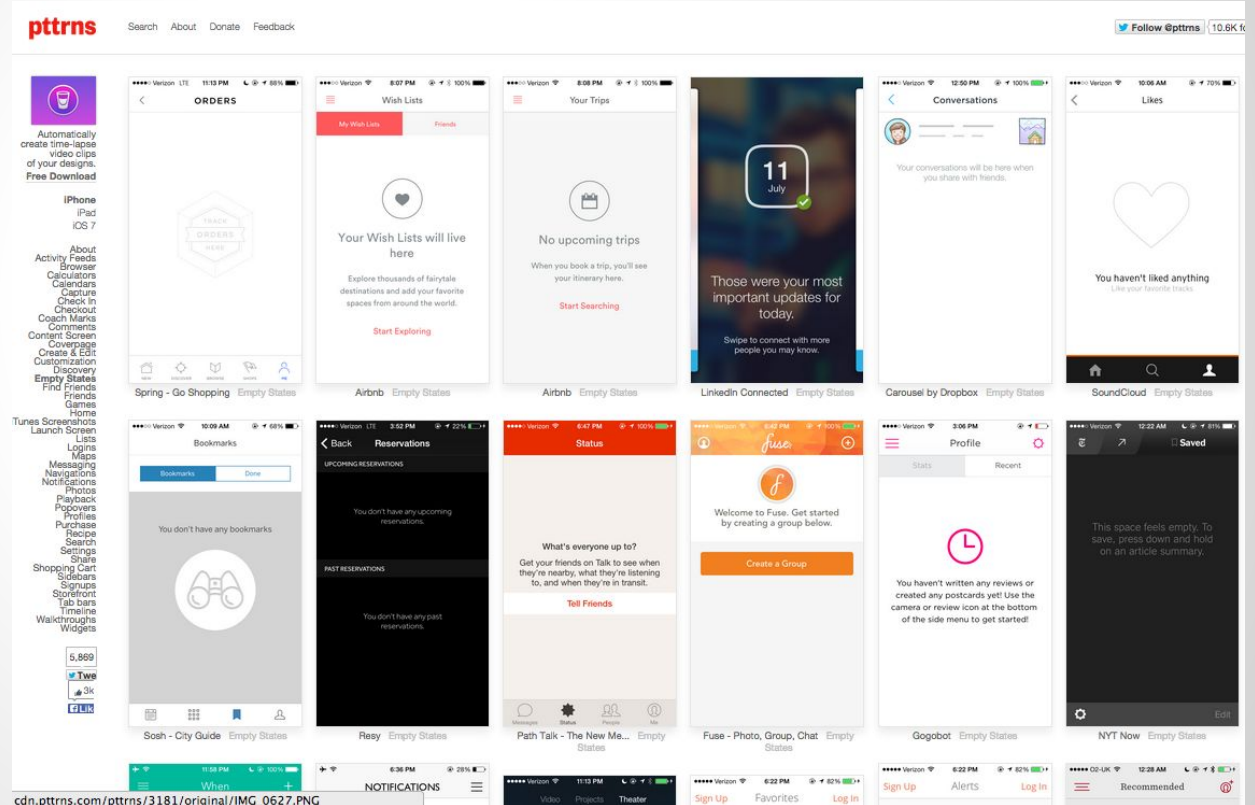# iOS Development

Architecting iOS Apps

Less than Useless Info

pttrns

Topics

Planning

Setting up Root View Controllers

Design Patterns

Modularization

It all starts with an idea

Ideation is hard.

Keep a notebook for your ideas

Who is the audience?

Is it free, freemium, paid, ad, etc?

Failing to plan is planning to fail

Scope out the intent of the app

Make a list of features:

    needs to haves

    nice to haves

    future stretch items

Fail early, fail often, so you don't fail

Design with sketches to rough out the idea

Wireframe with a tool

Proof of concept anything tricky

Use a design tool to mock out each screen

Prototype your idea

Budgeting - time and money

Can't manage what you don't understand

Can't change what you can't measure

List out each feature and mark it with a time estimate

3x those estimates

Hourly rate? ($ x 2,000 = yearly gross income)

Additional assets, software, services?

Commit early, commit often

Source control your projects

Enable continuous integration such as Fastlane

Use CocoaPods or Carthage

User Testing

Will reveal a ton of what you took for granted.

Will expose issues your users have.

Will uncover thoughts someone new has.

Iterate

Release early, release often.

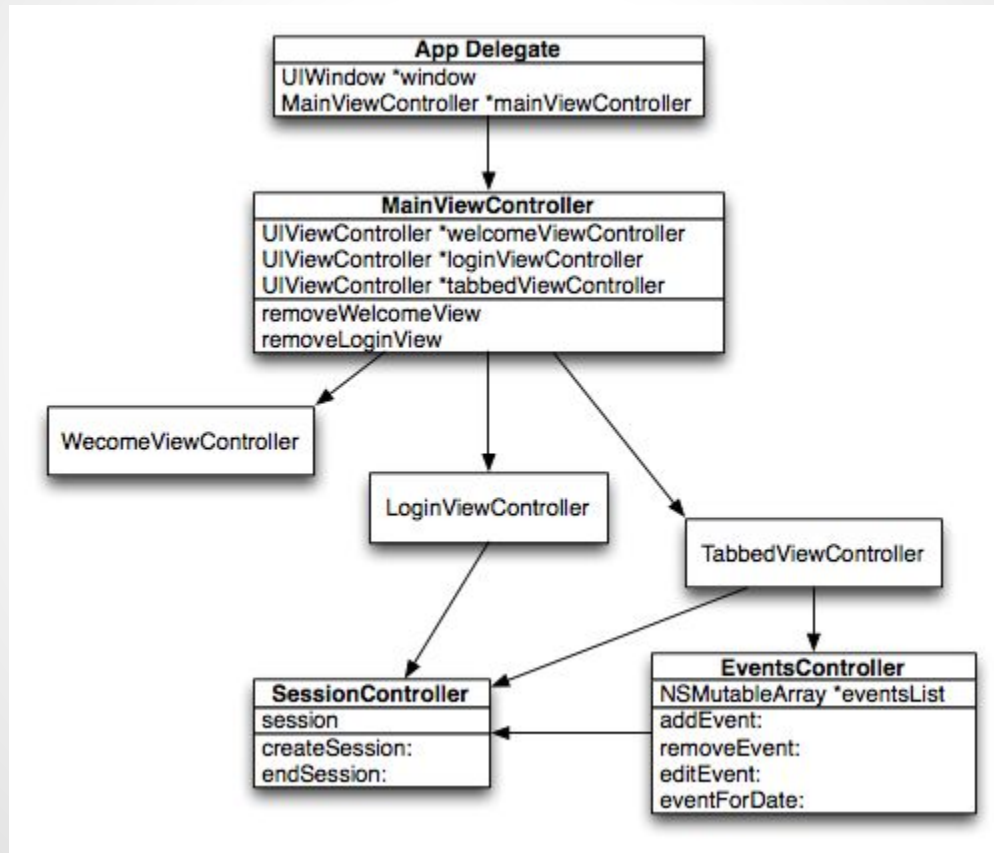Release, release, release.

Approval ~1.5 days

Enable crash reporting and analytics

Setting up a Main View Controller

A Main View Controller is launched from your App Delegate or Storyboard

It controls which subview is initially shown. Also controls if the main subview needs to be swapped out for another.

Class Diagrams

Design Patterns

Singletons - Only one instance will ever exist

Factories - Objects whose sole purpose is to produce other objects. Ex. (JSON -> Swift object, or `MazeFactory.createMaze()`)

Model-View-Controller - Architect Apple uses for interface development

Modularization

Wrap functionality into their own classes

Any kind functionality that needs to be accessed by several items is a candidate for a singleton

Architecting iOS Apps

Demo