iOS at Foursquare

Sam Grossberg and Mitch Livingston

–

January 2017

FOURSQUARE

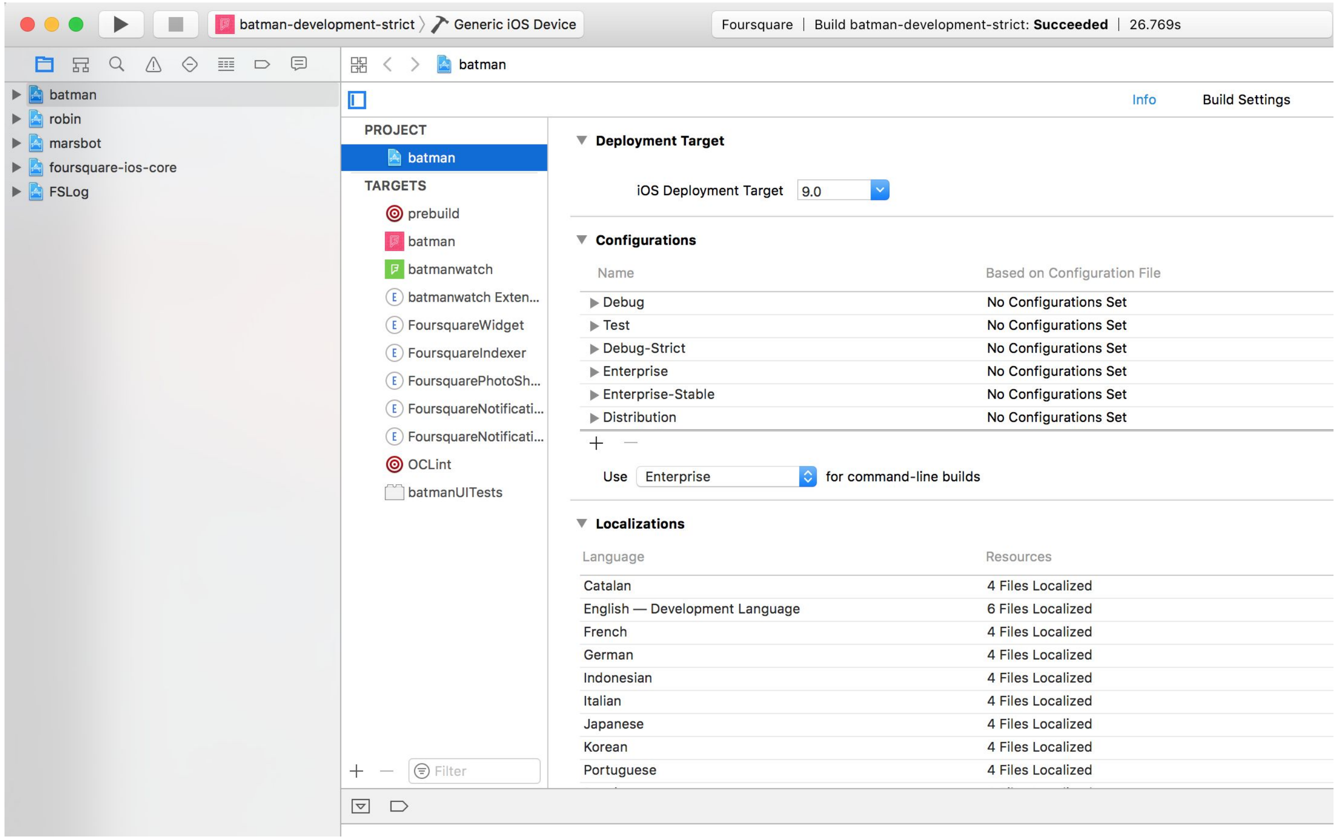
3 apps, 1 xcworkspace

FOURSQUARE
CITY GUIDE

FOURSQUARE
Swarm

FOURSQUARE
marsbot.

3 apps, 1 xcworkspace, ~100 provisioning profiles



batman-development-strict 〉 Generic iOS Device

Foursquare | Build batman-development-strict: **Succeeded** | 26.769s

- batman
- robin
- marsbot
- foursquare-ios-core
- FSLog

batman

PROJECT
- batman

TARGETS
- prebuild
- batman
- batmanwatch
- batmanwatch Exten...
- FoursquareWidget
- FoursquareIndexer
- FoursquarePhotoSh...
- FoursquareNotificati...
- FoursquareNotificati...
- OCLint
- batmanUITests

Info | Build Settings

Deployment Target

iOS Deployment Target: 9.0

Configurations

Name	Based on Configuration File
Debug	No Configurations Set
Test	No Configurations Set
Debug-Strict	No Configurations Set
Enterprise	No Configurations Set
Enterprise-Stable	No Configurations Set
Distribution	No Configurations Set

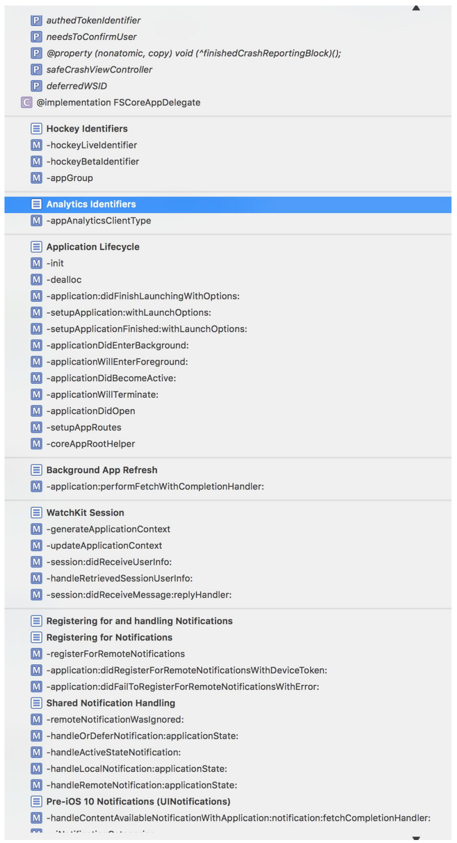
Use Enterprise for command-line builds

Localizations

Language	Resources
Catalan	4 Files Localized
English — Development Language	6 Files Localized
French	4 Files Localized
German	4 Files Localized
Indonesian	4 Files Localized
Italian	4 Files Localized
Japanese	4 Files Localized
Korean	4 Files Localized
Portuguese	4 Files Localized

FSCoreAppDelegate

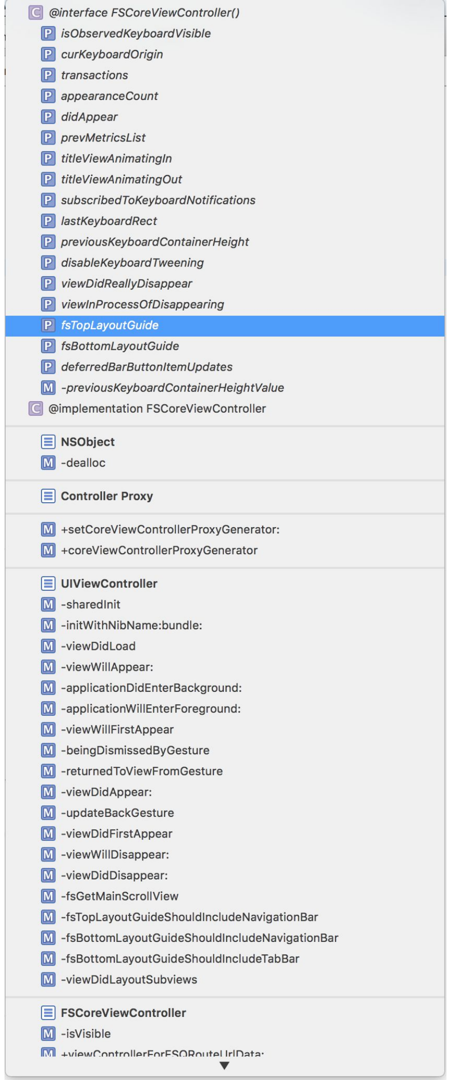
- Shared library initialization
  - AppsFlyer
  - Branch
  - Button
  - Facebook
  - Hockey
  - Moat
- User authentication
- URL routing
- Analytics and tracking
- Remote notification setup and handling



```
P authedTokenIdentifier
P needsToConfirmUser
P @property (nonatomic, copy) void (^finishedCrashReportingBlock)();
P safeCrashViewController
P deferredWSID
C @implementation FSCoreAppDelegate

Hockey Identifiers
M -hockeyLiveIdentifier
M -hockeyBetaIdentifier
M -appGroup

Analytics Identifiers
M -appAnalyticsClientType

Application Lifecycle
M -init
M -dealloc
M -application:didFinishLaunchingWithOptions:
M -setupApplication:withLaunchOptions:
M -setupApplicationFinished:withLaunchOptions:
M -applicationDidEnterBackground:
M -applicationWillEnterForeground:
M -applicationDidBecomeActive:
M -applicationWillTerminate:
M -applicationDidOpen
M -setupAppRoutes
M -coreAppRootHelper

Background App Refresh
M -application:performFetchWithCompletionHandler:

WatchKit Session
M -generateApplicationContext
M -updateApplicationContext
M -session:didReceiveUserInfo:
M -handleRetrievedSessionUserInfo:
M -session:didReceiveMessage:replyHandler:

Registering for and handling Notifications
Registering for Notifications
M -registerForRemoteNotifications
M -application:didRegisterForRemoteNotificationsWithDeviceToken:
M -application:didFailToRegisterForRemoteNotificationsWithError:
Shared Notification Handling
M -remoteNotificationWasIgnored:
M -handleOrDeferNotification:applicationState:
M -handleActiveStateNotification:
M -handleLocalNotification:applicationState:
M -handleRemoteNotification:applicationState:
Pre-iOS 10 Notifications (UINotifications)
M -handleContentAvailableNotificationWithApplication:notification:fetchCompletionHandler:
```

FSCoreViewController

- Keyboards are hard
- Custom view lifecycle notifications and delegate methods (viewWillFirstAppear, etc)
- URL routing support
- FSCoreScrollViewController, etc.



```
C @interface FSCoreViewController()
  P isObservedKeyboardVisible
  P curKeyboardOrigin
  P transactions
  P appearanceCount
  P didAppear
  P prevMetricsList
  P titleViewAnimatingIn
  P titleViewAnimatingOut
  P subscribedToKeyboardNotifications
  P lastKeyboardRect
  P previousKeyboardContainerHeight
  P disableKeyboardTweening
  P viewDidReallyDisappear
  P viewInProcessOfDisappearing
  P fsTopLayoutGuide
  P fsBottomLayoutGuide
  P deferredBarButtonItemUpdates
  M -previousKeyboardContainerHeightValue
C @implementation FSCoreViewController

  NSObject
  M -dealloc

  Controller Proxy

  M +setCoreViewControllerProxyGenerator:
  M +coreViewControllerProxyGenerator

  UIViewController
  M -sharedInit
  M -initWithNibName:bundle:
  M -viewDidLoad
  M -viewWillAppear:
  M -applicationDidEnterBackground:
  M -applicationWillEnterForeground:
  M -viewWillFirstAppear
  M -beingDismissedByGesture
  M -returnedToViewFromGesture
  M -viewDidAppear:
  M -updateBackGesture
  M -viewDidFirstAppear
  M -viewWillDisappear:
  M -viewDidDisappear:
  M -fsGetMainScrollView
  M -fsTopLayoutGuideShouldIncludeNavigationBar
  M -fsBottomLayoutGuideShouldIncludeNavigationBar
  M -fsBottomLayoutGuideShouldIncludeTabBar
  M -viewDidLayoutSubviews

  FSCoreViewController
  M -isVisible
  M +viewControllerForFSQRouteUrlData:
```

FSQCellManifest

January 31, 2017

FOURSQUARE

FSQCellManifest

A UITableView and UICollectionView delegate and datasource that provides a simpler unified interface for describing your sections and cells.

- Describe your cell structure in one place in code.
- Extensive configuration options and convenience properties to support almost any kind of table or collection view you may have.
- Provides a unified interface for table and collection views.

We use FSQCellManifest as a base for almost every view controller we make

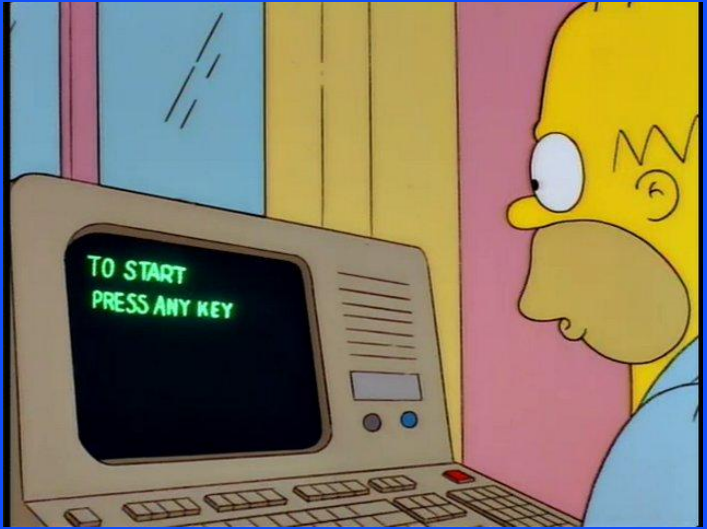
Simplification FTW!

- Moves height and cell-reuse/configuration code out of view controllers / data sources into the cell classes themselves, making cells easier to reuse in different screens.

- Allows you to define cell behaviors using blocks instead of delegate callbacks.

- Removes need to pre-register cell identifiers.

- Avoid series of huge switch statements and hard-to-maintain cell placement logic.



Example

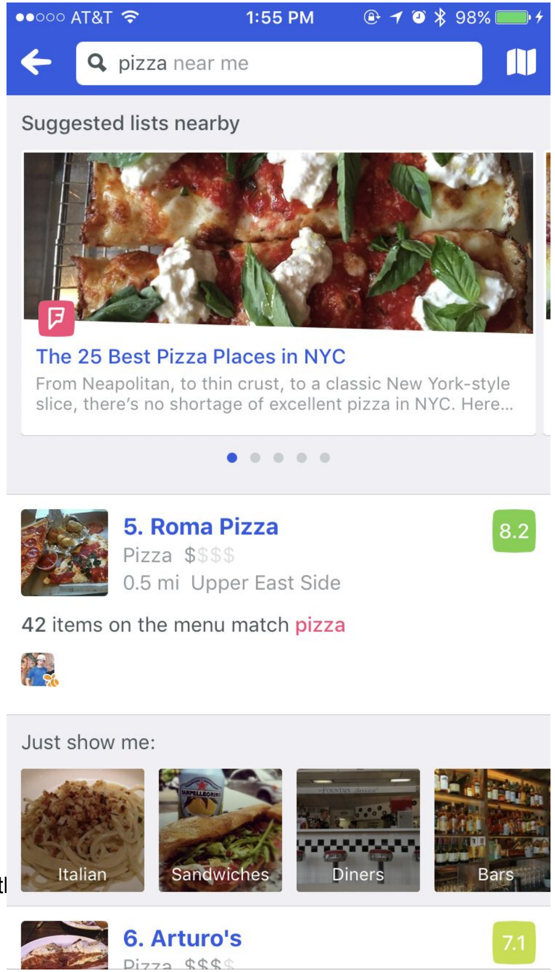
FSQCellManifest

Example

- Yup, these are in Objective-C.
- Yup, this works in Swift.

Tableviews the standard way

```
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    NSUInteger rowCount = self.venuesToDisplay.count;
    if (self.listPivots.count > 0) {
        ++rowCount;
    }
    rowCount += self.pivots.count;
    return rowCount;
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath {
    if (indexPath.row == 3 && self.listPivots.count > 0) {
        ListPivotsCell *cell = [tableView dequeueReusableCellWithIdentifier:@"listPivotsIdent" forIndexPath:indexPath];
        // config
        return cell;
    }

    NSUInteger pivotOffset = self.listPivots.count > 0 ? 5 : 4;
    if (indexPath.row >= pivotOffset && indexpath.row < (pivotOffset + self.pivots.count) {
        Pivot *pivot = self.pivots[indexPath.row - pivotOffset];
        UpsellPivotCell *cell = [tableView dequeueReusableCellWithIdentifier:@"upsellPivotsIdent" forIndexPath:indexPatl
        // config
        return cell;
    }
```


●●○○○ AT&T
1:55 PM
98%
pizza near me
Suggested lists nearby
The 25 Best Pizza Places in NYC
From Neapolitan, to thin crust, to a classic New York-style slice, there's no shortage of excellent pizza in NYC. Here...
5. Roma Pizza
8.2
Pizza $$$$
0.5 mi Upper East Side
42 items on the menu match pizza
Just show me:
Italian
Sandwiches
Diners
Bars
6. Arturo's
7.1
Pizza $$$$

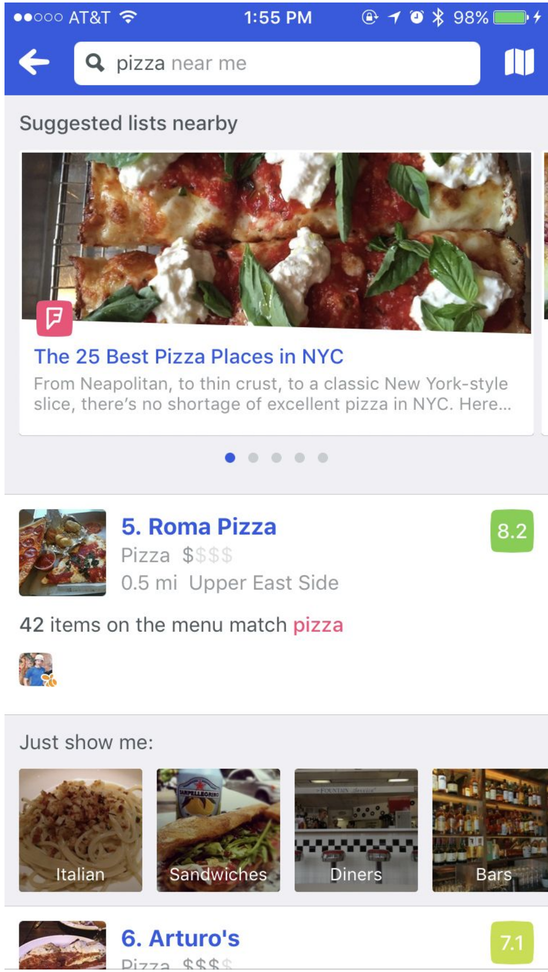
There is a lot more that didn’t fit on that slide!

Just for this quick example:

```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section;
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;
- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView;

- (void)tableView:(UITableView *)tableView willDisplayCell:(UITableViewCell *)cell forRowAtIndexPath:(NSIndexPath *)indexPath;
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath;
```

Tableviews using FSQCellManifest

```
- (void)reloadData {
  NSArray<FSQCellRecord *> *venueRecords = [FSQCellRecord withModels:self.venuesToDisplay
                                                          cellClass:[VenueCell class]
                                                        configBlock:nil
                                                        selectBlock:^(TableViewController *controller, VenueModel *cellModel, NSIndexPath *indexPath) {
                                                            [controller pushVenue:cellModel];
                                                        }];
  NSArray<FSQCellRecord *> *upsellPivots = [FSQCellRecord withModels:self.pivots
                                                           cellClass:[UpsellPivotCell class]
                                                         configBlock:nil
                                                         selectBlock:^(TableViewController *controller, UpsellPivot *cellModel, NSIndexPath *indexPath) {
                                                             [controller pivotTapped:cellModel];
                                                         }];
  venueRecords = [venueRecords arrayByInsertingObjects:upsellPivots startingAtIndex:4];
  if (self.listPivots.count > 0) {
    FSQCellRecord *listPivot = [FSQCellRecord withModel:self.listPivots
                                              cellClass:[ListPivot class]
                                            configBlock:^(TableViewController *controller, id cell, id cellModel, NSIndexPath *indexPath) {
                                                // set up tapping for each list
                                            }
                                            selectBlock:nil];
    venueRecords = [venueRecords arrayByInsertingObject:listPivot atIndex:3];
  }

  self.manifest.sectionRecords = [FSQSectionRecord withCellRecords:venueRecords];
}
```



●●○○○ AT&T 1:55 PM 98%
pizza near me
Suggested lists nearby
The 25 Best Pizza Places in NYC
From Neapolitan, to thin crust, to a classic New York-style slice, there's no shortage of excellent pizza in NYC. Here...
5. Roma Pizza
8.2
Pizza $$$$
0.5 mi Upper East Side
42 items on the menu match pizza
Just show me:
Italian
Sandwiches
Diners
Bars
6. Arturo's
7.1
Pizza $$$$

How is height (and size for collection views) of each cell calculated?

This logic lives in each cell subclass.

- Conform to FSQCellManifestTableViewCellProtocol
- The manifest will call the required height/size method, passing in the model and table/collection view size.
- The manifest is in charge of calling this.
- Let’s the cell own the size calculation.
    - Makes cell reuse and view controller refactoring a little easier.

Plugins

- Easily add on extra functionality in separate compartmentalized classes.
- A plugin is like an extra delegate object that can receive all of the manifest's many delegate callbacks in addition to the actual delegate object.

- Plugins can be set when the manifest is initialized, or added/removed later.
  - In this way, features that are only needed by certain screens can be added when needed.
- Plugins also gives you the ability easily write generic code that will work on both table views or collection views

FSQCellManifest

- https://github.com/foursquare/FSQCellManifest
- CocoaPods
- Carthage