1 Mobile Dashboards Overview

This application carries PBS&J dashboard web parts across the firewall and displays them in the small screen space of an Apple-manufactured mobile device such as an iPhone or iPad.

Heretofore, Applications Development has provided textual Windows-compatible forms to Blackberries and iPhones for the purpose of allowing approvals from various e*net applications and Remedy tickets.

This application expands our field communications ability and competitive advantage by:

- Displaying graphics on the mobile device such as graphic plots, tabular statistics and pie charts
- Leverages iPhoneôs ability to navigate among web parts
- Dynamically responds to usersô organization and period filter selections
- Queries the database residing on SQL Server behind the Firewall

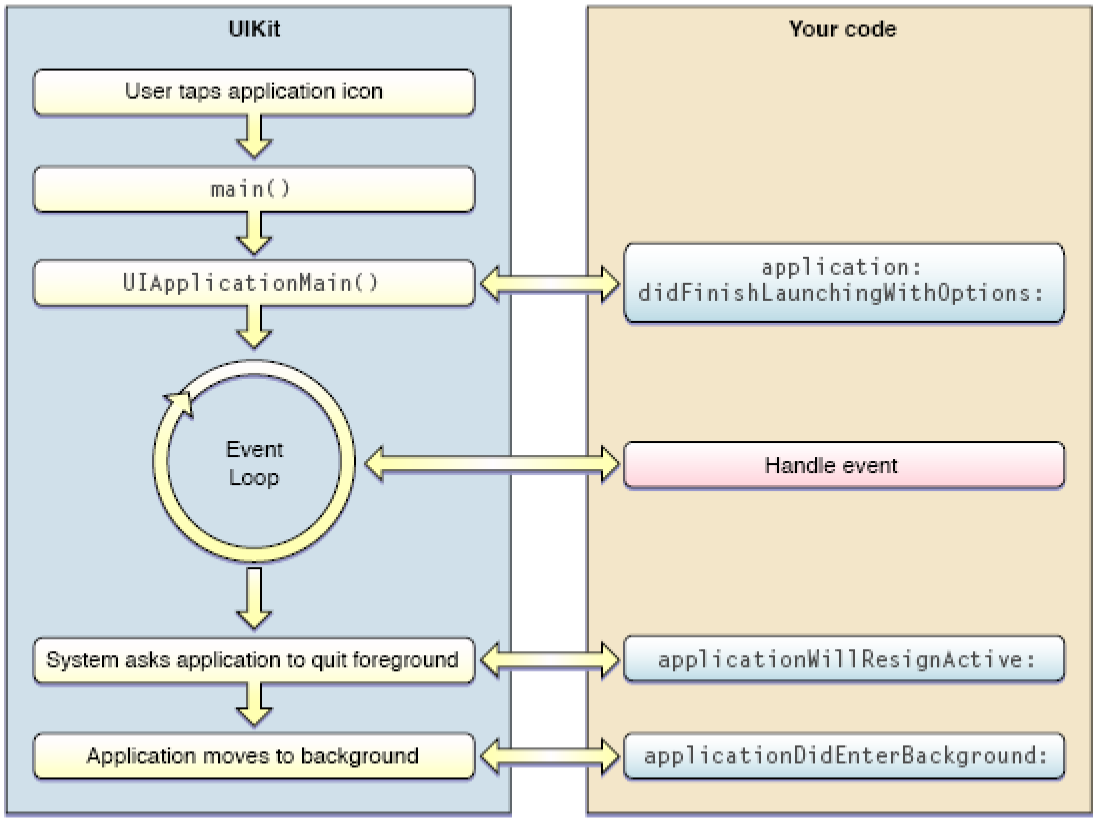
2 Mac Xcode Components for iPhone / iPad Applications

2.1 Functional Overview and Terms

2.1.1 UIKit framework

The UIKit System object manages the overall process and your custom code implements your applicationõs specific behavior.

Figure 1: Application Life Cycle


UIKit
Your code
User taps application icon
main()
UIApplicationMain()
application:
didFinishLaunchingWithOptions:
Event
Loop
Handle event
System asks application to quit foreground
applicationWillResignActive:
Application moves to background
applicationDidEnterBackground:

Source: iOS Application Programming Guide

An iOS application receives events continuously

- Receiving the events is the job of the UIApplication object,

  The `UIApplication` object manages the application event loop. Use this object as is.

- Responding to the events is the responsibility of your custom Application Delegate object.

  You provide the application delegate object to hold custom code for modifying the behavior of framework objects. The `UIApplication` object notifies this object when specific application-level events occur, such as when the user selects (taps) a dashboard or web part.

2.1.2 File Types

Every object, or class, has three files to define itð h, m, and xib.

2.1.2.1 .h and .m

Use the class name for Interface and implementation files. The interface file usually has the `.h` extension typical of header files. The implementation file has the `.m` extension, indicating that it contains Objective-C source code. For example, the Rectangle class would be declared in `Rectangle.h` and defined in `Rectangle.m`.

Interface declaration (.h file)

```
#import <UIKit/UIKit.h>
#import "MySuperclass.h"
@interface ClassName : MySuperclass
{
    instance variable declarations
}
@properties /* getters and setters */
method declarations
     -method /* minus sign signifies instance method*/
     + method /* Plus sign signifies class method*/
@end
```

Implementation (.m file)

```
#import "ClassName.h"
@implementation ClassName : MySuperclass
{
    instance variables
}
method definitions
@end
```

Every implementation file must import its own interface. For example, `Rectangle.m` imports `Rectangle.h`.

2.1.2.2 nib | xib file

Appleȇs Interface Builder tool provides a graphical palette of buttons, text boxes, input boxes and other controls that you drag into a virtual iPhone screen to paint your view. The visual screen you paint is stored in a syntax that the iPhone interprets and renders. The Interface Builder output file extension changed to xib (Xcode Interface Builder) but developers commonly still refer to them as nib files.

2.2 Web services

Web Services middleware bridges the iPhone view outside the firewall with data residing on servers within the firewall. DWH Web Services modules do the following:

- Interface with AD to authenticate the user
- Select data requested by the iPhone from the DWH
- Format plots of raw data

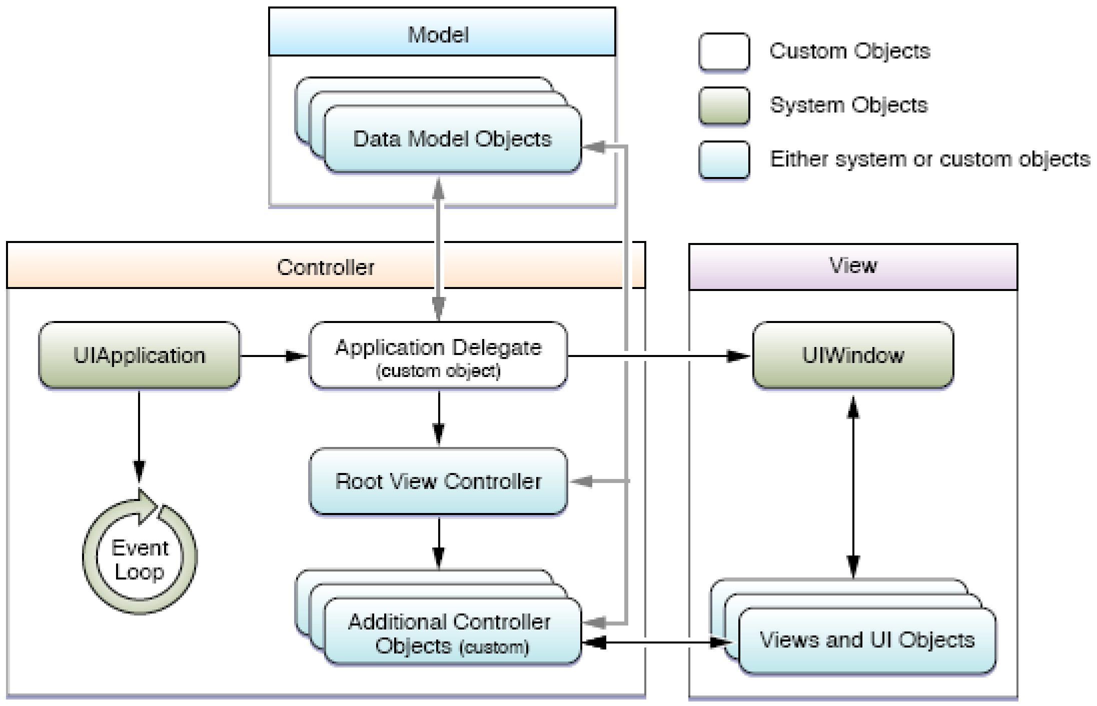
You supply an AppConfig.plist file to specify the network server housing the DWH, as well as the context and protocol needed to reach it via the Internet..

2.3 Model-View-Controller (MVC) development

The MVC methodology divides your code into three independent functional areas.

- **model** objects define your data.
- **view** portion defines the user interface
- **controller** object acts as a bridge between the model and view and coordinates the passing of data between them.

Figure 2: MVC Components



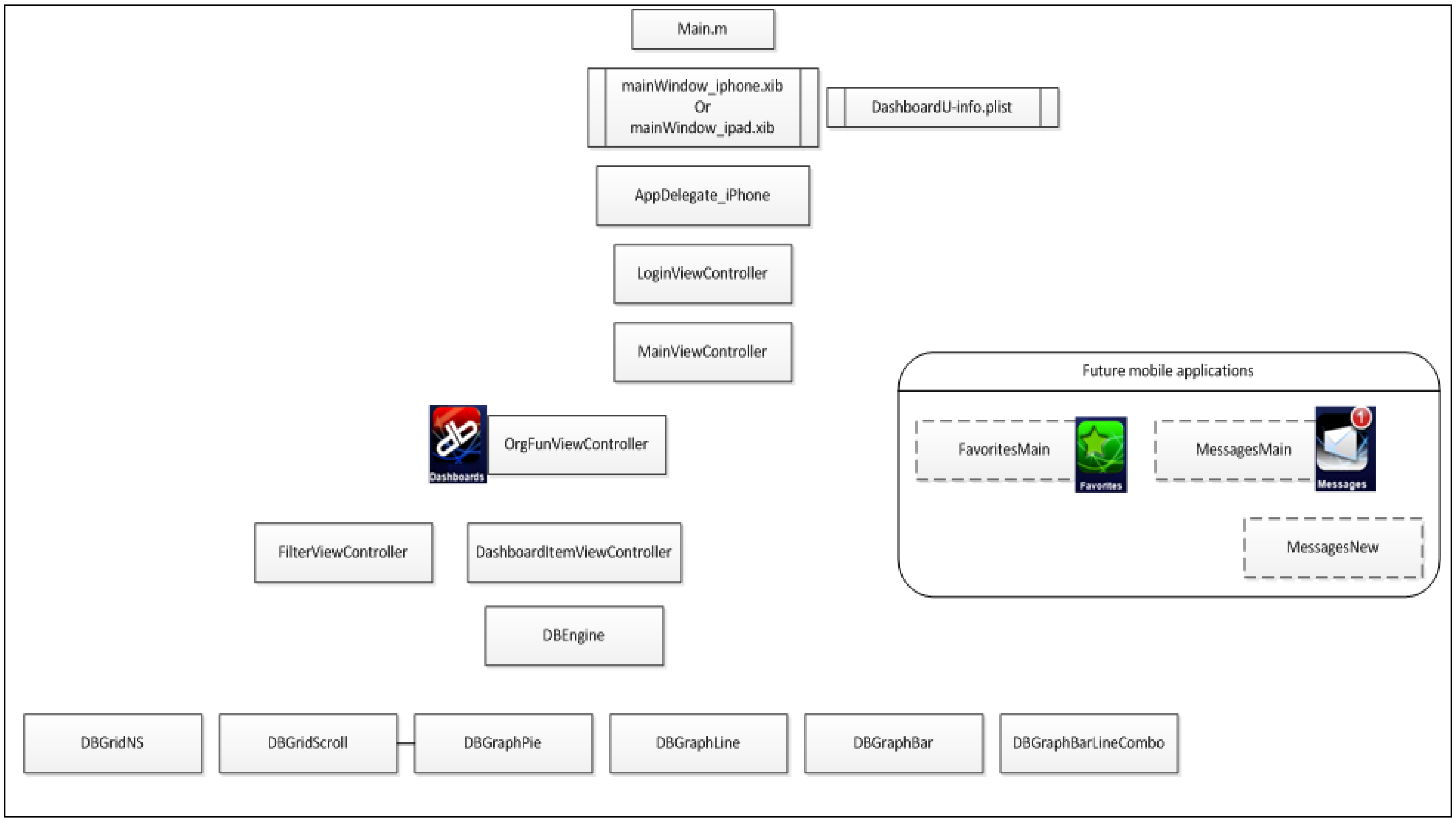
Source: iOS Application Programming Guide

2.4 Objects

Data Model objects	Data Model objects describe your applicationȇs content. Examples • Sequence of drawing commands that plots a graph. The actual rendering of that graph takes place elsewhere in your application. • Column headings, row headings and cell values for a table. The iPhone-resident application calls Web services to instantiate data object values from the DWH database. The DWH formats data for return it as JSON, an xml-like syntax. For details, see
View objects	A view object draws content in a designated rectangular area and responds to events within that area. The UIKit framework provides standard views for presenting many different types of content. Define custom views by subclassing `UIView` (or its descendants) directly.
View Controller Objects	View controller objects create the views described in the xib files and manage interactions between the views and your data model objects. The base class for all view controller objects--`UIViewController`--provides default functionality for animating the appearance of views, handling device rotations, and other standard system behaviors. Other UIKit view controller classes manage standard system interfaces, such as navigation interfaces or the image picker. Controls, a specialized type of view, implement interface objects such as buttons, text fields, and toggle switches.
`UIWindow` object	A `UIWindow` object hosts views on the device screen. and delivers events to those views and to their managing view controllers.

3 Applications Development Components for iPhone Mobile Dashboard

Figure 3: Objects in PBSJ's Mobile Dashboard Application



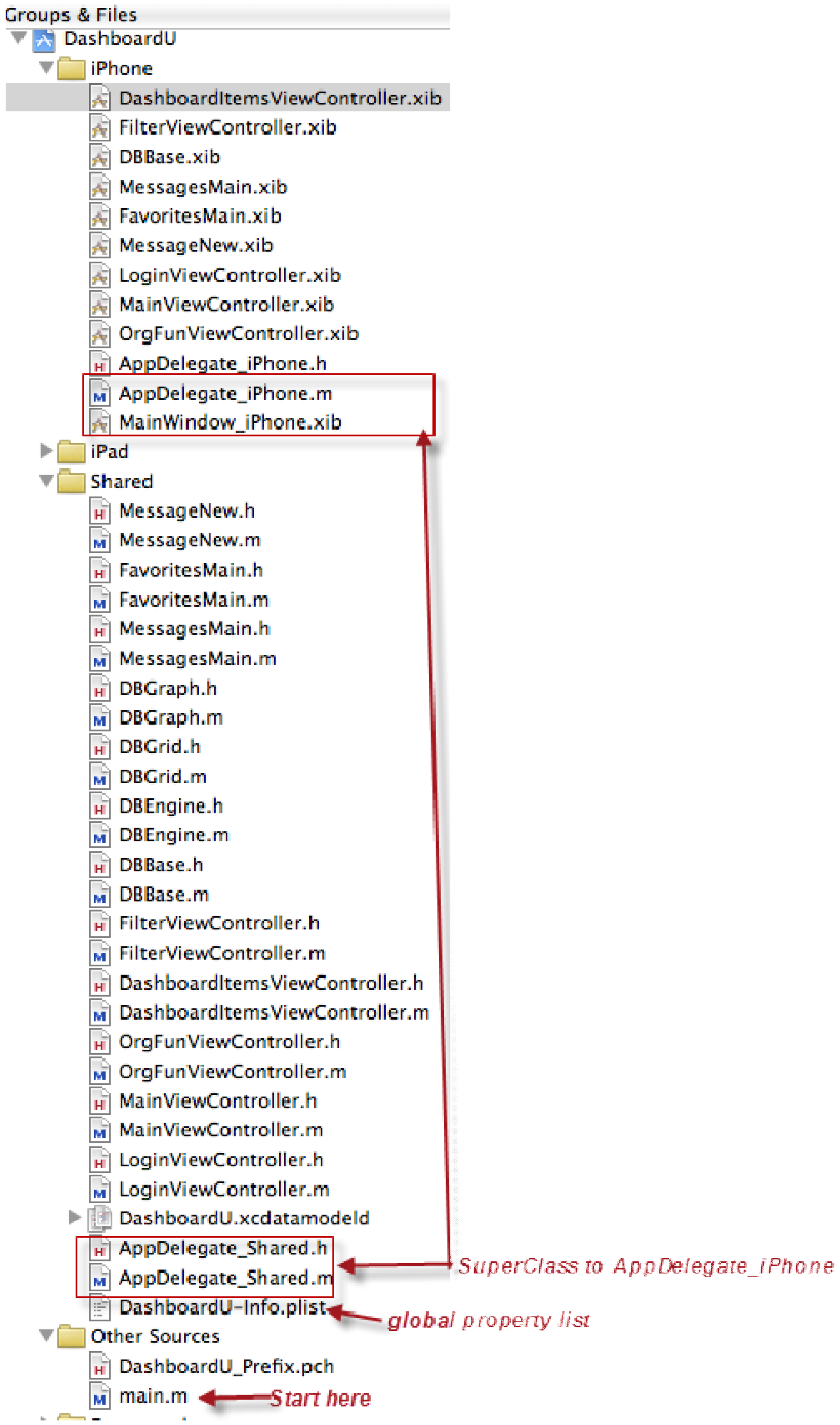
3.1 Source Code Structure

As shown in Figure 4 on the next page, the source code directory tree divides this application as follows:

- iPad- holds
  - the AppDelegate that adjusts dashboards to fill an iPad window
  - all iPad view xib files.
- iPhone - holds
  - the AppDelegate that adjusts dashboards to fill an iPhone window
  - all iPhone view xib files..
- Shared ï holds
  - the AppDelegate superclass that the iPad and iPhone AppDelegate objects import
  - all object and plist files common to both iPhone and iPad
- Other Sourcesï holds main

Figure 4 shows the application directory tree inside Xcode Explorer on the Mac. When copied to Team Foundation Server / Virtual Studio on a PC, the hierarchy changes.

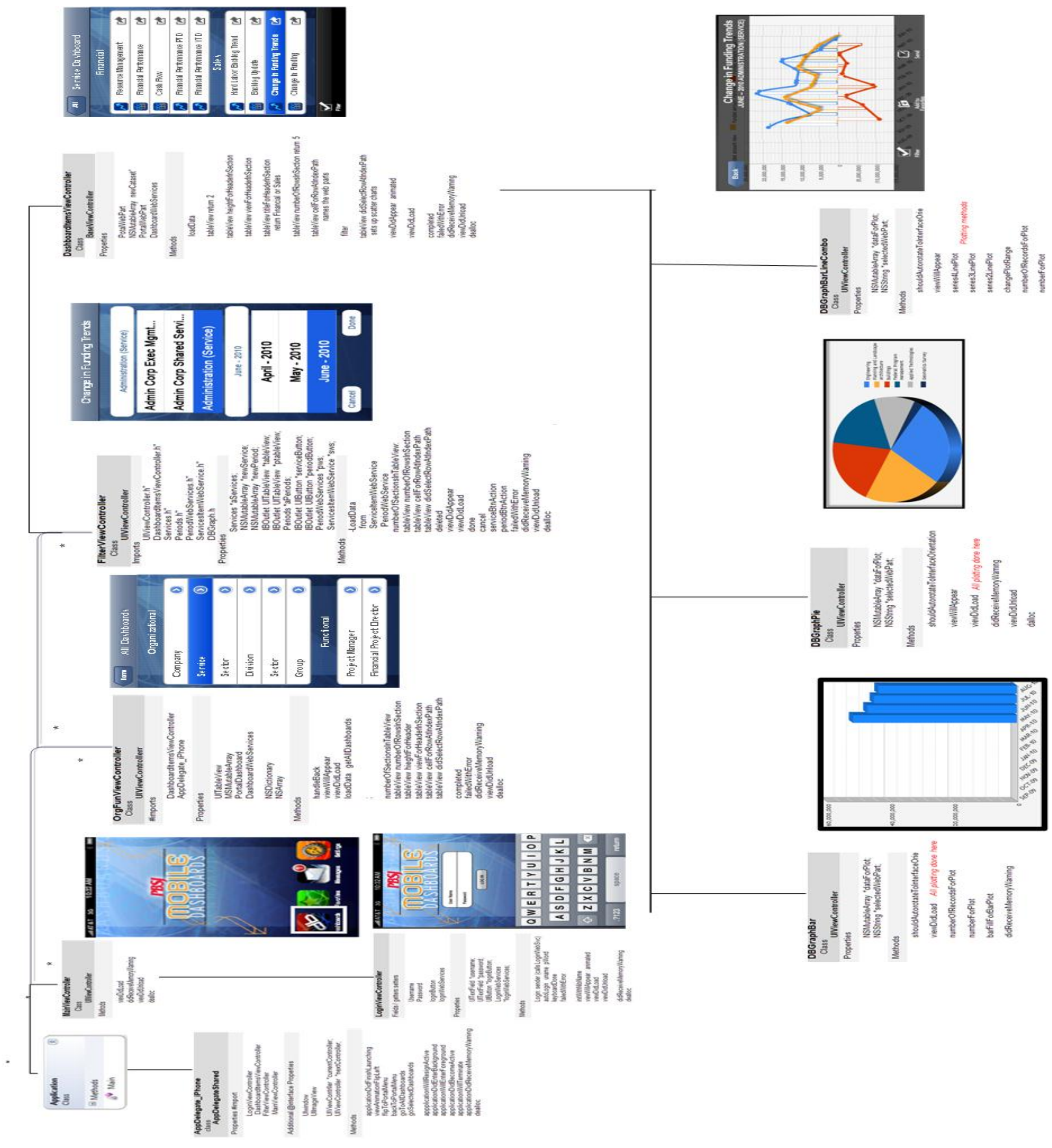
Figure 4: Location of objects in the source code tree.



```
Groups & Files
▼ DashboardU
  ▼ iPhone
      DashboardItemsViewController.xib
      FilterViewController.xib
      DBBase.xib
      MessagesMain.xib
      FavoritesMain.xib
      MessageNew.xib
      LoginViewController.xib
      MainViewController.xib
      OrgFunViewController.xib
      AppDelegate_iPhone.h
      AppDelegate_iPhone.m
      MainWindow_iPhone.xib
  ▶ iPad
  ▼ Shared
      MessageNew.h
      MessageNew.m
      FavoritesMain.h
      FavoritesMain.m
      MessagesMain.h
      MessagesMain.m
      DBGraph.h
      DBGraph.m
      DBGrid.h
      DBGrid.m
      DBEngine.h
      DBEngine.m
      DBBase.h
      DBBase.m
      FilterViewController.h
      FilterViewController.m
      DashboardItemsViewController.h
      DashboardItemsViewController.m
      OrgFunViewController.h
      OrgFunViewController.m
      MainViewController.h
      MainViewController.m
      LoginViewController.h
      LoginViewController.m
    ▶ DashboardU.xcdatamodeld
      AppDelegate_Shared.h
      AppDelegate_Shared.m
      DashboardU-Info.plist
  ▼ Other Sources
      DashboardU_Prefix.pch
      main.m
```

3.2 Application Flow

After the user taps the Mobile Dashboards icon ( ), the system displays some transitional graphics and proceeds to launch your application by calling the default `main` function, which passes control to the UIKit framework..


Application
AppDelegate_iPhone
AppDelegateShared
MainViewController
LoginViewController
OrgFunViewController
FilterViewController
DashboardItemsViewController
DBGraphBarLineCombo
DBGraphPie
DBGraphBar
Plotting methods
All plotting done here
Change in Funding Trends
All Dashboards
Service Dashboard