Advanced app development for iOS

Lab 4

Working with files, UITableView and maps.

Gdańsk 2015
Tomasz Idzi

Introduction

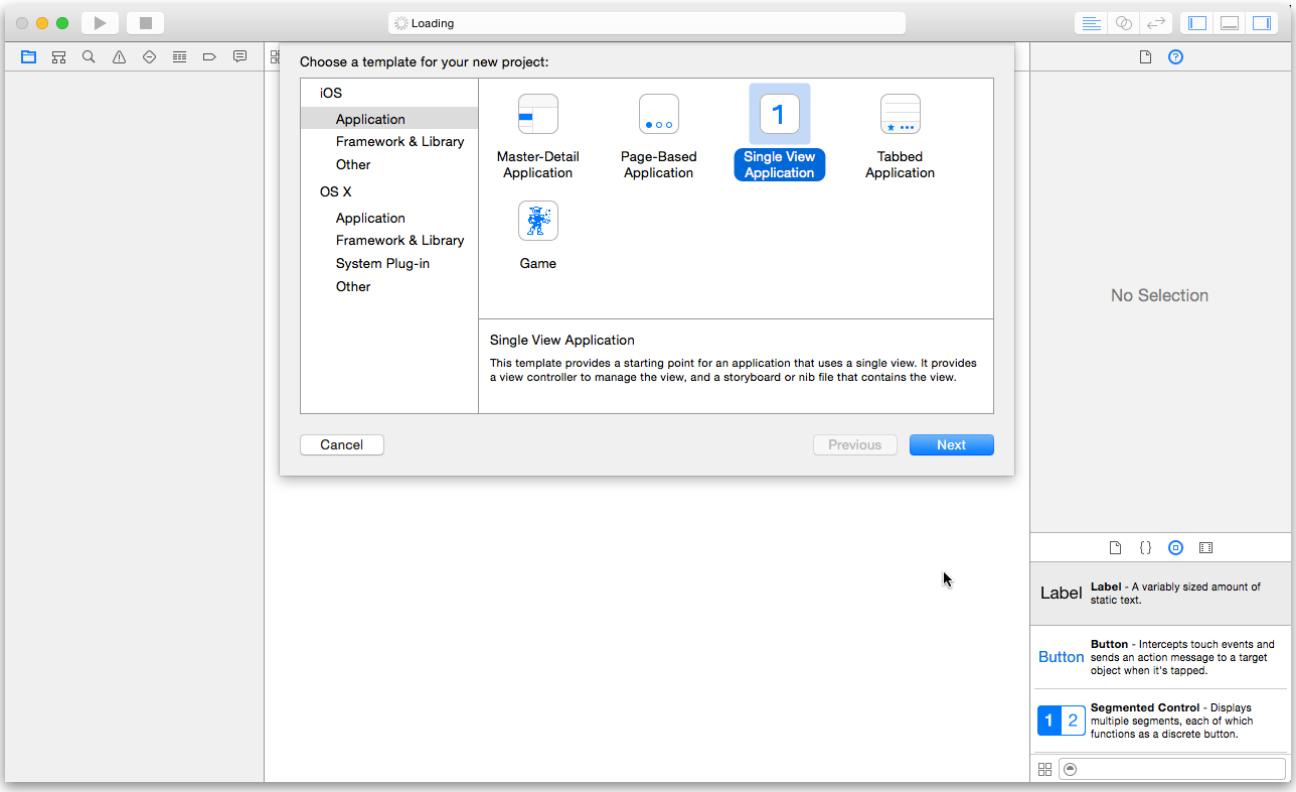
In this lab you will create application which have a few new controls such as TabBar, UITableView, MapView. Thanks to this you will be more familiar with Cocoa Touch, creating custom User Interface in Interface Builder, and using Objective-C objects such as NSArray or NSDictionary.

For this lab, you can get **5 points**:

1. Create User Interface
2. Working with files
3. Create custom UITableViewCell
4. Create Detail View
5. Working with maps

Create User Interface (1 point)

1. Run Xcode and select **Create a new Xcode project**
2. From iOS section choose Application and **Single View Application** as a type of template.

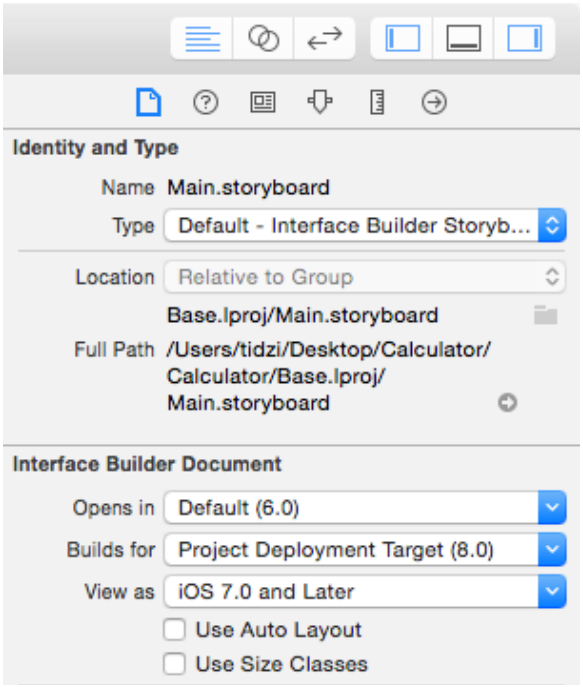


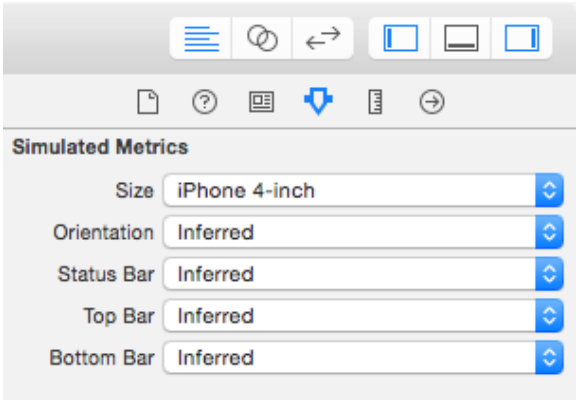
3. Choose following options for your project:
   - **Product Name**: Restaurants.
   - **Organization Name**: You can leave this blank.
   - **Organization Identifier**: com.yourName
   - **Language**: Objective-C
   - **Devices**: iPhone

- **Use Core Data**: Leave unselected

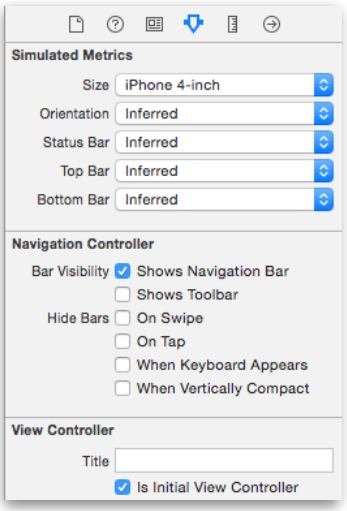
4. Choose place on disk to save a project.
5. From Navigation area which is on the left side of Xcode select file Main.storyboard.

6. In Interface Builder select file View Controller and move to Utilities area, which is on the right side of the Xcode. If you can't see it, probably it's hidden. To open any of hidden part of Xcode window use this buttons: 
7. Choose **File Inspector** section and unselect **Use Auto Layout**.
8. Choose **Attributes Inspector** and select **iPhone 4-inch** in **Size** option.

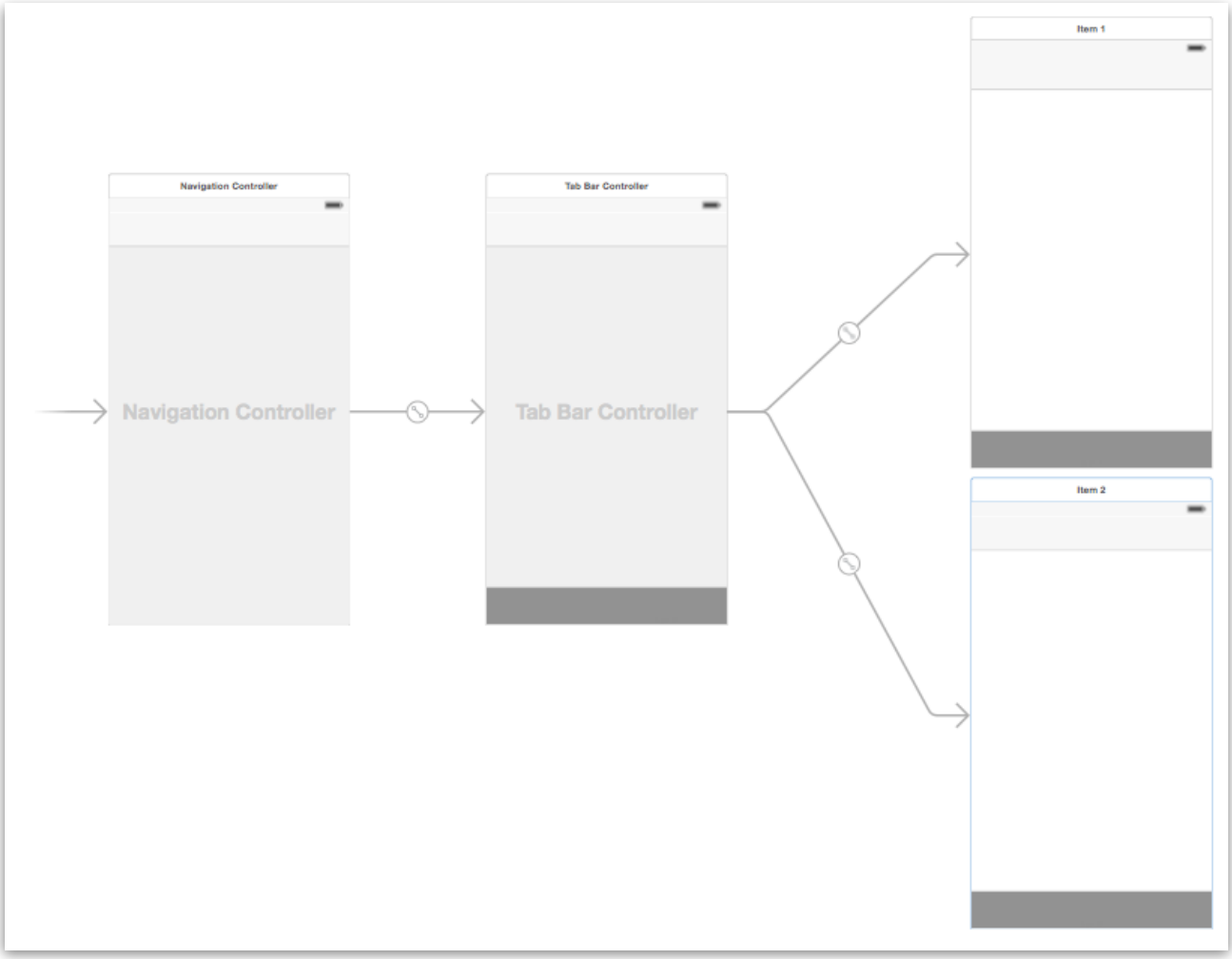




9. Open Main.Storyboard and remove everything what you have there.
10. Add **Navigation Controller** and **Tab Bar Controller** and make connection between them, select **Relationship Segue -> root view controller**.
11. Select **Navigation Controller** object and in **Utilities area** on the right side of Xcode window select **Attributes inspector.** Find **Is Initial View Controller** and select it.

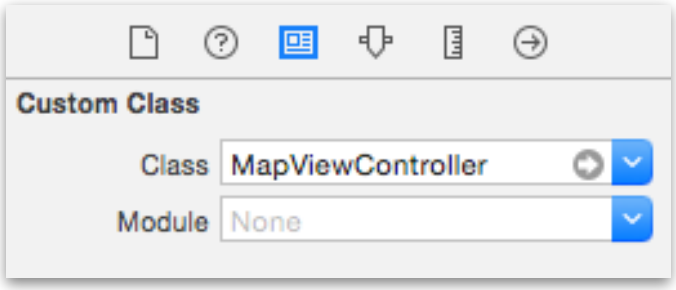


12. Your Storyboard should looks like:

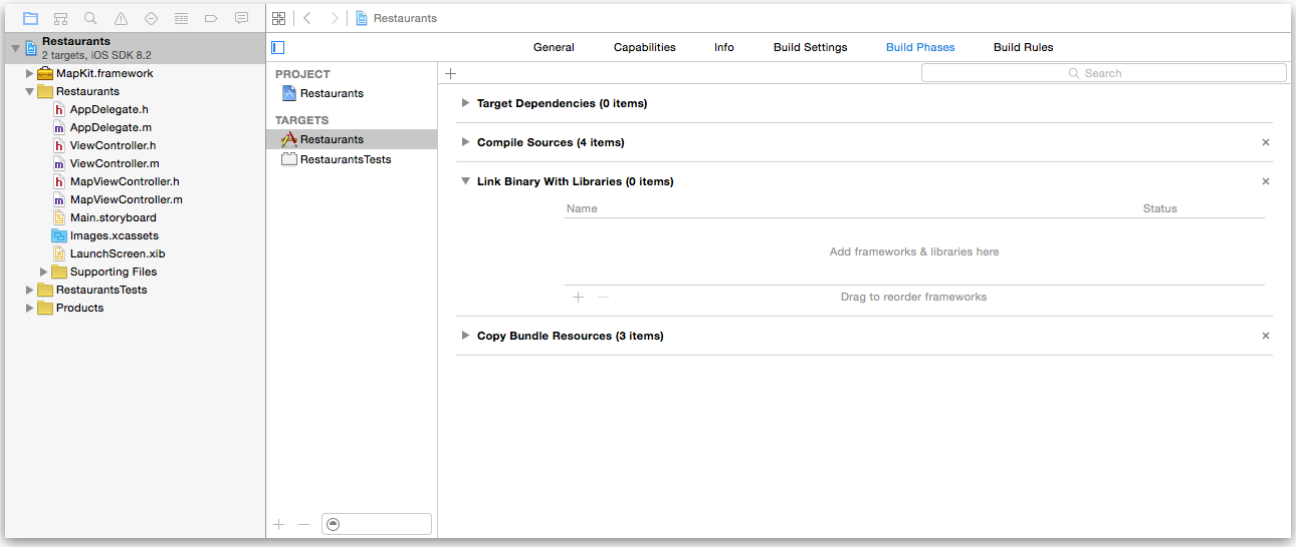


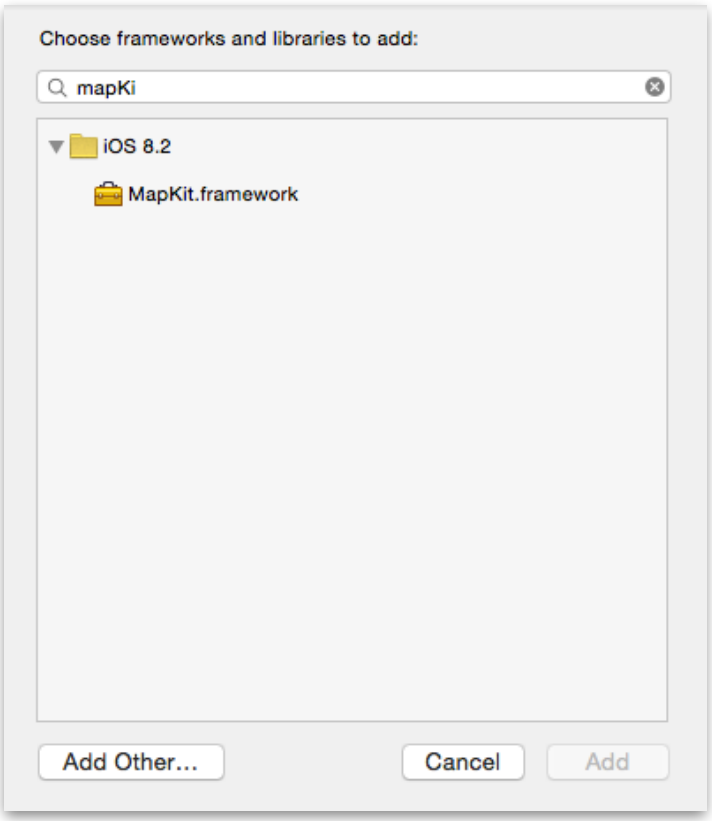
13. Run iOS Simulator and everythink is ok, you should see:

14. To the first Item of TabBar (ViewController) add UITableView, to the second MapView. In both cases set is to full screen.
15. Create new class and name it as **MapViewController** (subclass of **UIViewController**).
16. In Storyboard select View Controller where you added MapView. In Utilites area select Identity Inspector and set class as MapViewController.



17. Now you can make a connection to create a property of **MapView** in **MapViewController** class.
18. You have to add Apple's framework MapKit to start working with Maps. To do it, select project name on the left side of Xcode window in Navigator area. Next select your project in Targets and select **Build Phases**. Find **Link Binary With Libraries** open it and using + button add MapKit.framework





19. In the header file where you add a new property of MapView you have to import Framework, which is responsible for working with maps. To do it add:

```
#import <MapKit/MapKit.h>
```

20. The same do it in the View Controller object where you add UITableView. In this case you don't need to create a new class, due to the fact that Xcode created ViewController class during create a new project. Just set it in Interface Builder and make a connection to create a property of UITablewView in ViewController class.
21. Run iOS Simulator to check if everything works good.

Working with files (1 point)

1. Add resource files to the project, which contain restaurant's picture and Restaurants.plist file. In .plist file you can find information about seven restaurants. All the data comes from Forsquare API.
2. Create new class, name it **DataManager** and set it as subclass of NSObject. Add factory method, which return .plist structure as Objective-C objects:

```
+ (NSArray*)dataFromPlist
{
    NSString *fileName = @"Restaurants";
    NSArray *paths =
NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
NSUserDomainMask, YES);
    NSString *documentsDirectory = [paths objectAtIndex:0];
    NSString *path = [documentsDirectory
stringByAppendingPathComponent:[NSString
stringWithFormat:@"%@.plist",fileName]];

    if (![[NSFileManager defaultManager] fileExistsAtPath:path])
    {

        path = [[NSBundle mainBundle] pathForResource:fileName
ofType:@"plist"];
    }

    NSArray *data = [[NSArray alloc] initWithContentsOfFile:path];

    return data;
}
```

3. Now you have to set up ViewController class to display data in UITableView
   - In private interface set UITableView's delegates and create NSArray object to data source:

```
@interface ViewController () <UITableViewDelegate,
UITableViewDataSource>
{
    NSArray *_dataSource;
}
@end
```

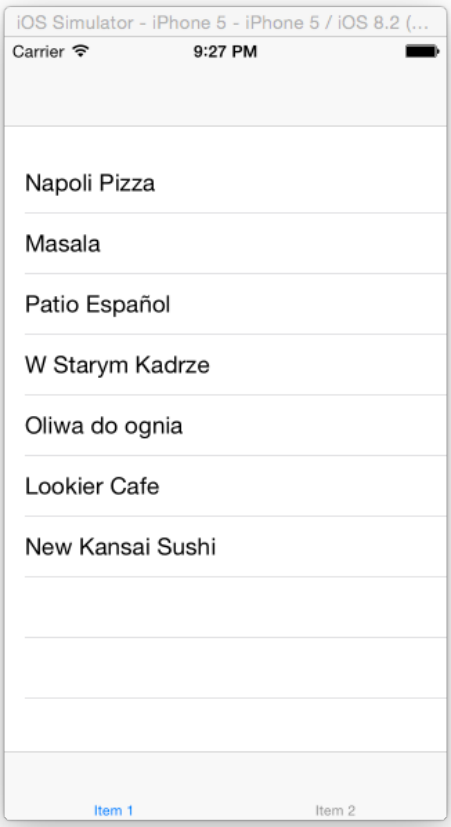
   - In **viewDidLoad** methods set delgates and get data from file.

```
- (void)viewDidLoad {
    [super viewDidLoad];

    self.tableView.delegate = self;
    self.tableView.dataSource = self;

    _dataSource = [DataManager dataFromPlist];
}
```

   - Implement required methods from UITableViewDelegate and UITableViewDataSource.

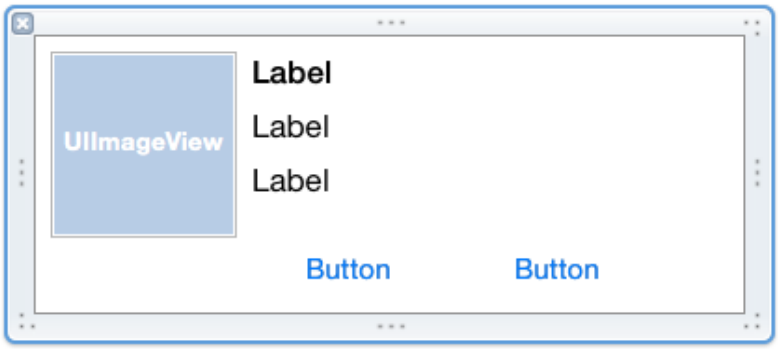
```
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:
(NSInteger)section {

    return _dataSource.count;
}

- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *MyIdentifier = @"MyIdentifier";

    UITableViewCell *cell = [tableView
dequeueReusableCellWithIdentifier:MyIdentifier];

    if (cell == nil)
    {
        cell = [[UITableViewCell alloc]
initWithStyle:UITableViewCellStyleDefault
                                      reuseIdentifier:MyIdentifier];
    }

    NSDictionary *dict = _dataSource[indexPath.row];
    cell.textLabel.text = dict[@"name"];
    return cell;
}
```

- Run iOS Simulator. You should see list of restaurant's name from .plist.



Create custom UITableViewCell (1 point)

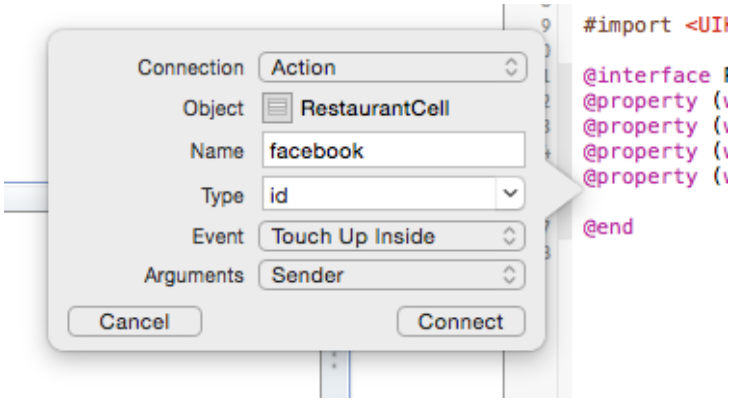
1. Create new class, which is subclass of UITableViewCell and name it RestaurantCell.
2. Depends on Xcode version select .xib file during creation or add separately. Create your own UITableViewCell in InterfaceBuilder which should looks like:



3. Create property for UIImageView na three UILabel and two methods for buttons. Your public interface should looks like:

```
@interface RestaurantCell : UITableViewCell
@property (weak, nonatomic) IBOutlet UIImageView *picture;
@property (weak, nonatomic) IBOutlet UILabel *name;
@property (weak, nonatomic) IBOutlet UILabel *street;
@property (weak, nonatomic) IBOutlet UILabel *city;

- (IBAction)facebook:(id)sender;
- (IBAction)website:(id)sender;
@end
```

🔑 To create method for UIButton, change connection type from Outlet to Action:



3. In ViewController import new class using:

```
#import "RestaurantCell.h"
```

4. Register nib file to UITableView, in viewDidLoad method:

```
[self.tableView registerNib:[UINib
nibWithNibName:NSStringFromClass([RestaurantCell class]) bundle:
[NSBundle mainBundle]] forCellReuseIdentifier:@"RestaurantCell"];
```

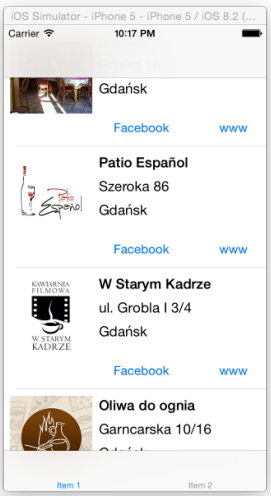
5. Change cellForRowAtIndexPath implementation to start working with custom cell.

```
- (UITableViewCell *)tableView:(UITableView *)tableView
cellForRowAtIndexPath:(NSIndexPath *)indexPath
{
    static NSString *MyIdentifier = @"RestaurantCell";

    RestaurantCell *cell = [tableView
dequeueReusableCellWithIdentifier:MyIdentifier];

    if (cell == nil)
    {
        cell = [RestaurantCell new];
    }

    NSDictionary *dict = _dataSource[indexPath.row];
    cell.name.text = dict[@"name"];
    cell.street.text = dict[@"address"];
    cell.city.text = dict[@"city"];
    cell.picture.image = [UIImage imageNamed:dict[@"photo"]];
    return cell;
}
```

6. Check in Interface Builder height of your cell and return this value using method:

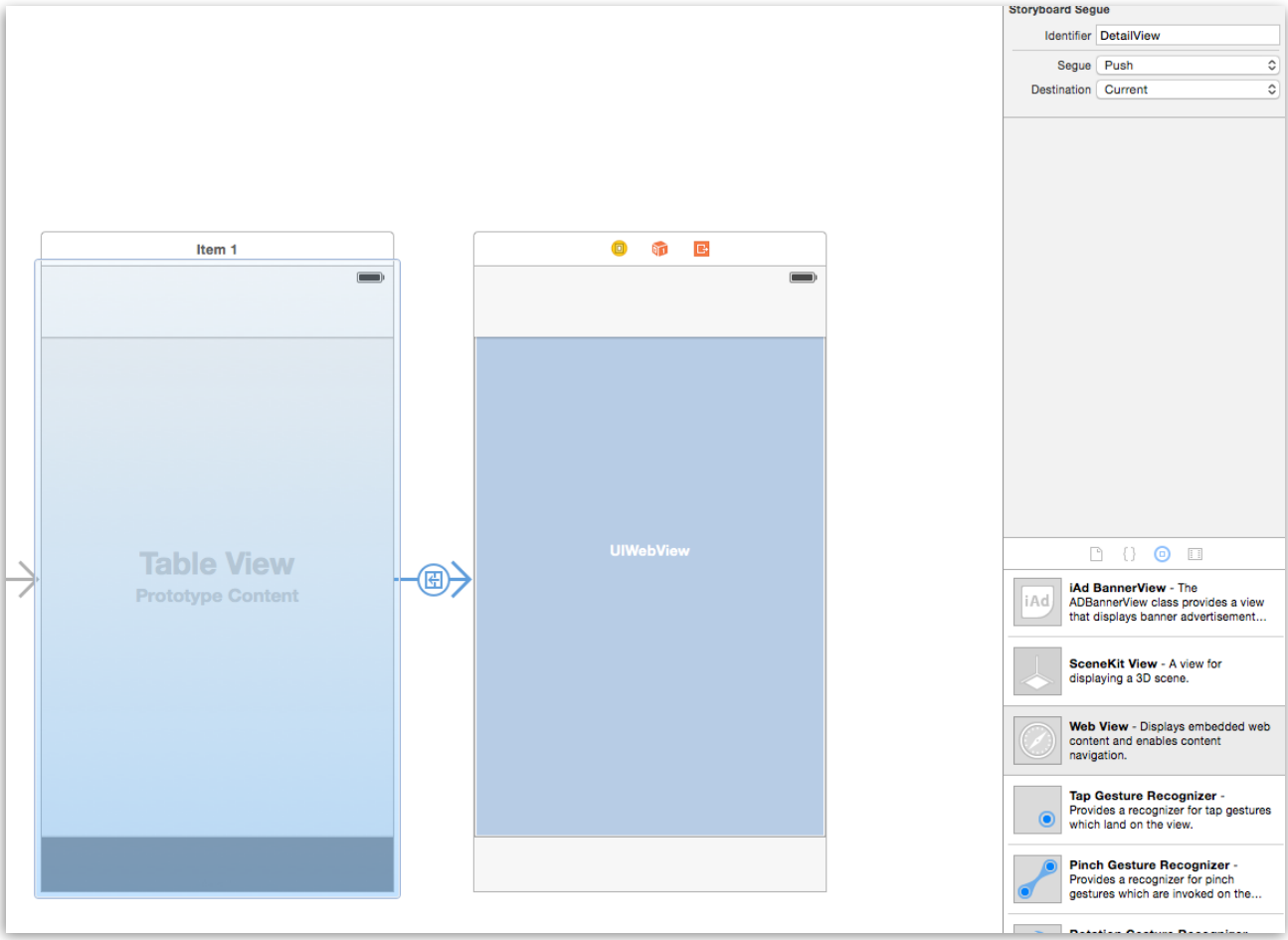
```
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:
(NSIndexPath *)indexPath
{
    return 148;
}
```

7. Run iOS Simulator, you should see:

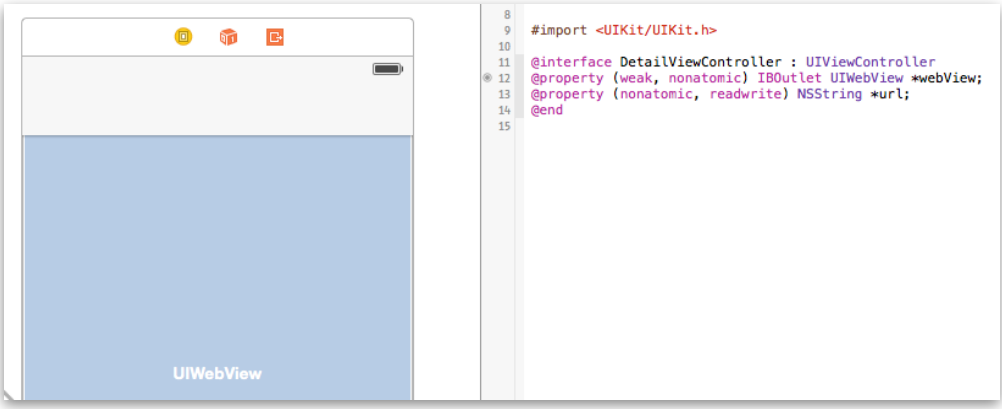


Detail view (1 point)

1. Go to Storyboard and add new UIViewController.
2. Connect it with ViewController (where you have UITableView) and select **push** option.
3. Select new ViewController and in Utilities area set DetailView as Storyboard ID
4. To the new ViewController add WebView control.



5. Create new class which is subclass of UIViewController and name it DetailViewController
6. Set new class to ViewControll which contain UIWebView
7. Creat property of UIWebView in DetailViewController class.
8. Add new NSString property to keep url address.



9. In viewDidLoad method in DetailViewController class run website from URL.

```
    if (self.webView) {
        [self.webView loadRequest:[NSURLRequest requestWithURL:[NSURL
                URLWithString:self.url]]];
    }
```

10. Go to RestaurantCel.h and create protocol with methods to notify ViewController in which cell specific button was tapped.

```
@protocol RestaurantCellDelegate;

@interface RestaurantCell : UITableViewCell
@property (weak, nonatomic) IBOutlet UIImageView *picture;
@property (weak, nonatomic) IBOutlet UILabel *name;
@property (weak, nonatomic) IBOutlet UILabel *street;
@property (weak, nonatomic) IBOutlet UILabel *city;

@property (nonatomic, assign) id <RestaurantCellDelegate> delegate;

- (IBAction)facebook:(id)sender;
- (IBAction)website:(id)sender;
@end

@protocol RestaurantCellDelegate <NSObject>
- (void)facebookButtonTapedInCell:(RestaurantCell*)cell;
- (void)websiteButtonTapedInCell:(RestaurantCell*)cell;
@end
```

11. In RestaurantCell.m action methods. Before sending a message check if protocol's methods are implemented in specific class.

```
- (IBAction)facebook:(id)sender {
    if ([self.delegate
respondsToSelector:@selector(facebookButtonTapedInCell:)]) {
        [self.delegate facebookButtonTapedInCell:self];
    }
}

- (IBAction)website:(id)sender {
    if ([self.delegate
respondsToSelector:@selector(websiteButtonTapedInCell:)]) {
        [self.delegate websiteButtonTapedInCell:self];
    }

}
```

12. In ViewController Interface section add new protocol.

```
@interface ViewController () <UITableViewDelegate,
UITableViewDataSource, RestaurantCellDelegate>
```

13. In cellForRowAtIndexPath, where you create a cell, set protocol to them.

```
cell.delegate = self;
```

14. Import DetailViewController class to ViewController, to open it with specific URL when user tap on the button.

```
#import "DetailViewController.h"
```

15. Implement RestaurantCellDelegate methods.

```
- (void)facebookButtonTapedInCell:(RestaurantCell*)cell
{
    NSIndexPath *indexPath = [self.tableView indexPathForCell:cell];
    NSDictionary *dict = _dataSource[indexPath.row];

    DetailViewController *controller = [self.storyboard
instantiateViewControllerWithIdentifier:@"DetailView"];
    controller.url = dict[@"facebook"];
    [self.navigationController pushViewController:controller
animated:YES];
}

- (void)websiteButtonTapedInCell:(RestaurantCell*)cell
{
    NSIndexPath *indexPath = [self.tableView indexPathForCell:cell];
    NSDictionary *dict = _dataSource[indexPath.row];

    DetailViewController *controller = [self.storyboard
instantiateViewControllerWithIdentifier:@"DetailView"];
    controller.url = dict[@"www"];
    [self.navigationController pushViewController:controller
animated:YES];
}
```

16. Run iOS Simulator and check how buttons work.

Working with maps (1 point)

1. Create new class MapViewAnnotation as a subclass of NSObject to create annotation for map.
2. Add property for coordinate and title.
3. Implement custom init method.

```
#import <MapKit/MapKit.h>

@interface MapViewAnnotation : NSObject <MKAnnotation>
@property (nonatomic,copy) NSString *title;
@property (nonatomic, readonly) CLLocationCoordinate2D coordinate;

- (id)initWithTitle:(NSString *) title AndCoordinate:
(CLLocationCoordinate2D)coordinate;
@end
```

```
- (id)initWithTitle:(NSString *) title AndCoordinate:
(CLLocationCoordinate2D)coordinate
{
    self = [super init];
    _title = title;
    _coordinate = coordinate;
    return self;
}
```

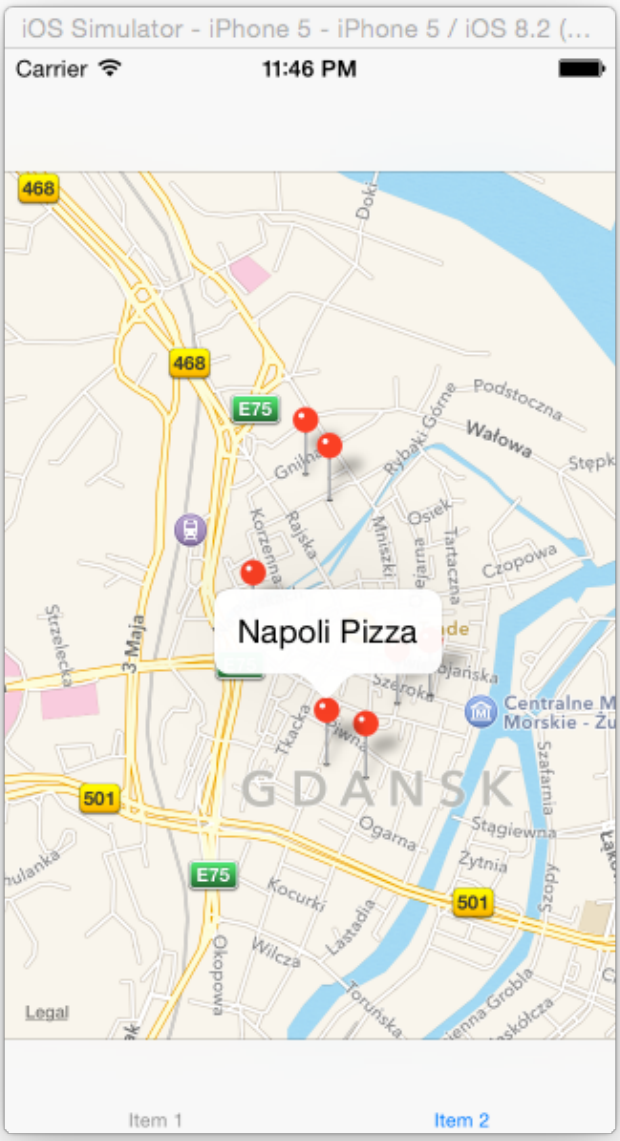
4. Go to MapViewController class and implement methods for preparing NSArray of annotations.

```
- (NSMutableArray*)createAnnotations
{
    NSMutableArray *annotations = [NSMutableArray new];

    NSArray *data = [DataManager dataFromPlist];

    for (NSDictionary *dict in data) {
        NSNumber *latitude = dict[@"lat"];
        NSNumber *longitude = dict[@"lon"];
        NSString *title = dict[@"name"];

        CLLocationCoordinate2D coord;
        coord.latitude = latitude.doubleValue;
        coord.longitude = longitude.doubleValue;
        MapViewAnnotation *annotation = [[MapViewAnnotation alloc]
initWithTitle:title AndCoordinate:coord];
        [annotations addObject:annotation];
    }

    return annotations;
}
```

5. Add annotations to MapView in viewDidLoad method.

```
[self.mapView addAnnotations:[self createAnnotations]];
```

6. Run iOS Simulator and check map.


iOS Simulator - iPhone 5 - iPhone 5 / iOS 8.2 (...
Carrier
11:46 PM
468
468
E75
Napoli Pizza
GDAŃSK
501
E75
501
Legal
Item 1
Item 2