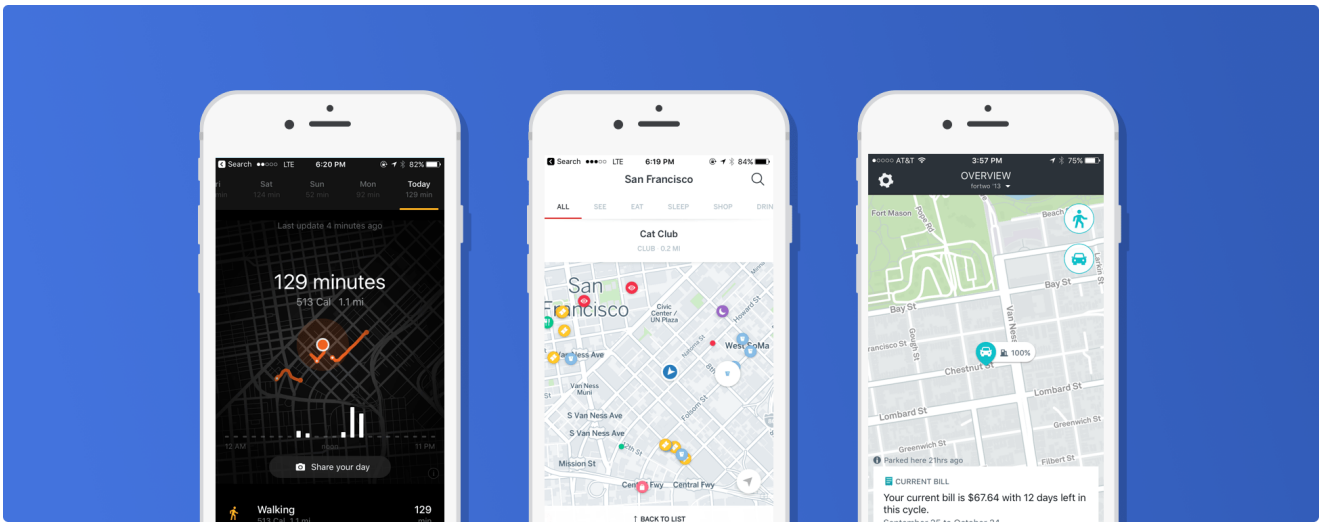
MapLibre Native for iOS

MapLibre Native for iOS is an open-source framework for embedding interactive map views with scalable, customizable vector maps into Cocoa Touch applications on iOS 9.0 and above using Objective-C, Swift, or Interface Builder. It takes stylesheets that conform to the MapLibre Style Specification, applies them to vector tiles that conform to the Mapbox Vector Tile Specification, and renders them using OpenGL.

For more information, check out the MapLibre Native for iOS repository and the full changelog online.



Installation

MapLibre Native for iOS may be installed as either a dynamic framework or a static framework. (To reduce the download size, the static framework is omitted from some distributions; you may need to download the full package from the release page.)

Integrating MapLibre Native for iOS requires Xcode 8.0 or higher.

{{DYNAMIC}}

Dynamic framework

This is the recommended workflow for manually integrating the SDK into an application:

1. Open the project editor, select your application target, then go to the General tab. Drag Mapbox.framework from the `dynamic` folder into the “Embedded Binaries” section. (Don’t drag it into the “Linked Frameworks and Libraries” section; Xcode will add it there automatically.) In the sheet that appears, make sure “Copy items if needed” is checked, then click Finish.

2. In the Build Phases tab, click the + button at the top and select “New Run Script Phase”. Enter the following code into the script text field:

```
bash
"${BUILT_PRODUCTS_DIR}/${FRAMEWORKS_FOLDER_PATH}/Mapbox.framework/strip-frameworks.sh"
```

(The last step, courtesy of Realm, is required for working around an iOS App Store bug when archiving universal binaries.)

{{/DYNAMIC}}
{{STATIC}}

Static framework

You can alternatively install the SDK as a static framework:

1. Drag Mapbox.bundle and Mapbox.framework from the `static` folder into the Project navigator. In the sheet that appears, make sure “Copy items if needed” is checked, then click Finish. Open the project editor and select your application target to verify that the following changes occurred automatically:
    - In the General tab, Mapbox.framework is listed in the “Linked Frameworks and Libraries” section.
    - In the Build Settings tab, the “Framework Search Paths” (`FRAMEWORK_SEARCH_PATHS`) build setting includes the directory that contains Mapbox.framework. For most projects, the default value of `$(inherited) $(PROJECT_DIR)` should be sufficient.
    - In the Build Phases tab, Mapbox.bundle is listed in the “Copy Bundle Resources” build phase.
2. Add the following Cocoa Touch frameworks and libraries to the “Linked Frameworks and Libraries” section:
    - GLKit.framework
    - ImageIO.framework
    - MobileCoreServices.framework
    - QuartzCore.framework
    - SystemConfiguration.framework
    - libc++.tbd
    - libsqlite3.tbd
    - libz.tbd
3. In the Build Settings tab, find the Other Linker Flags setting and add `-ObjC`.

{{/STATIC}}

Configuration

1. Some vector tiles servers require a API key. In the project editor, select the application target, then go to the Info tab. Under the “Custom iOS Target Properties” section, set `MLNApiKey` to your api key.

2. In order to show the user's current location on the map, the SDK must ask for the user's permission to access Location Services. Go to the Info tab of the project editor. If your application supports iOS 7, set the `NSLocationUsageDescription` key to a message that explains to the user what their location is used for. If your application supports iOS 8 and above, set the `NSLocationAlwaysUsageDescription` and/or `NSLocationWhenInUseUsageDescription` key to this message instead.

    - In the General tab, Settings.bundle is listed in the "Copy Bundle Resources" build phase.

Usage

In a storyboard or XIB, add a view to your view controller. (Drag View from the Object library to the View Controller scene on the Interface Builder canvas.) In the Identity inspector, set the view's custom class to `MLNMapView`. If you need to manipulate the map view programmatically:

1. Switch to the Assistant Editor.
2. Import the `Mapbox` module.
3. Connect the map view to a new outlet in your view controller class. (Control-drag from the map view in Interface Builder to a valid location in your view controller implementation.) The resulting outlet declaration should look something like this:

```
// ViewController.m
@import Mapbox;

@interface ViewController : UIViewController

@property (strong) IBOutlet MLNMapView *mapView;

@end
```

```
// ViewController.swift
import Mapbox

class ViewController: UIViewController {
    @IBOutlet var mapView: MLNMapView!
}
```

Full API documentation is included in this package, within the `documentation` folder.

We welcome your bug reports, feature requests, and contributions.