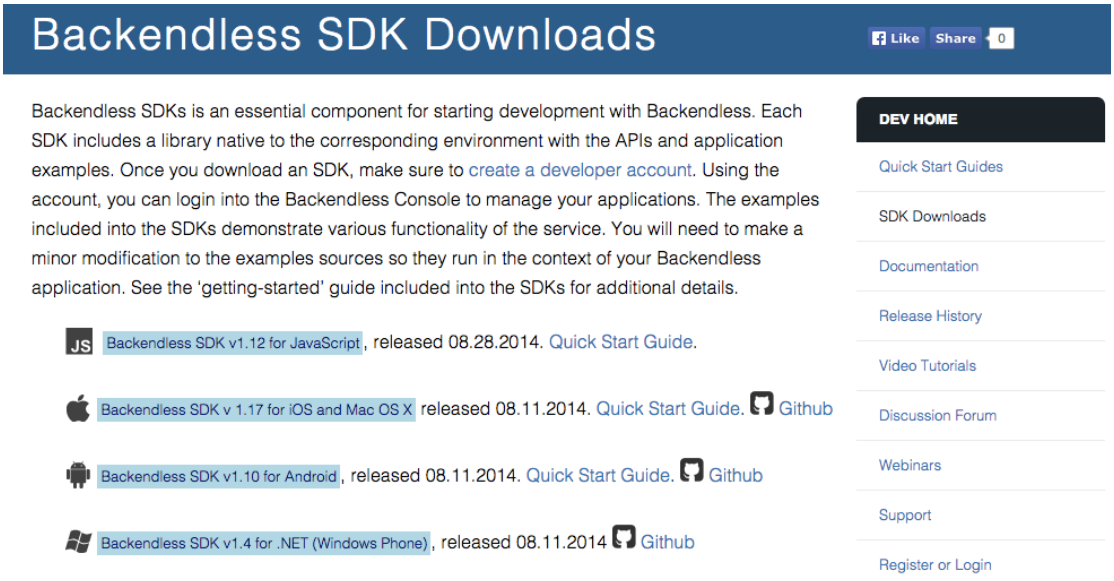
Qiuck Start Guide for iOS

This guide will help you get up and running with Backendless quickly so you can develop iOS applications with it. At the end of the guide you will have a configured working environment and a basic project with the code communicating with the Backendless services.

1. Login to your Backendless account or register to create a new one.
2. Download Backendless SDK for iOS from the Backendless SDK Downloads page and unzip it to a temporary folder.

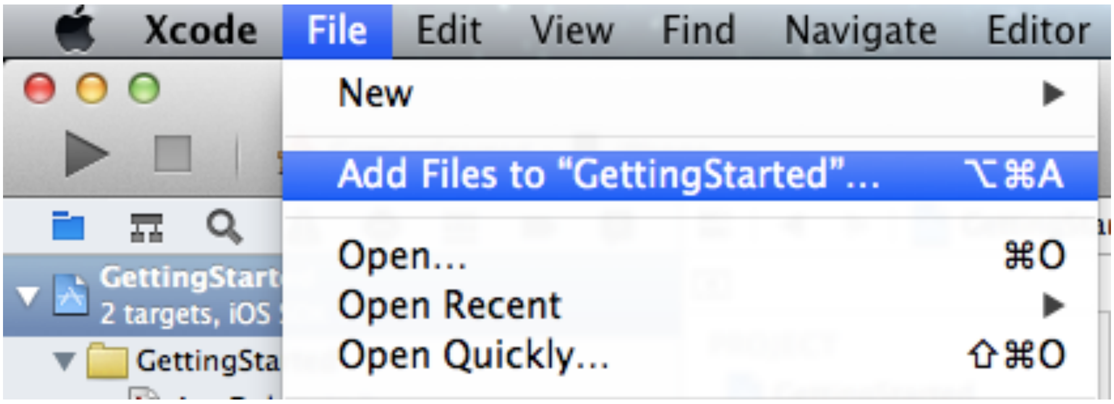


Backendless SDK Downloads

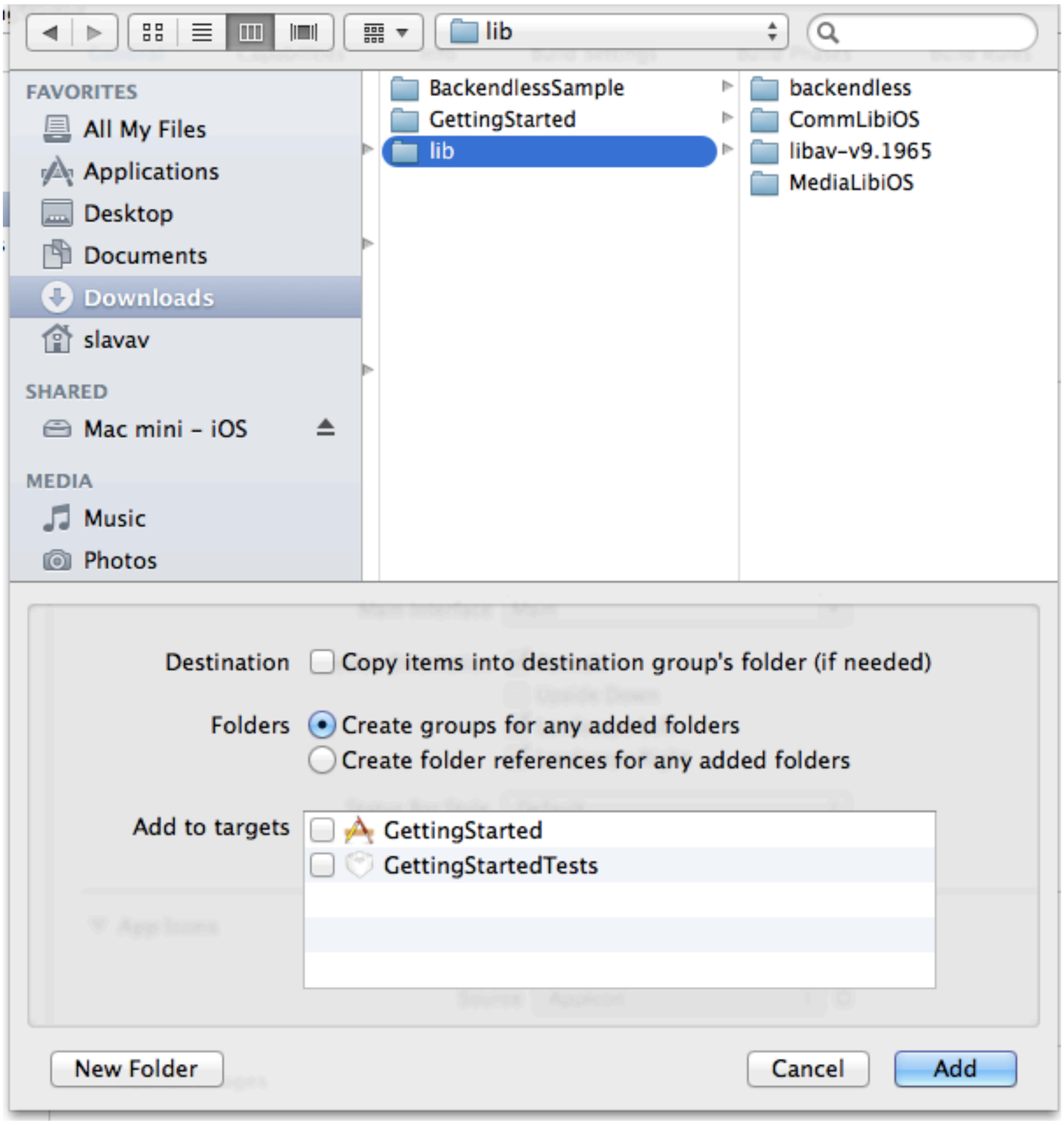
Backendless SDKs is an essential component for starting development with Backendless. Each SDK includes a library native to the corresponding environment with the APIs and application examples. Once you download an SDK, make sure to create a developer account. Using the account, you can login into the Backendless Console to manage your applications. The examples included into the SDKs demonstrate various functionality of the service. You will need to make a minor modification to the examples sources so they run in the context of your Backendless application. See the 'getting-started' guide included into the SDKs for additional details.

- Backendless SDK v1.12 for JavaScript, released 08.28.2014. Quick Start Guide.
- Backendless SDK v 1.17 for iOS and Mac OS X released 08.11.2014. Quick Start Guide. Github
- Backendless SDK v1.10 for Android, released 08.11.2014. Quick Start Guide. Github
- Backendless SDK v1.4 for .NET (Windows Phone), released 08.11.2014 Github

3. Create a new iOS application project or open an existing project. Copy the “lib” folder from SDK to your project folder.
4. Mark the project and choose “File”->”Add Files to …” menu item:



then in window choose the “lib” folder from the project folder. Please note that the “Add to tagets” checkboxes must be unchecked. Push “Add” button.



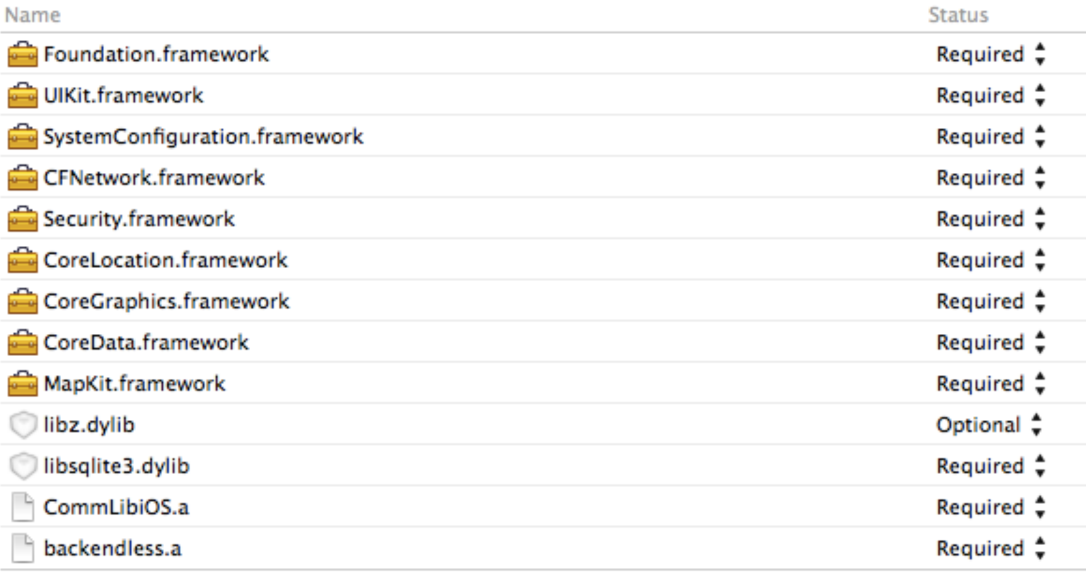
Destination □ Copy items into destination group's folder (if needed)

Folders ● Create groups for any added folders
○ Create folder references for any added folders

Add to targets □ GettingStarted
□ GettingStartedTests

5. Add frameworks and libraries and make sure the complete list includes the following:
   - Foundation.framework
   - CoreData.framework
   - CFNetwork.framework
   - CoreLocation.framework
   - MapKit.framework

- SystemConfiguration.framework
- Security.framework
- CoreGraphics.framework
- UIKit.framework
- libsqlite3.dylib
- libz.dylib
- CommLibiOS.a
- backendless.a

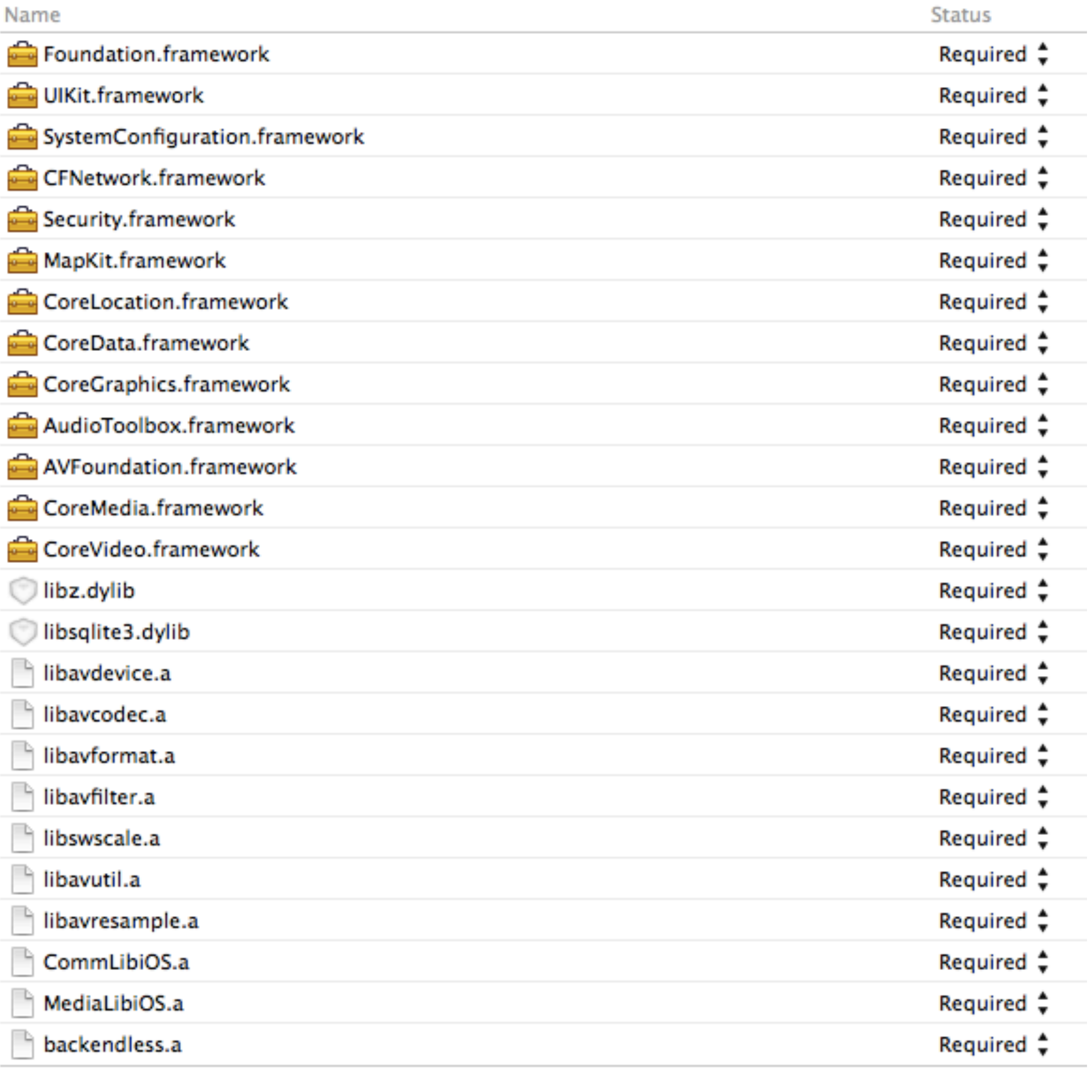


Name	Status
Foundation.framework	Required
UIKit.framework	Required
SystemConfiguration.framework	Required
CFNetwork.framework	Required
Security.framework	Required
CoreLocation.framework	Required
CoreGraphics.framework	Required
CoreData.framework	Required
MapKit.framework	Required
libz.dylib	Optional
libsqlite3.dylib	Required
CommLibiOS.a	Required
backendless.a	Required

If the application will use the Backendless Media API, make sure to also add the following:

- AudioToolbox.framework
- AVFoundation.framework
- CoreMedia.framework
- CoreVideo.framework
- libavdevice.a
- libavresample.a
- libavformat.a
- libswscale.a
- libavutil.a
- libavfilter.a
- libavcodec.a

- MediaLibiOS.a

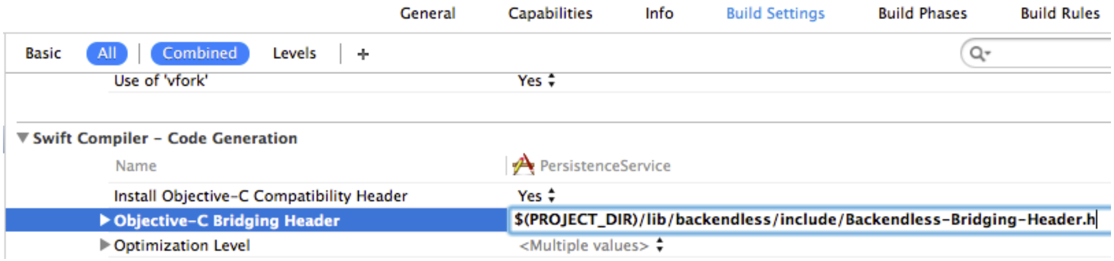


Name	Status
Foundation.framework	Required
UIKit.framework	Required
SystemConfiguration.framework	Required
CFNetwork.framework	Required
Security.framework	Required
MapKit.framework	Required
CoreLocation.framework	Required
CoreData.framework	Required
CoreGraphics.framework	Required
AudioToolbox.framework	Required
AVFoundation.framework	Required
CoreMedia.framework	Required
CoreVideo.framework	Required
libz.dylib	Required
libsqlite3.dylib	Required
libavdevice.a	Required
libavcodec.a	Required
libavformat.a	Required
libavfilter.a	Required
libswscale.a	Required
libavutil.a	Required
libavresample.a	Required
CommLibiOS.a	Required
MediaLibiOS.a	Required
backendless.a	Required

6. If your project use Objective-C, add the following import statement to your application delegate .m file

```
#import "Backendless.h"
```

7. If your project use Swift, add the following option to the Build Setting:

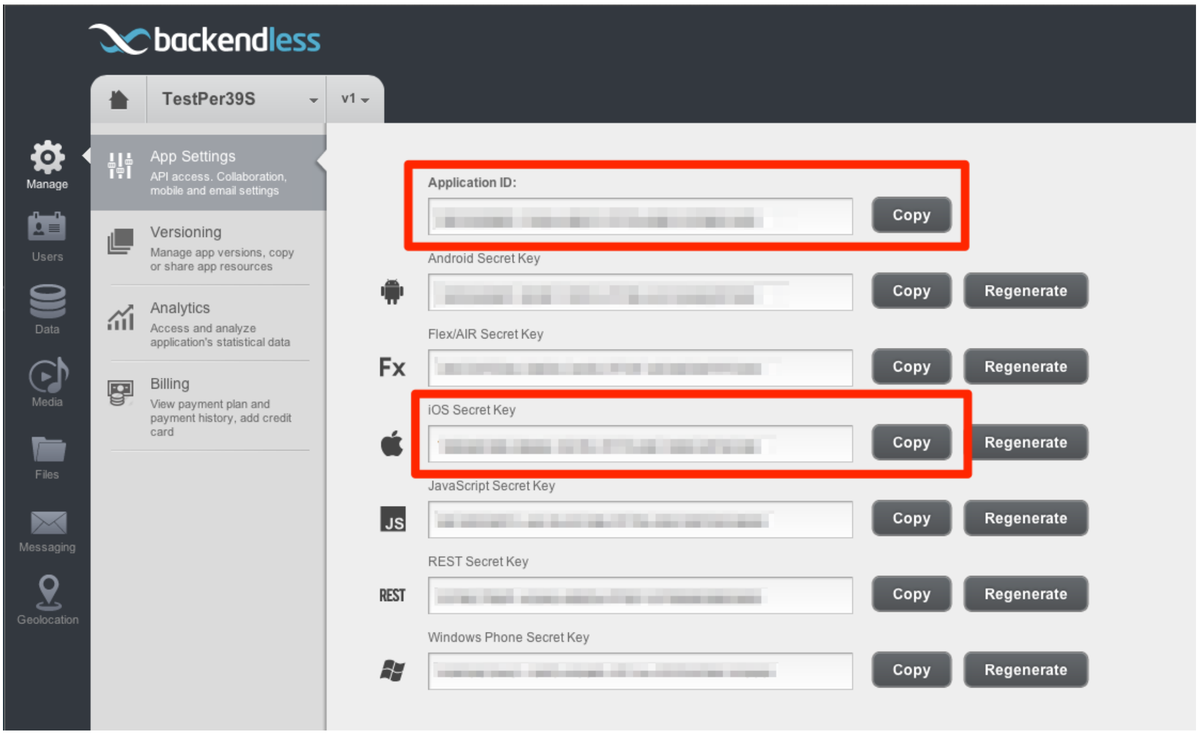


Use of 'vfork' Yes

▼ Swift Compiler - Code Generation

Name	PersistenceService
Install Objective-C Compatibility Header	Yes
▶ Objective-C Bridging Header	$(PROJECT_DIR)/lib/backendless/include/Backendless-Bridging-Header.h
▶ Optimization Level	<Multiple values>

8. Get your Backendless application and secret keys for iOS from the Backendless Console. The keys can be found on the Manage > App Settings section:



9. Add Backendless application initialization code block to your application delegate .m file. The recommended approach is to add the initialization logic inside of the “-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions” method as shown below:

```
 - (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
        [backendless initApp:{Application_id} secret:{iOS_secret_key}
version:{Application_version}];
        return YES;
 }
```

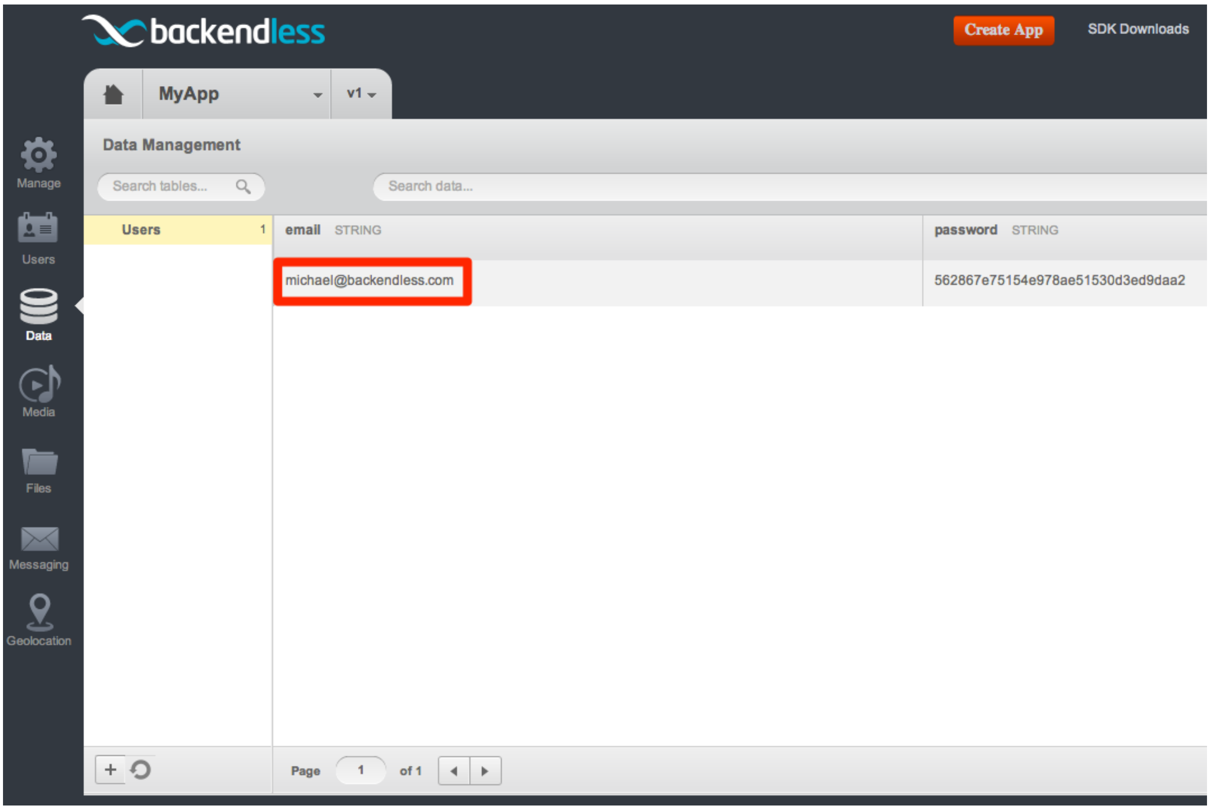
10. Add some code from Backendless API to test your application. For example, add the code below inside of the “-(BOOL)application:(UIApplication *)application

didFinishLaunchingWithOptions:(NSDictionary *)launchOptions" method to register a new Backendless user:

□

```
BackendlessUser *user = [BackendlessUser new];
user.email = @"michael@backendless.com";
user.password = @"my_super_password";
[backendless.userService registering:user];
```

11. Run your application. A new user will be registered with Backendless. You can verify the registered user in the Data section of the Backendless Console:



email STRING	password STRING
michael@backendless.com	562867e75154e978ae51530d3ed9daa2

12. Another Backendless API usage to try is to create and save some plain object from the iOS client. Create a new "Comment" class:

□

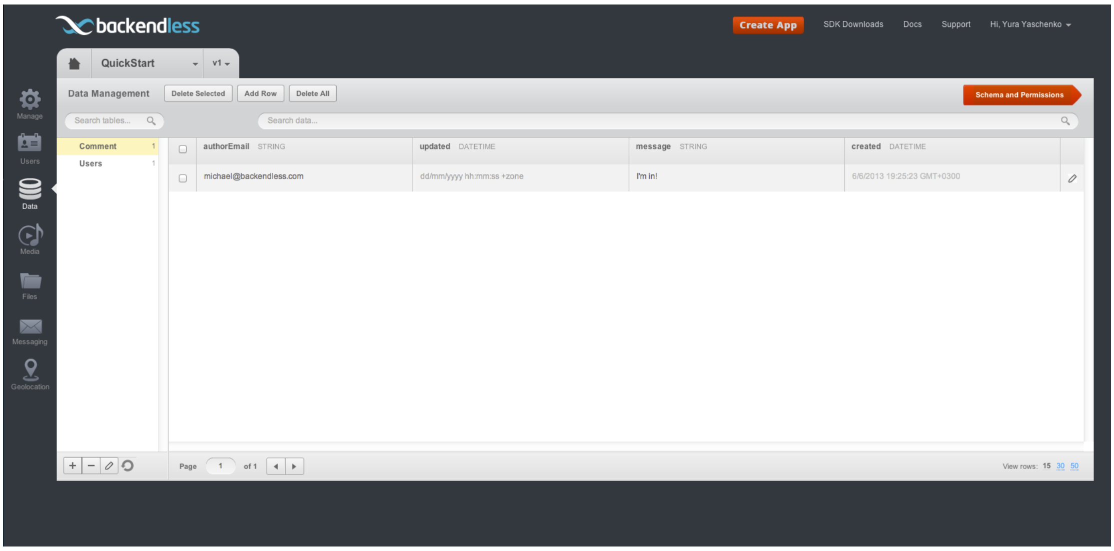
```
@interface Comment : NSObject
@property (nonatomic, strong) NSString *objectId;
@property (nonatomic, strong) NSString *message;
@property (nonatomic, strong) NSString *authorEmail;

+(Comment *)commentWithMessage:(NSString *)message
authorEmail:(NSString *)authorEmail;
@end
```

```

@implementation Comment
@synthesize authorEmail, message, objectId;

+(Comment *)commentWithMessage:(NSString *)message
authorEmail:(NSString *)authorEmail
{
    Comment *comment = [Comment new];
    comment.message = message;
    comment.authorEmail = authorEmail;
    return comment;
}
@end
```

13. Use the code snippet below to store a “Comment” instance in Backendless:

```
[backendless.persistenceService save:[Comment commentWithMessage:@"I'm in!" authorEmail:user.email]];
```

Now you can verify in the Backendless Console that the “Comment” data table has been created. Notice the table’s columns match the properties of the Comment class.



	authorEmail STRING	updated DATETIME	message STRING	created DATETIME
□	michael@backendless.com	dd/mm/yyyy hh:mm:ss +zone	I'm in!	6/6/2013 19:25:23 GMT+0300

Enjoy!

Using CocoaPods

CocoaPods manages library dependencies for your Xcode projects.

The dependencies for your projects are specified in a single text file called a Podfile. CocoaPods will resolve dependencies between libraries, fetch the resulting source code, then link it together in an Xcode workspace to build your project.

Make sure you have the Cocoapods ruby gem installed your system. If you don't please follow the directions at CocoaPods Getting Started , or just fire up a Terminal window and run `$ sudo gem install cocoapods` .

Creating a new Xcode project with CocoaPods

Before you begin you can check Backendless-ios-SDK folder of CocoaPods Specs repository, and choose the library version you will use (the latest is recommended if you don’t have another reason).

To create a new project with CocoaPods, follow these simple steps:

1. Create a new project in Xcode as you would normally, then close this project.

2. Open a Terminal window, and `$ cd` into your project directory.

3. Create a Podfile. This can be done by running `$ touch Podfile`.

4. Open your Podfile using your favorite text editor (or Xcode), and add a text that looks like this:

```
platform :ios, '7.0'

pod 'Backendless-ios-SDK', '~>1.19.1'
```

The first line specifies the platform and version supported, the second line specifies the name of Backendless folder in CocoaPods Specs repository and library version you choose.

5. Save Podfile, return to Terminal window and run `$ pod install`. Once all of the pod data is downloaded, Xcode project workspace file will be created. This should be the file you use everyday to create your app.

6. Open *.xcworkspace* file to launch your project, and build it.

Enjoy!