

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help

Smack > iPhone 6s Running Smack on iPhone 6s 32

Smack > Smack > Services > AuthService.swift > AuthService

AuthService.swift

```
import Foundation
class AuthService {
    static let instance = AuthService()
    let defaults = UserDefaults.standard
    var isLoggedIn : Bool {
        get {
            return defaults.bool(forKey: LOGGED_IN_KEY)
        }
        set {
            defaults.set(newValue, forKey: LOGGED_IN_KEY)
        }
    }
    var authToken: String {
        get {
            return defaults.value(forKey: TOKEN_KEY) as! String
        }
        set {
            defaults.set(newValue, forKey: TOKEN_KEY)
        }
    }
    var userEmail: String {
        get {
            return defaults.value(forKey: USER_EMAIL) as! String
        }
        set {
            defaults.set(newValue, forKey: USER_EMAIL)
        }
    }
}
```

AuthService.swift

Smack

Smack

Services

AuthService.swift

Utilities

Model

View

GradientView.swift

Controller

ChatVC.swift

ChannelVC.swift

LoginVC.swift

CreateAccountVC.swift

Supporting Files

SWRevealViewController.h

SWRevealViewController.m

Smack-Bridging-Header.h

AppDelegate.swift

Main.storyboard

Assets.xcassets

LaunchScreen.storyboard

Info.plist

Products

Pods

Frameworks

Pods



The screenshot shows the Xcode interface with the following details:

- File menu:** Xcode, File, Edit, View, Find, Navigate, Editor, Product, Debug, Source Control, Window, Help.
- Toolbar:** Standard Xcode toolbar with icons for play, stop, run, etc.
- Project Navigator:** Shows the project structure under "Smack".
- Search Bar:** "Smack > iPhone 6s" and "Running Smack on iPhone 6s".
- Status Bar:** "AuthService.swift" and "No Selection".
- Alerts:** 32 warnings indicated by a yellow triangle icon.
- Code Editor:** Displays the `AuthService.swift` file content. The line `import Alamofire` is highlighted with a red rounded rectangle.

```
1 //  
2 // AuthService.swift  
3 // Smack  
4 //  
5 // Created by Jonny B on 7/17/17.  
6 // Copyright © 2017 Jonny B. All rights reserved.  
7 //  
8  
9 import Foundation  
10 import Alamofire  
11  
12 class AuthService {  
13  
14     static let instance = AuthService()  
15  
16     let defaults = UserDefaults.standard  
17  
18     var isLoggedIn : Bool {  
19         get {  
20             return defaults.bool(forKey: LOGGED_IN_KEY)  
21         }  
22         set {  
23             defaults.set(newValue, forKey: LOGGED_IN_KEY)  
24         }  
25     }  
26 }
```

## Resources available

will be listed as either `Alamofire iOS`, `Alamofire macOS`, `Alamofire tvOS` or `Alamofire watchOS`.

- And that's it!

The `Alamofire.framework` is automagically added as a target dependency, linked framework and embedded framework in a copy files build phase which is all you need to build on the simulator and a device.

## Usage

# Alamofire

### Making a Request

```
import Alamofire

Alamofire.request("https://httpbin.org/get")
```

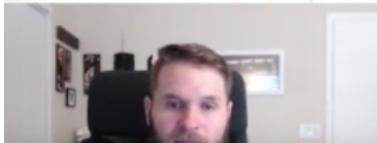
### Response Handling

Handling the `Response` of a `Request` made in Alamofire involves chaining a response handler onto the `Request`.

```
Alamofire.request("https://httpbin.org/get").responseJSON { response in
    print("Request: \(String(describing: response.request))") // original url request
    print("Response: \(String(describing: response.response))") // http url response
    print("Result: \(response.result)") // response serialization result

    if let json = response.result.value {
        print("JSON: \(json)") // serialized json response
    }

    if let data = response.data, let utf8Text = String(data: data, encoding: .utf8) {
        print("Data: \(utf8Text)") // original server data as UTF8 string
    }
}
```



## Response Data Handler

The `responseData` handler uses the `responseDataSerializer` (the object that serializes the server data into some other type) to extract the `Data` returned by the server. If no errors occur and `Data` is returned, the response `Result` will be a `.success` and the value will be of type `Data`.

```
Alamofire.request("https://httpbin.org/get").responseData { response in
    debugPrint("All Response Info: \(response)")

    if let data = response.result.value, let utf8Text = String(data: data, encoding: .utf8) {
        print("Data: \(utf8Text)")
    }
}
```

## Response String Handler

The `responseString` handler uses the `responseStringSerializer` to convert the `Data` returned by the server into a `String` with the specified encoding. If no errors occur and the server data is successfully serialized into a `String`, the response `Result` will be a `.success` and the value will be of type `String`.

```
Alamofire.request("https://httpbin.org/get").responseString { response in
    print("Success: \(response.result.isSuccess)")
    print("Response String: \(response.result.value)")
}
```

If no encoding is specified, Alamofire will use the text encoding specified in the `HTTPURLResponse` from the server. If the text encoding cannot be determined by the server response, it defaults to `.isoLatin1`.

## Response JSON Handler

Runner Import

Builder Team Library IN SYNC ... Jonny B Examples (0) Filter

No Environment

Find all mess 1. Auth Regis 2. Auth Login User 3. Add User +

1. Auth Register User

POST http://localhost:3005/v1/account/register Params Send Save

Authorization Headers (1) Body Pre-request Script Tests Cookies Code

Body (1) x-www-form-urlencoded raw binary JSON (application/json)

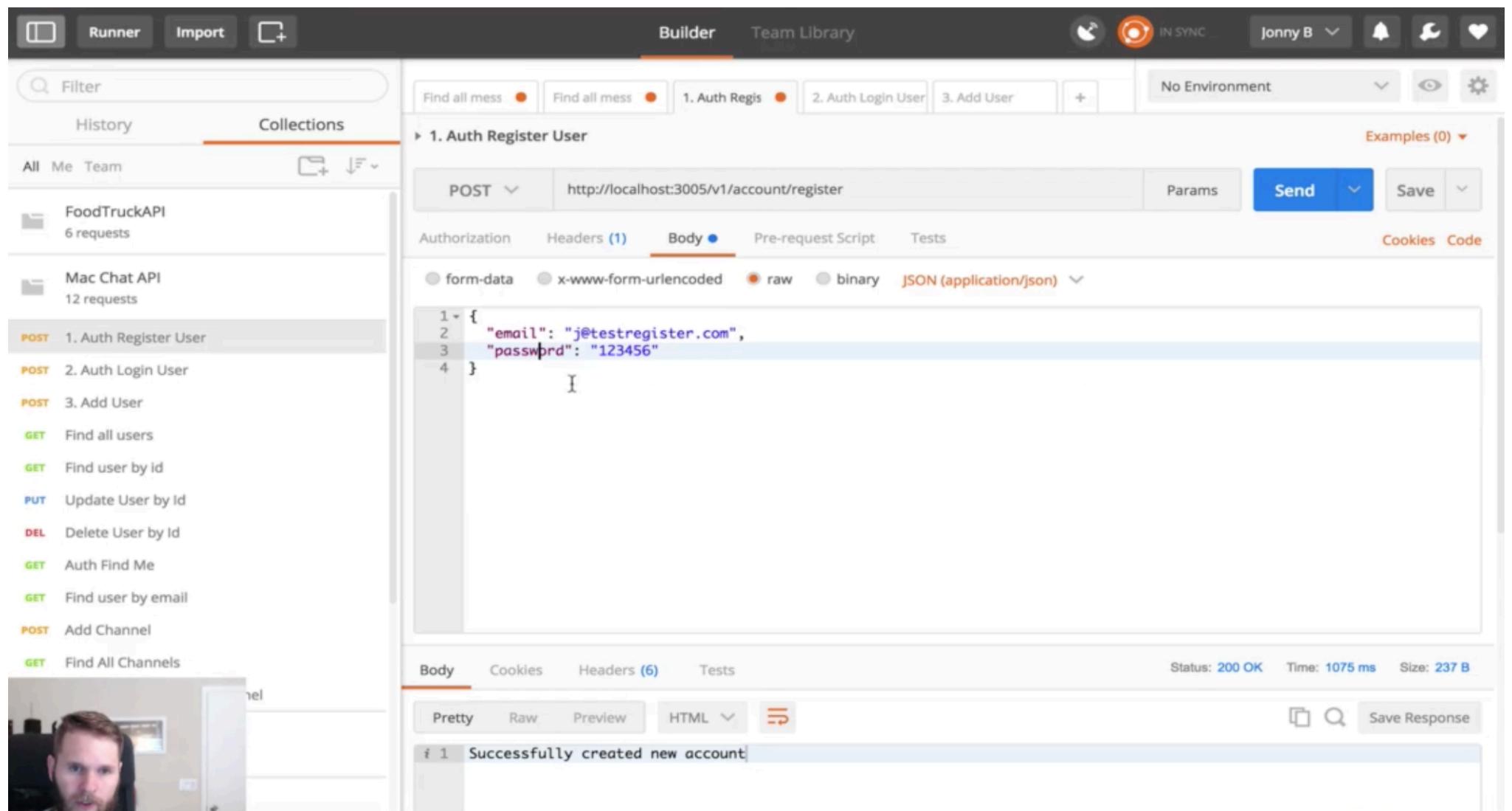
```
1+ {  
2 "email": "j@testregister.com",  
3 "password": "123456"  
4 }
```

POST 1. Auth Register User  
POST 2. Auth Login User  
POST 3. Add User  
GET Find all users  
GET Find user by id  
PUT Update User by Id  
DEL Delete User by Id  
GET Auth Find Me  
GET Find user by email  
POST Add Channel  
GET Find All Channels

Status: 200 OK Time: 1075 ms Size: 237 B

Pretty Raw Preview HTML Save Response

i 1 Successfully created new account



Safari File Edit View History Bookmarks Develop Window Help

Heroku, Inc.

Bitbucket Devslopes - Dropbox Android Developers SurePay Apple Twitter

Alamofire/Alamofire: Elegant HTTP Networking in Swift chattychatjb · Settings | Heroku https://chattychatjb.herokuapp.com

HEROKU Jump to Favorites, Apps, Pipelines, Spaces...

**Buildpacks**

Buildpacks are scripts that are run when your app is deployed. They are used to install dependencies for your app and configure your environment.

[Find New Buildpacks at Heroku Elements](#)

**Add buildpack**

heroku/nodejs X

**Domains and certificates**

Add your custom domains here then [point your DNS to Heroku](#).

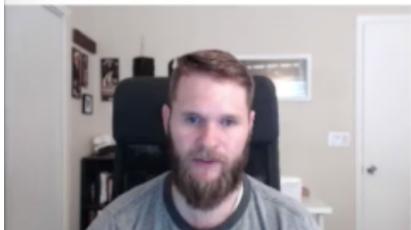
Domain Your app can be found at <https://chattychatjb.herokuapp.com/> Refresh status

SSL Upgrade to paid dynos to configure Heroku SSL

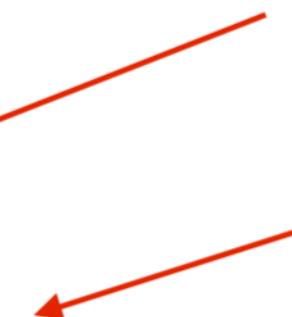
**Add domain**

Custom domains will appear here  
Custom domains allow you to access your app via one or more non-Heroku domain names (for example, [www.yourcustomdomain.com](http://www.yourcustomdomain.com))

Select a new owner...



```
1 //  
2 // Constants.swift  
3 // Smack  
4 //  
5 // Created by Jonny B on 7/14/17.  
6 // Copyright © 2017 Jonny B. All rights reserved.  
7 //  
8  
9 import Foundation  
10  
11 typealias CompletionHandler = (_ Success: Bool) -> ()  
12  
13 // URL Constants  
14 let BASE_URL = "https://chattychatjb.herokuapp.com/v1/"  
15 let URL_REGISTER = "\(BASE_URL)account/register"  
16  
17 // Segues  
18 let TO_LOGIN = "toLogin"  
19 let TO_CREATE_ACCOUNT = "toCreateAccount"  
20 let UNWIND = "unwindToChannel"  
21  
22 // User Defaults  
23 let TOKEN_KEY = "token"  
24 let LOGGED_IN_KEY = "loggedIn"  
25 let USER_EMAIL = "userEmail"  
26
```



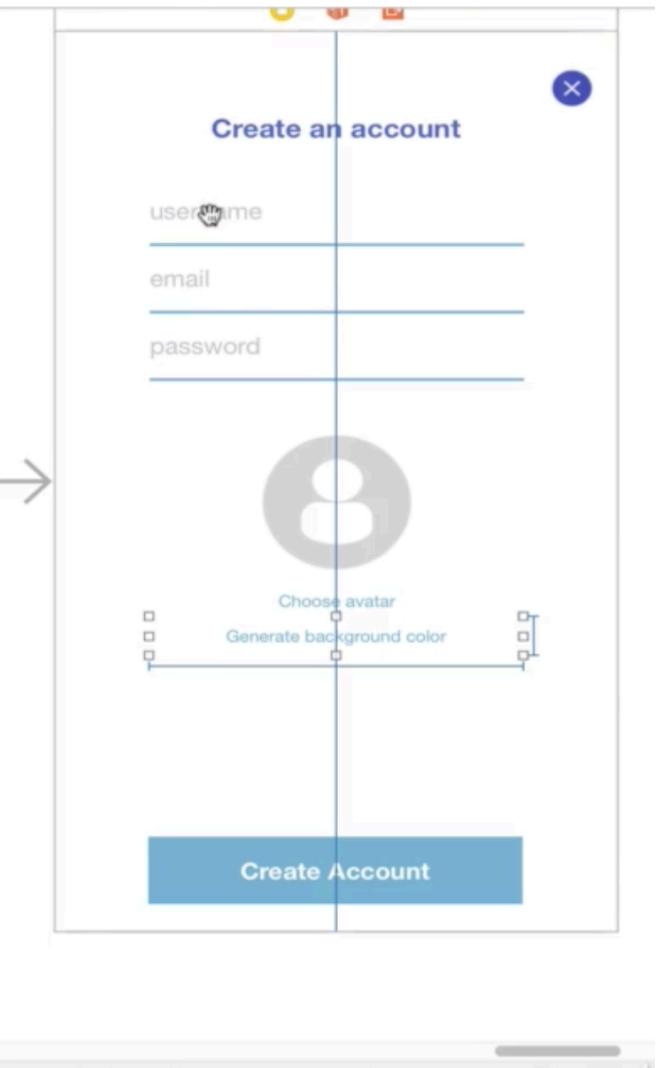
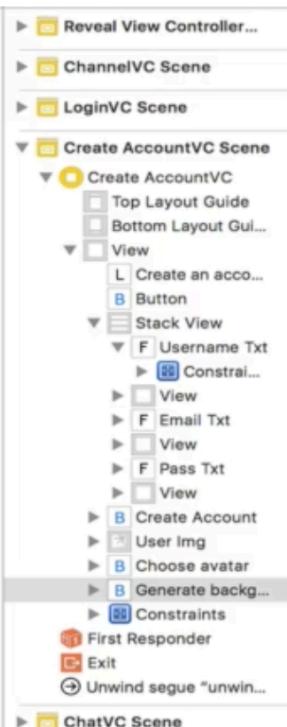


```
41         defaults.set(newValue, forKey: USER_EMAIL)
42     }
43 }
44
45 func registerUser(email: String, password: String, completion:
46                     @escaping CompletionHandler) {
47
48     let lowerCaseEmail = email.lowercased()
49
50     let header = [
51         "Content-Type": "application/json; charset=utf-8"
52     ]
53
54     let body: [String: Any] = [
55         "email": lowerCaseEmail,
56         "password": password
57     ]
58
59     Alamofire.request(URL_REGISTER, method: .post, parameters: body,
60                       encoding: JSONEncoding.default, headers: header).responseString
61     { (response) in
62
63         if response.result.error == nil {
64             completion(true)
65         } else {
66             completion(false)
67             debugPrint(response.result.error as Any)
68         }
69     }
70
71 }
```

```

6 // Copyright © 2017 Jonny B. All rights reserved.
7 //
8
9 import UIKit
10
11 class CreateAccountVC: UIViewController {
12
13     // Outlets
14     @IBOutlet weak var usernameTxt: UITextField!
15     @IBOutlet weak var emailTxt: UITextField!
16     @IBOutlet weak var passTxt: UITextField!
17     @IBOutlet weak var userImg: UIImageView!
18
19
20     override func viewDidLoad() {
21         super.viewDidLoad()
22
23     }
24
25     @IBAction func createAccntPressed(_ sender: Any) {
26
27     }
28
29     @IBAction func pickAvatarPressed(_ sender: Any) {
30
31     }
32
33     @IBAction func pickBGColorPressed(_ sender: Any) {
34
35     }
36
37     @IBAction func closePressed(_ sender: Any) {
38         performSegue(withIdentifier: UNWIND, sender: nil)
}

```





The screenshot shows the Xcode IDE with a project named "Smack". The left sidebar displays the project structure, including folders for Smack, Services, Utilities, Model, View, Controller, and Supporting Files, along with files like AuthService.swift, Constants.swift, ChatVC.swift, ChannelVC.swift, LoginVC.swift, CreateAccountVC.swift, SWRevealViewController.h, SWRevealViewController.m, Smack-Bridging-Header.h, AppDelegate.swift, Main.storyboard, Assets.xcassets, LaunchScreen.storyboard, and Info.plist. The right pane shows the code for CreateAccountVC.swift.

```
9 import UIKit
10
11 class CreateAccountVC: UIViewController {
12
13     // Outlets
14     @IBOutlet weak var usernameTxt: UITextField!
15     @IBOutlet weak var emailTxt: UITextField!
16     @IBOutlet weak var passTxt: UITextField!
17     @IBOutlet weak var userImg: UIImageView!
18
19
20     override func viewDidLoad() {
21         super.viewDidLoad()
22
23     }
24
25     @IBAction func createAccntPressed(_ sender: Any) {
26         guard let email = emailTxt.text , emailTxt.text != "" else { return }
27             }
28         guard let pass = passTxt.text , passTxt.text != "" else { return }
29         AuthService.instance.registerUser(email: email, password: pass)
30             { (success) in
31                 if success {
32                     print("registered user!")
33                 }
34             }
35
36     @IBAction func pickAvatarPressed(_ sender: Any) {
37     }
38
39     @IBAction func pickBGColorPressed(_ sender: Any) {
40 }
```

The screenshot shows the Xcode interface with the project navigation bar at the top. The project name is "Smack" and it is running on an "iPhone 6s". The code editor displays the "RoundedButton.swift" file under the "View" folder. The file contains Swift code for a rounded button class.

```
6 // Copyright © 2017 Jonny B. All rights reserved.
7 //
8
9 import UIKit
10
11 @IBDesignable
12 class RoundedButton: UIButton {
13
14     @IBInspectable var cornerRadius: CGFloat = 3.0 {
15         didSet {
16             self.layer.cornerRadius = cornerRadius
17         }
18     }
19
20     override func awakeFromNib() {
21         self.setupView()
22     }
23
24     override func prepareForInterfaceBuilder() {
25         super.prepareForInterfaceBuilder()
26         self.setupView()
27     }
28
29     func setupView() {
30         self.layer.cornerRadius = cornerRadius
31     }
32 }
33
```