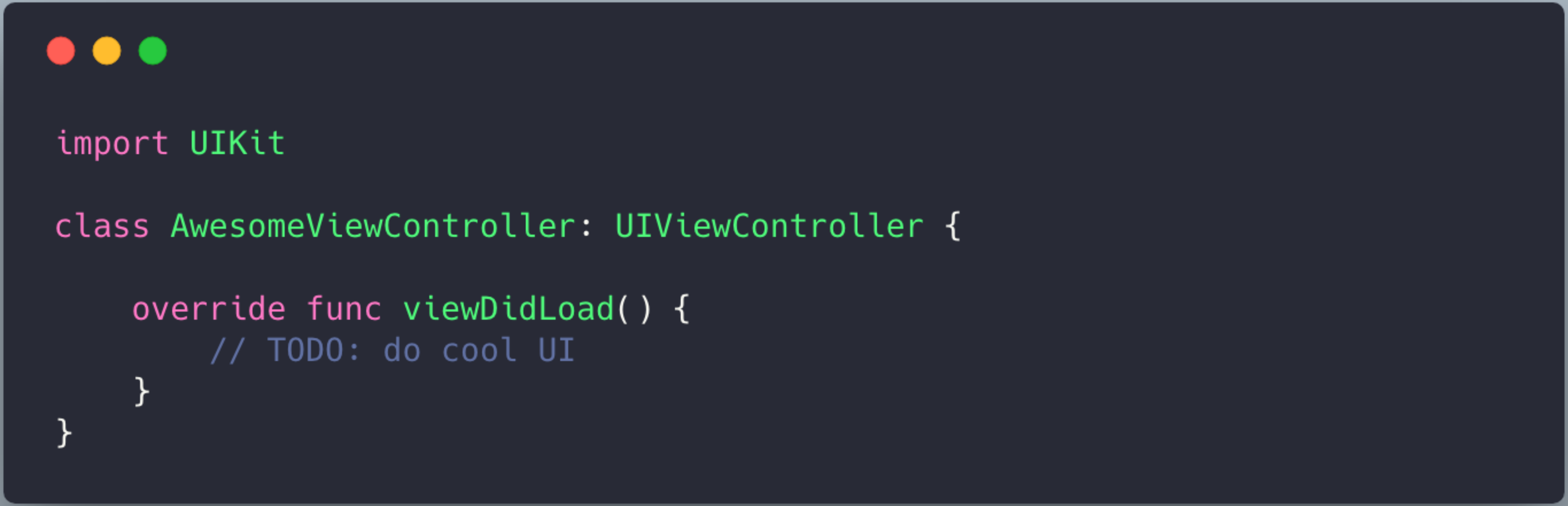
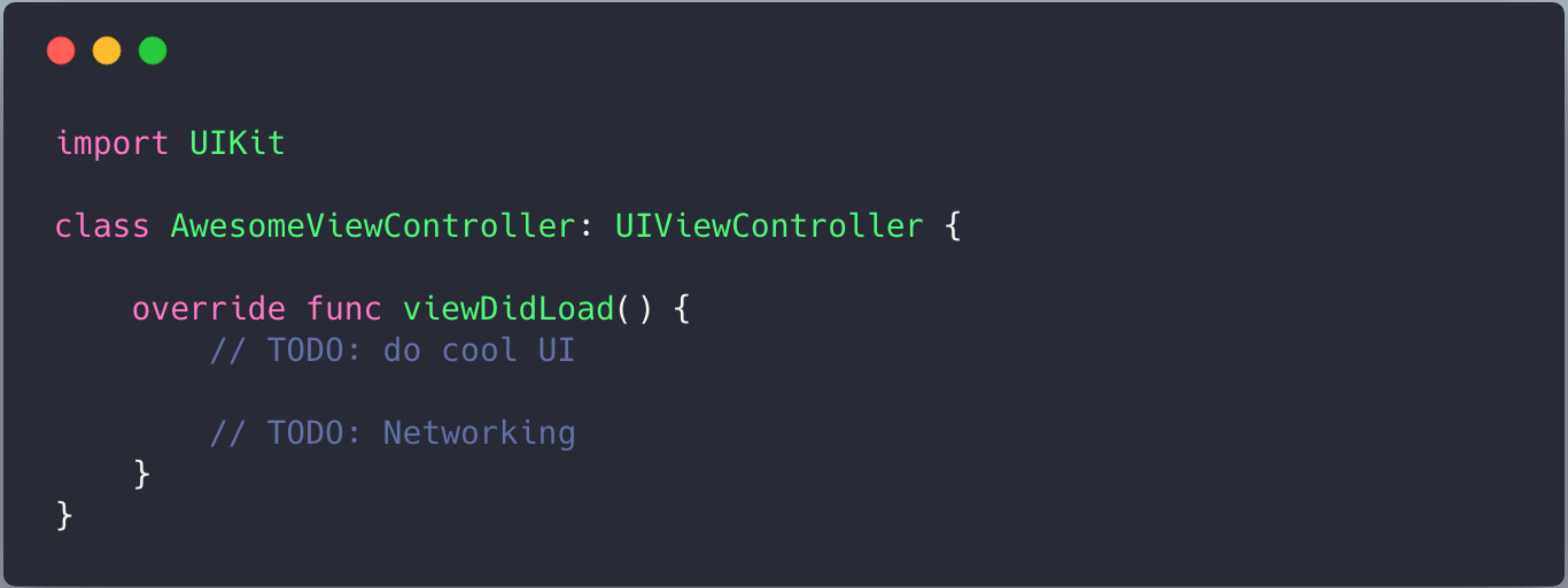
The Life and Times of a Networking Client

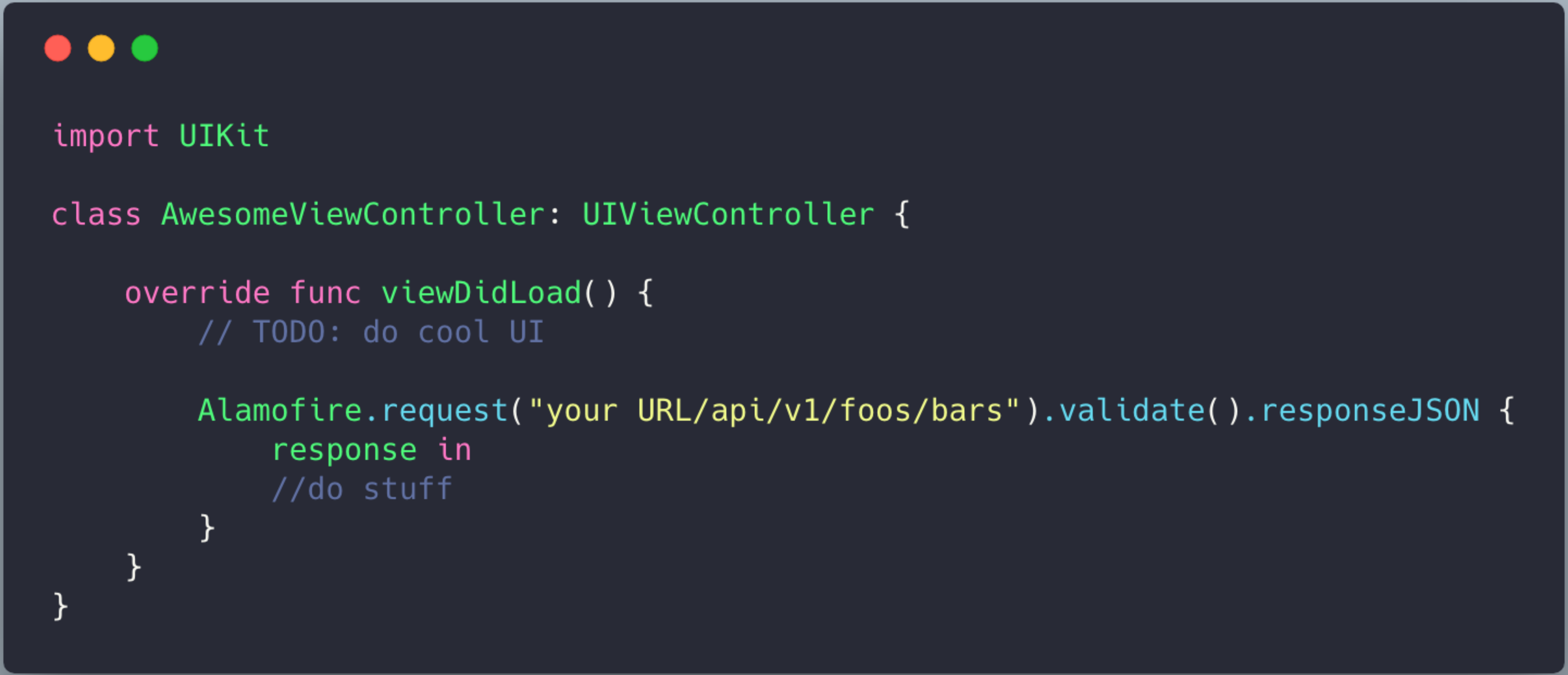
(ALT-CRINGE) NETWORKING CLIENT KI KAHAANI, ROBIN KI ZUBAANI



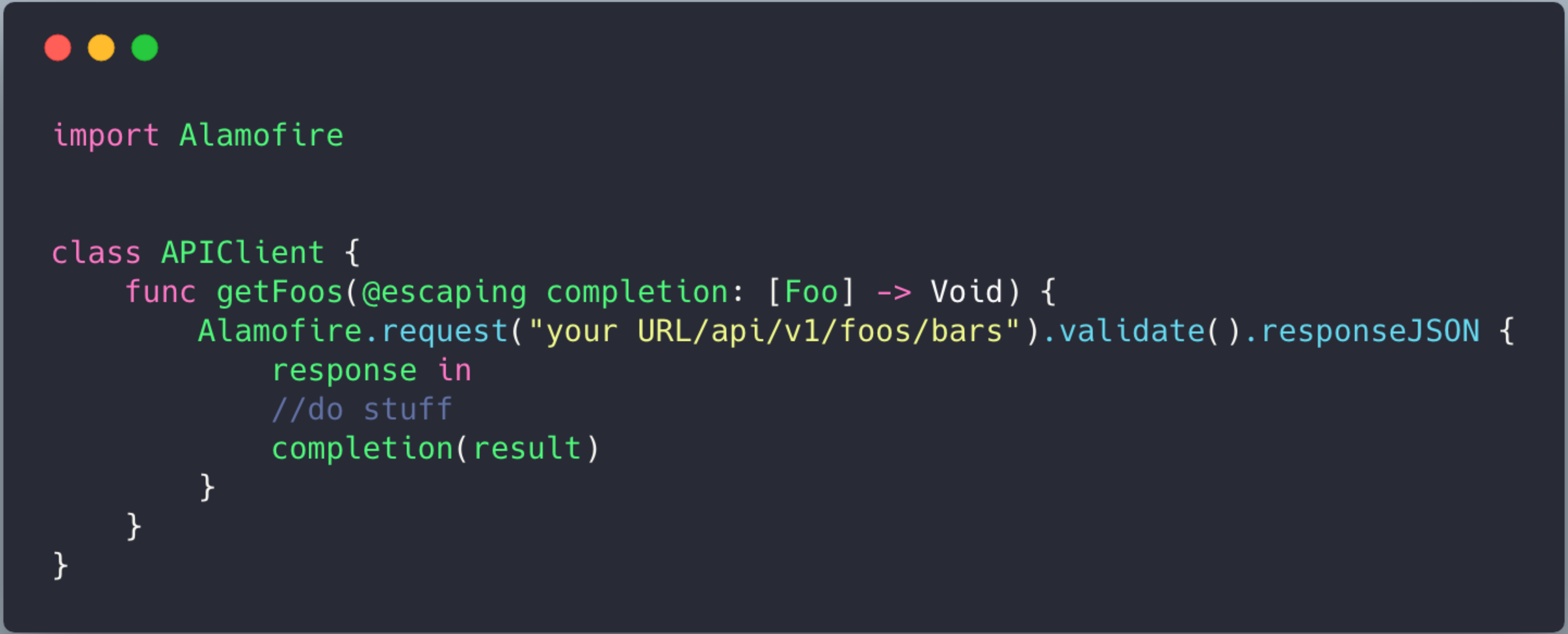
```swift
import UIKit

class AwesomeViewController: UIViewController {

    override func viewDidLoad() {
        // TODO: do cool UI
    }
}
```



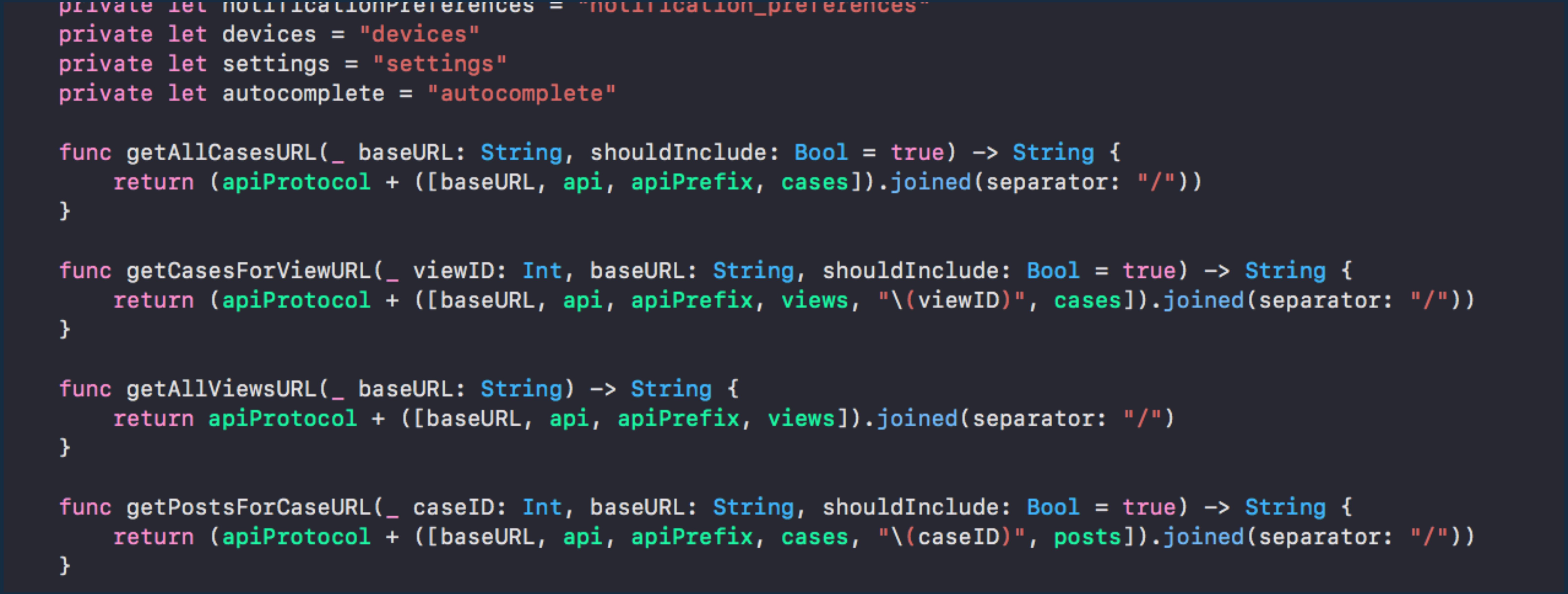
```swift
import UIKit

class AwesomeViewController: UIViewController {

    override func viewDidLoad() {
        // TODO: do cool UI

        // TODO: Networking
    }
}
```



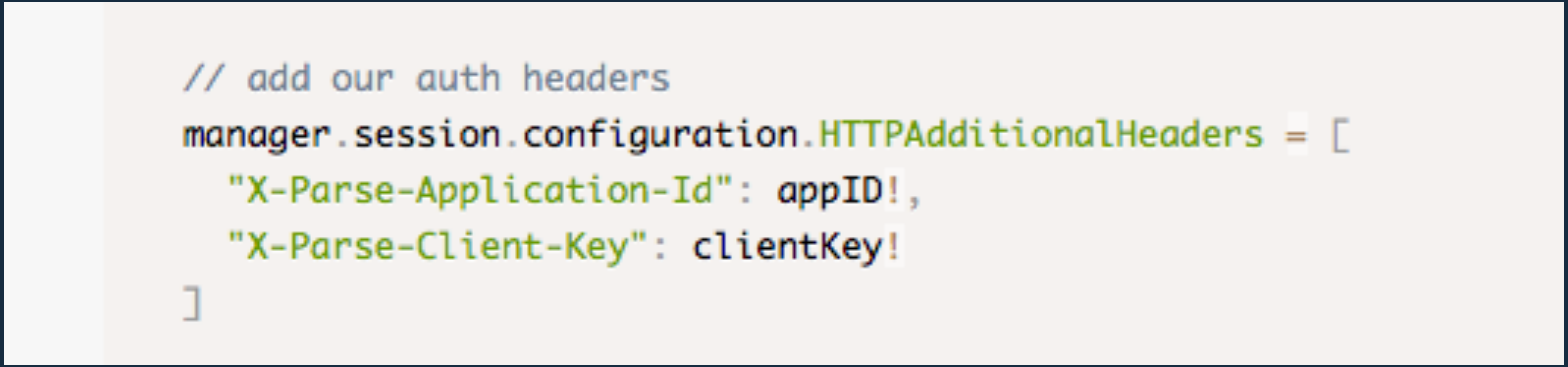
```
import UIKit

class AwesomeViewController: UIViewController {

    override func viewDidLoad() {
        // TODO: do cool UI

        Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
            response in
            //do stuff
        }
    }
}
```



```swift
import Alamofire


class APIClient {
    func getFoos(@escaping completion: [Foo] -> Void) {
        Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
            response in
            //do stuff
            completion(result)
        }
    }
}
```

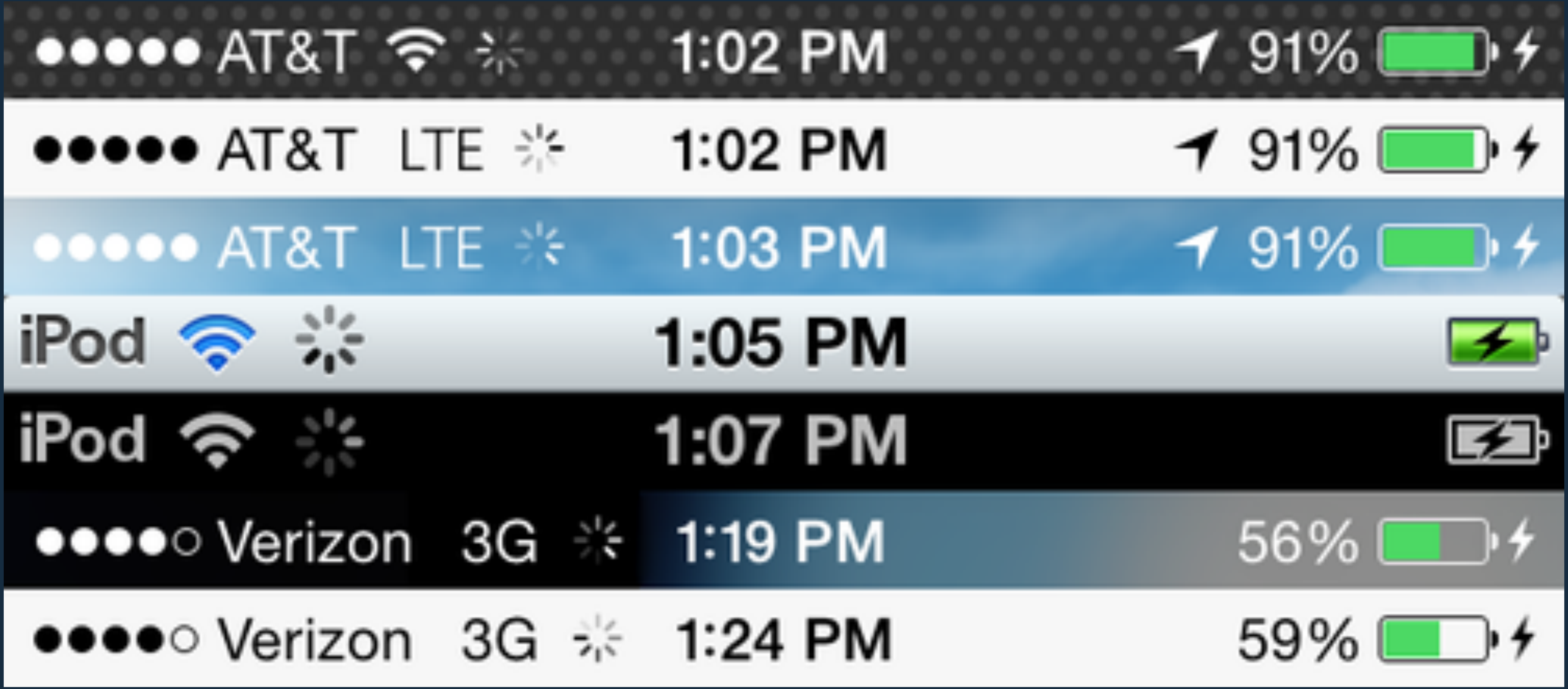


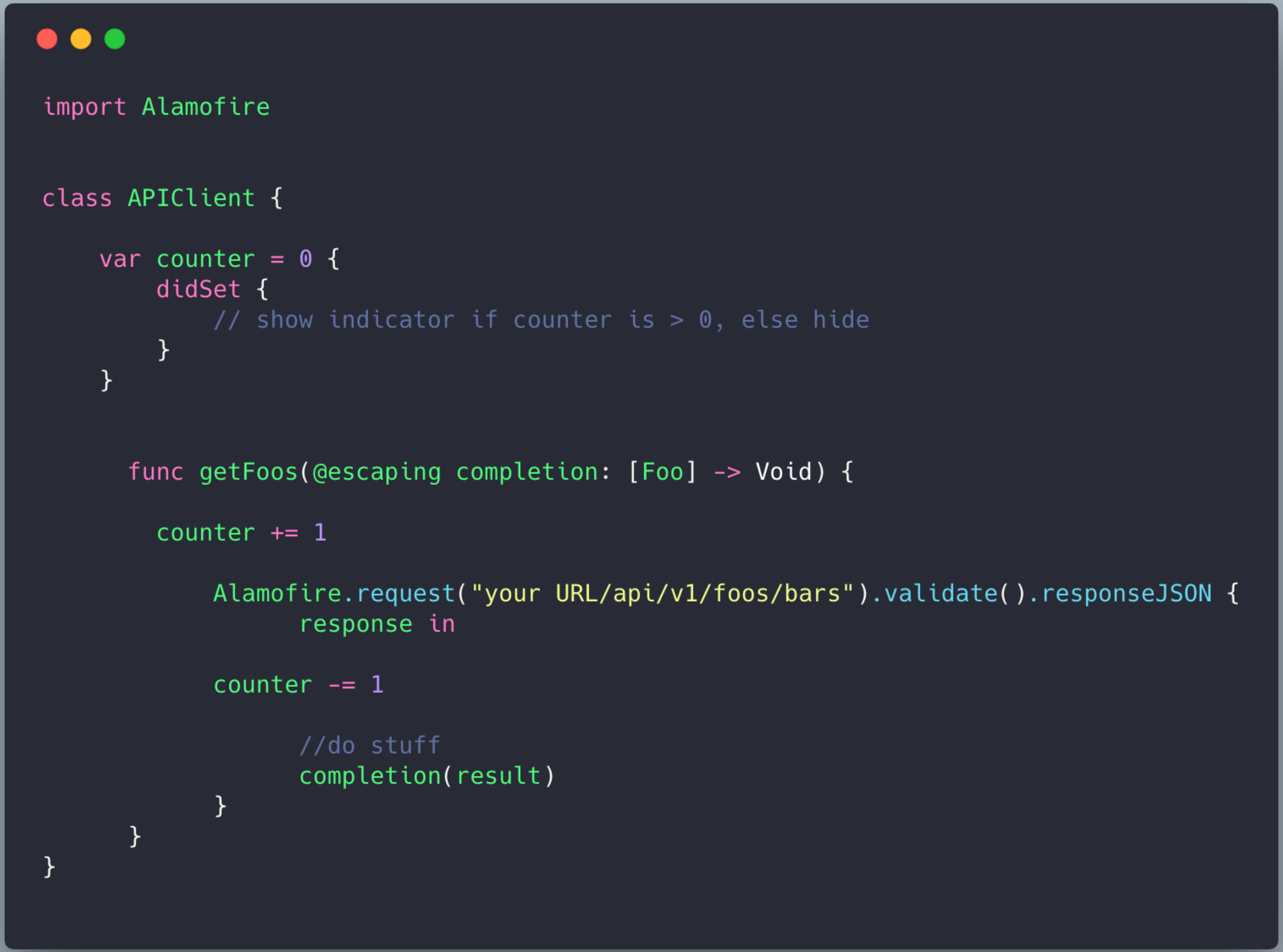
```swift
    private let notificationPreferences = "notification_preferences"
    private let devices = "devices"
    private let settings = "settings"
    private let autocomplete = "autocomplete"

    func getAllCasesURL(_ baseURL: String, shouldInclude: Bool = true) -> String {
        return (apiProtocol + ([baseURL, api, apiPrefix, cases]).joined(separator: "/"))
    }

    func getCasesForViewURL(_ viewID: Int, baseURL: String, shouldInclude: Bool = true) -> String {
        return (apiProtocol + ([baseURL, api, apiPrefix, views, "\(viewID)", cases]).joined(separator: "/"))
    }

    func getAllViewsURL(_ baseURL: String) -> String {
        return apiProtocol + ([baseURL, api, apiPrefix, views]).joined(separator: "/")
    }

    func getPostsForCaseURL(_ caseID: Int, baseURL: String, shouldInclude: Bool = true) -> String {
        return (apiProtocol + ([baseURL, api, apiPrefix, cases, "\(caseID)", posts]).joined(separator: "/"))
    }
```

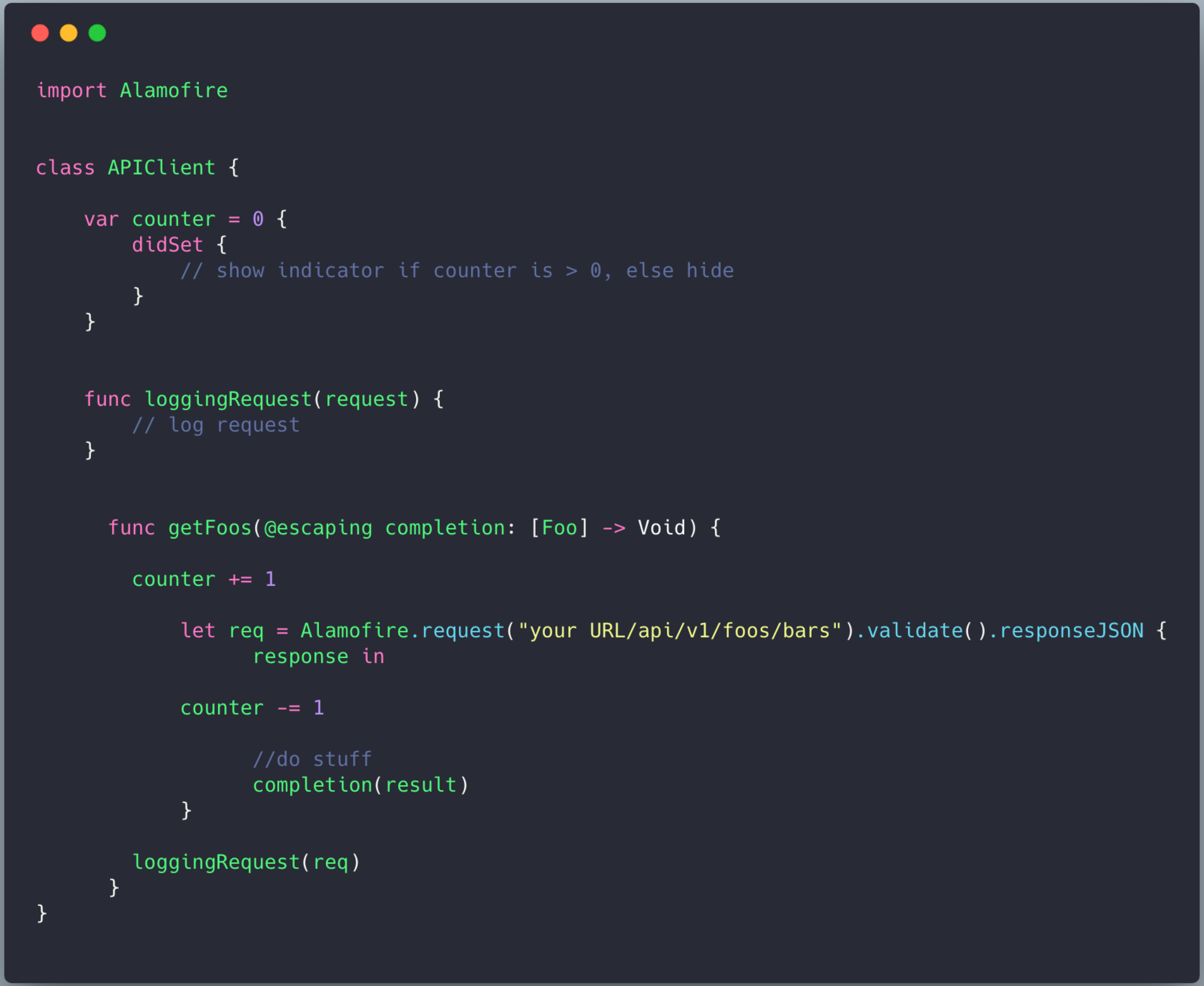


```
// add our auth headers
manager.session.configuration.HTTPAdditionalHeaders = [
  "X-Parse-Application-Id": appID!,
  "X-Parse-Client-Key": clientKey!
]
```

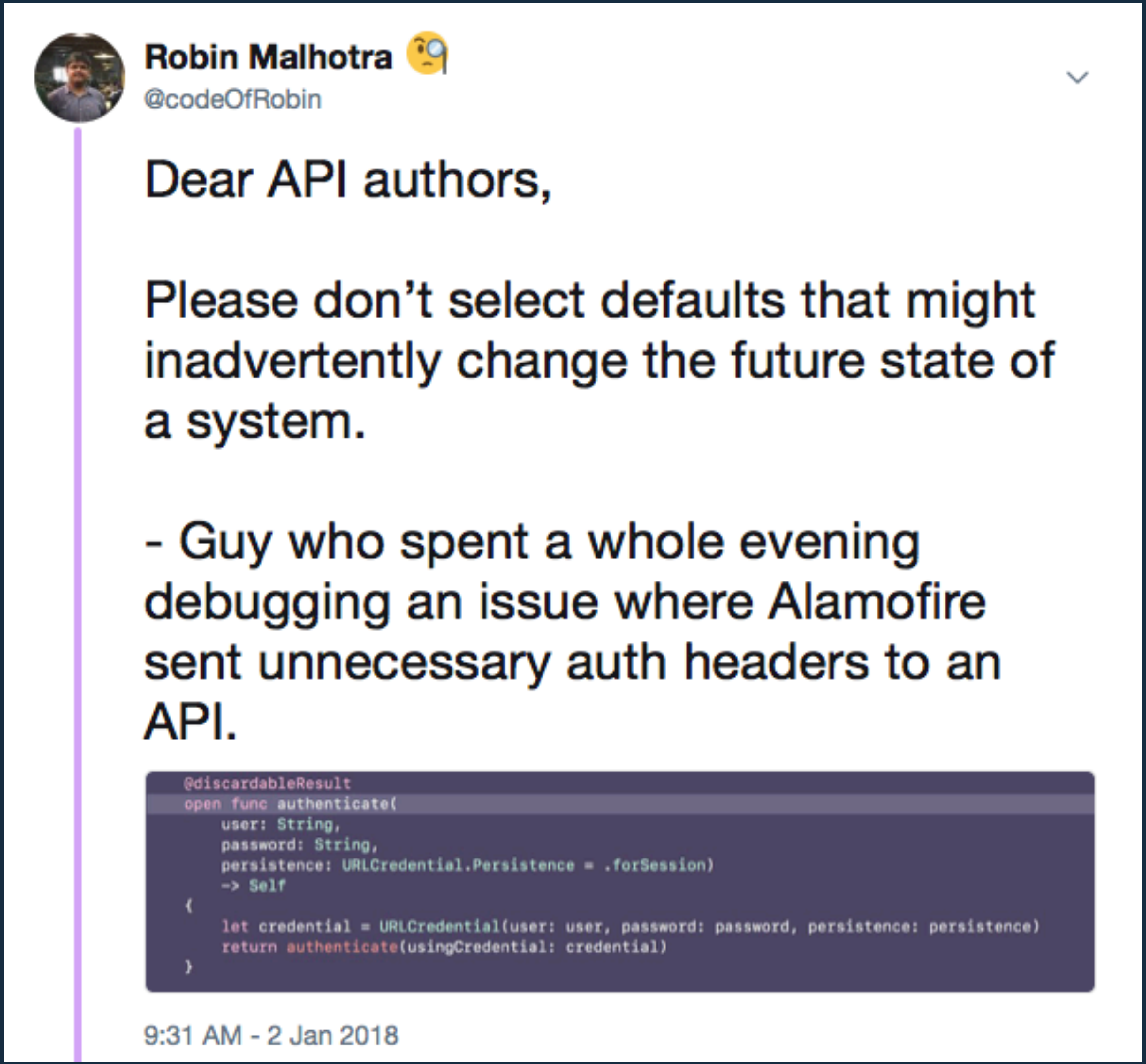
(image for representational purposes)


●●●●● AT&T 1:02 PM 91%
●●●●● AT&T LTE 1:02 PM 91%
●●●●● AT&T LTE 1:03 PM 91%
iPod 1:05 PM
iPod 1:07 PM
●●●●○ Verizon 3G 1:19 PM 56%
●●●●○ Verizon 3G 1:24 PM 59%


```
import Alamofire


class APIClient {

    var counter = 0 {
        didSet {
            // show indicator if counter is > 0, else hide
        }
    }


      func getFoos(@escaping completion: [Foo] -> Void) {

        counter += 1

            Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
                  response in

            counter -= 1

                  //do stuff
                  completion(result)
            }
      }
}
```



```
import Alamofire


class APIClient {

    var counter = 0 {
        didSet {
            // show indicator if counter is > 0, else hide
        }
    }


    func loggingRequest(request) {
        // log request
    }


      func getFoos(@escaping completion: [Foo] -> Void) {

        counter += 1

            let req = Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
                  response in

            counter -= 1

                  //do stuff
                  completion(result)
            }

        loggingRequest(req)
      }
}
```



Robin Malhotra 🧐
@codeOfRobin

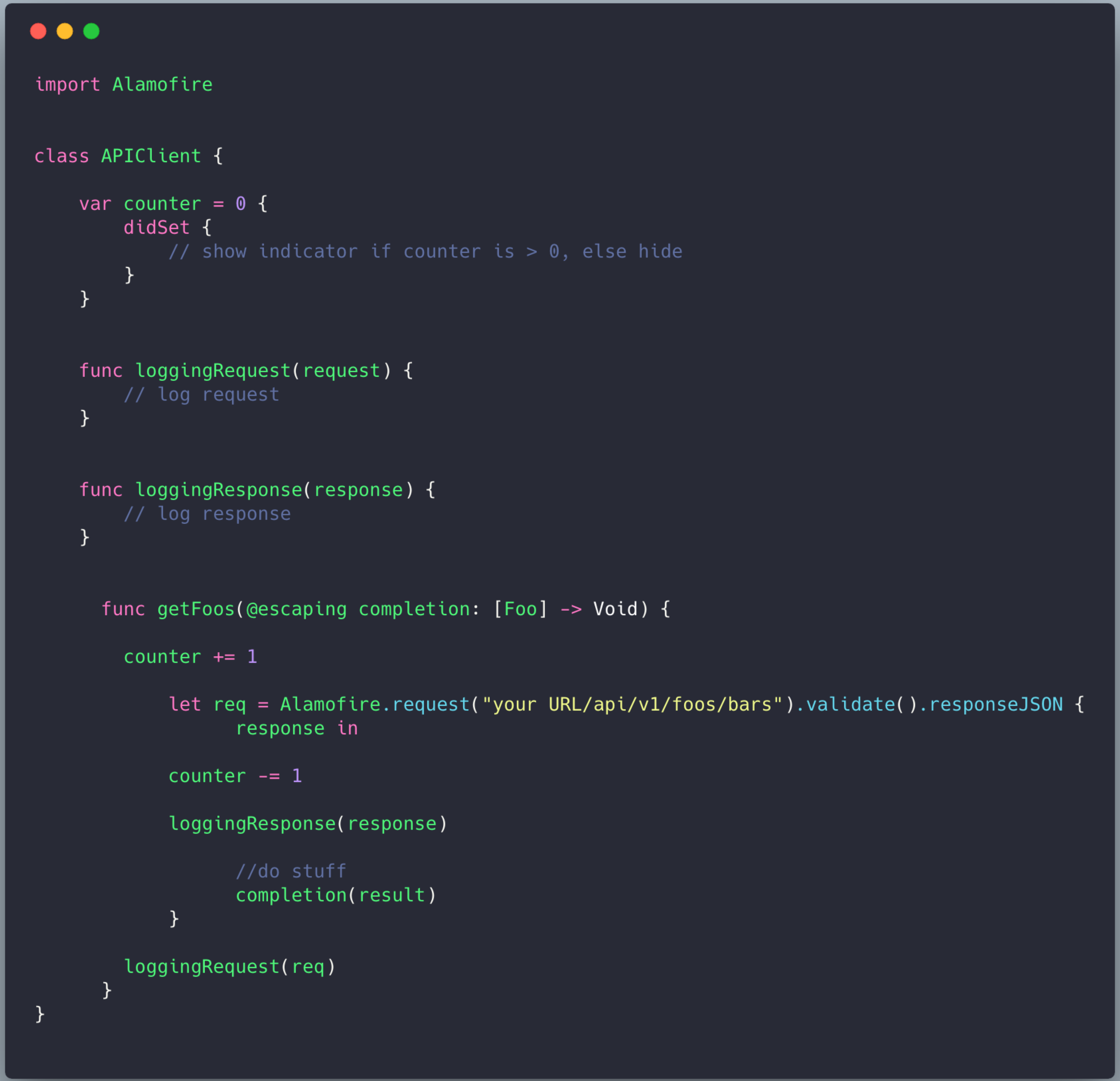
Dear API authors,

Please don’t select defaults that might inadvertently change the future state of a system.

- Guy who spent a whole evening debugging an issue where Alamofire sent unnecessary auth headers to an API.

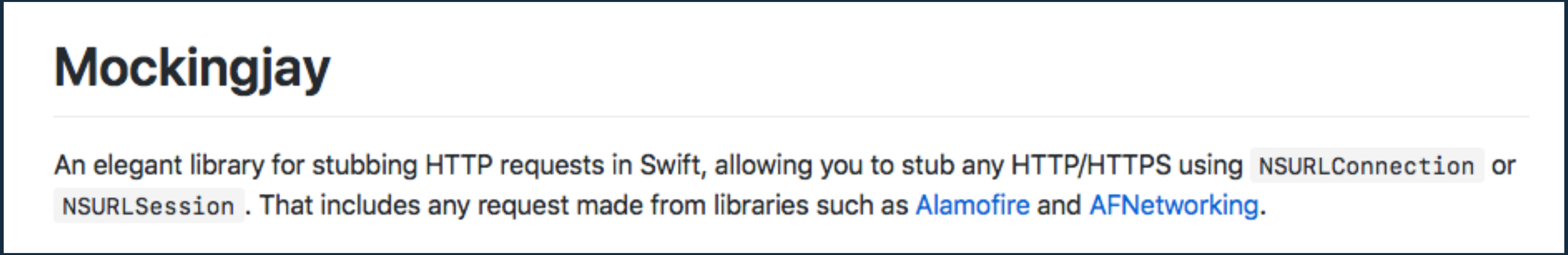
```swift
@discardableResult
open func authenticate(
    user: String,
    password: String,
    persistence: URLCredential.Persistence = .forSession)
    -> Self
{
    let credential = URLCredential(user: user, password: password, persistence: persistence)
    return authenticate(usingCredential: credential)
}
```

9:31 AM - 2 Jan 2018



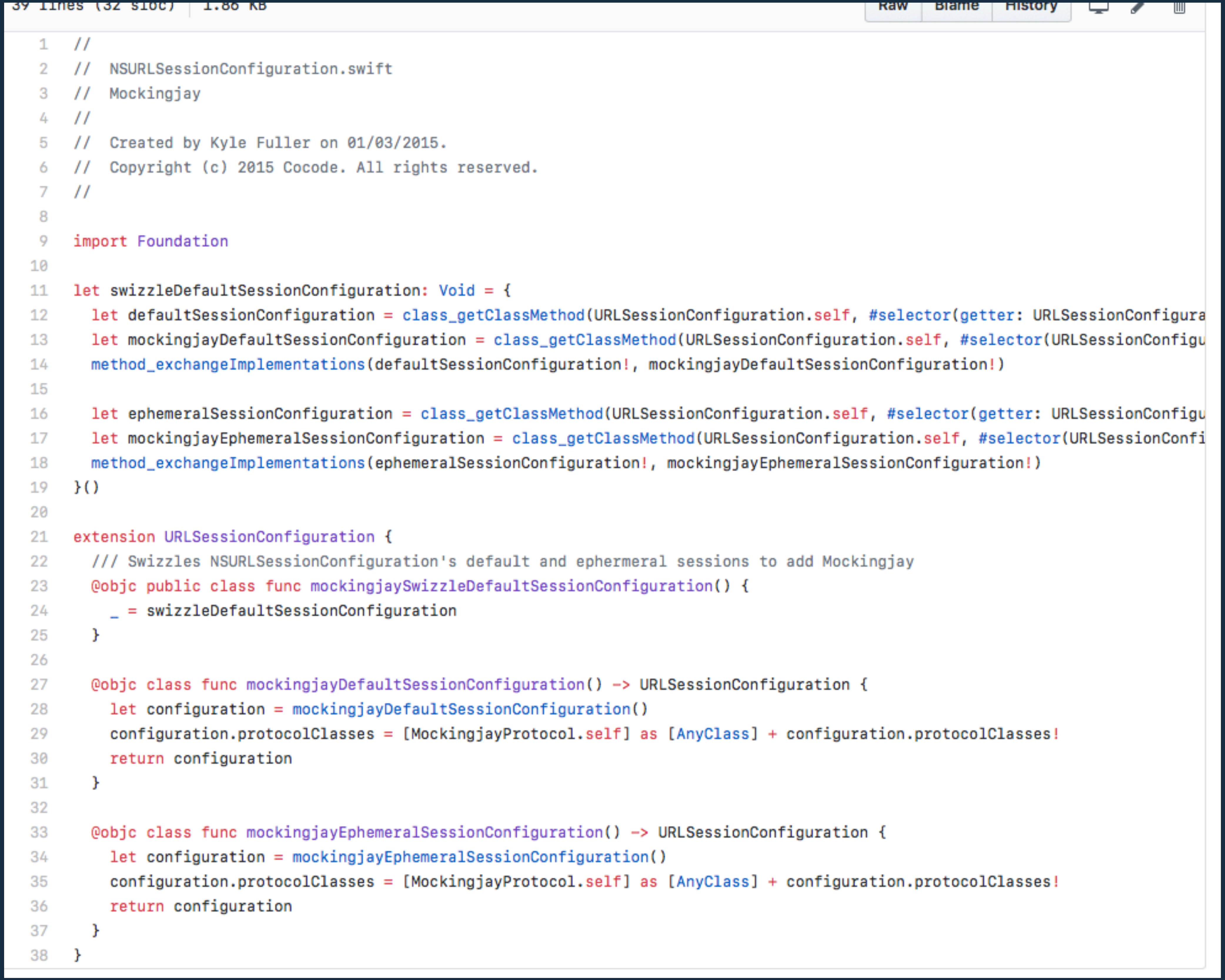
```
import Alamofire


class APIClient {

    var counter = 0 {
        didSet {
            // show indicator if counter is > 0, else hide
        }
    }


    func loggingRequest(request) {
        // log request
    }


    func loggingResponse(response) {
        // log response
    }


      func getFoos(@escaping completion: [Foo] -> Void) {

        counter += 1

            let req = Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
                  response in

            counter -= 1

            loggingResponse(response)

                  //do stuff
                  completion(result)
            }

        loggingRequest(req)
      }
}
```

TESTING

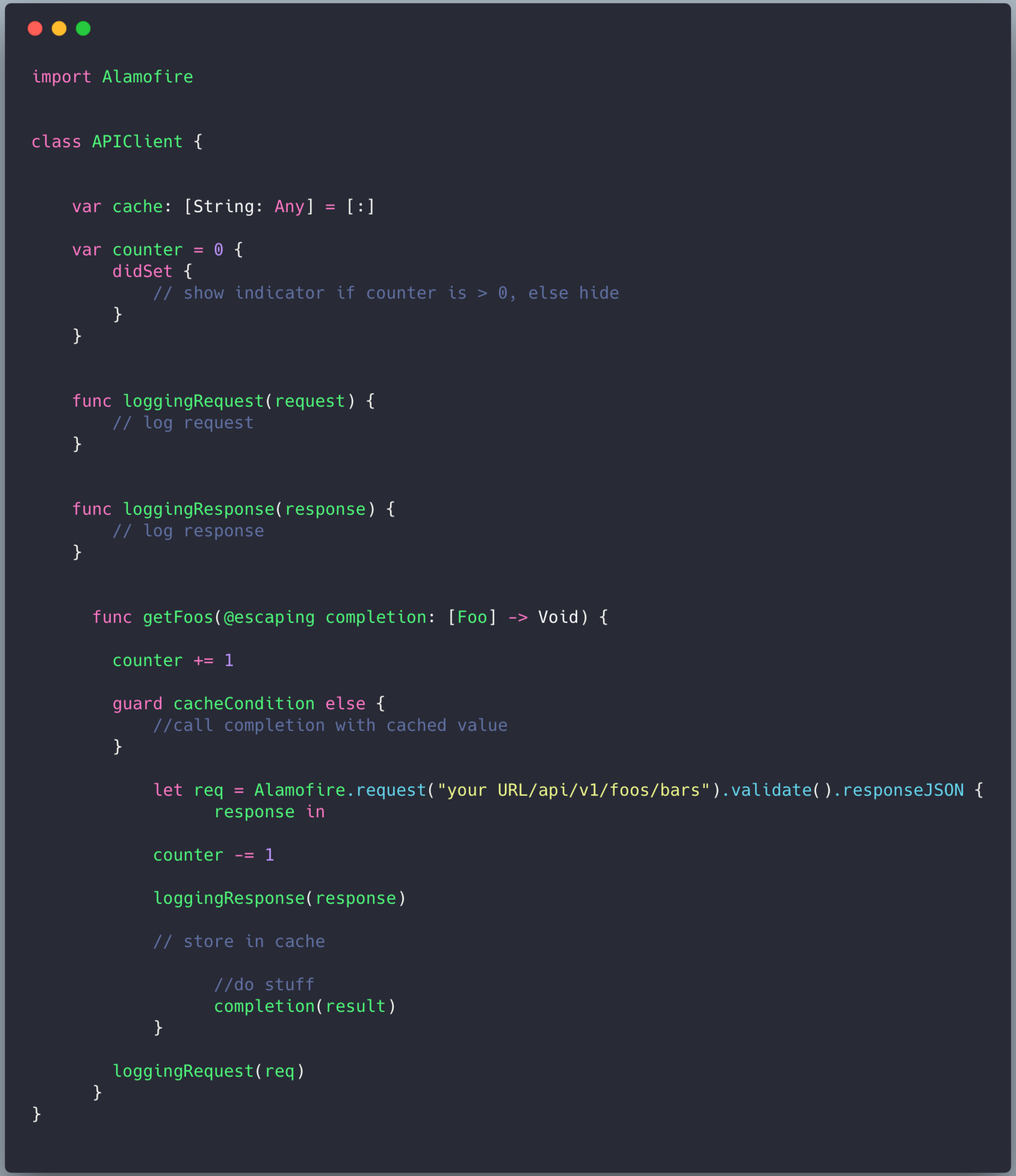


Mockingjay

An elegant library for stubbing HTTP requests in Swift, allowing you to stub any HTTP/HTTPS using `NSURLConnection` or `NSURLSession`. That includes any request made from libraries such as Alamofire and AFNetworking.



```swift
//
//  NSURLSessionConfiguration.swift
//  Mockingjay
//
//  Created by Kyle Fuller on 01/03/2015.
//  Copyright (c) 2015 Cocode. All rights reserved.
//

import Foundation

let swizzleDefaultSessionConfiguration: Void = {
  let defaultSessionConfiguration = class_getClassMethod(URLSessionConfiguration.self, #selector(getter: URLSessionConfigura
  let mockingjayDefaultSessionConfiguration = class_getClassMethod(URLSessionConfiguration.self, #selector(URLSessionConfigu
  method_exchangeImplementations(defaultSessionConfiguration!, mockingjayDefaultSessionConfiguration!)

  let ephemeralSessionConfiguration = class_getClassMethod(URLSessionConfiguration.self, #selector(getter: URLSessionConfigu
  let mockingjayEphemeralSessionConfiguration = class_getClassMethod(URLSessionConfiguration.self, #selector(URLSessionConfi
  method_exchangeImplementations(ephemeralSessionConfiguration!, mockingjayEphemeralSessionConfiguration!)
}()

extension URLSessionConfiguration {
  /// Swizzles NSURLSessionConfiguration's default and ephermeral sessions to add Mockingjay
  @objc public class func mockingjaySwizzleDefaultSessionConfiguration() {
    _ = swizzleDefaultSessionConfiguration
  }

  @objc class func mockingjayDefaultSessionConfiguration() -> URLSessionConfiguration {
    let configuration = mockingjayDefaultSessionConfiguration()
    configuration.protocolClasses = [MockingjayProtocol.self] as [AnyClass] + configuration.protocolClasses!
    return configuration
  }

  @objc class func mockingjayEphemeralSessionConfiguration() -> URLSessionConfiguration {
    let configuration = mockingjayEphemeralSessionConfiguration()
    configuration.protocolClasses = [MockingjayProtocol.self] as [AnyClass] + configuration.protocolClasses!
    return configuration
  }
}
```

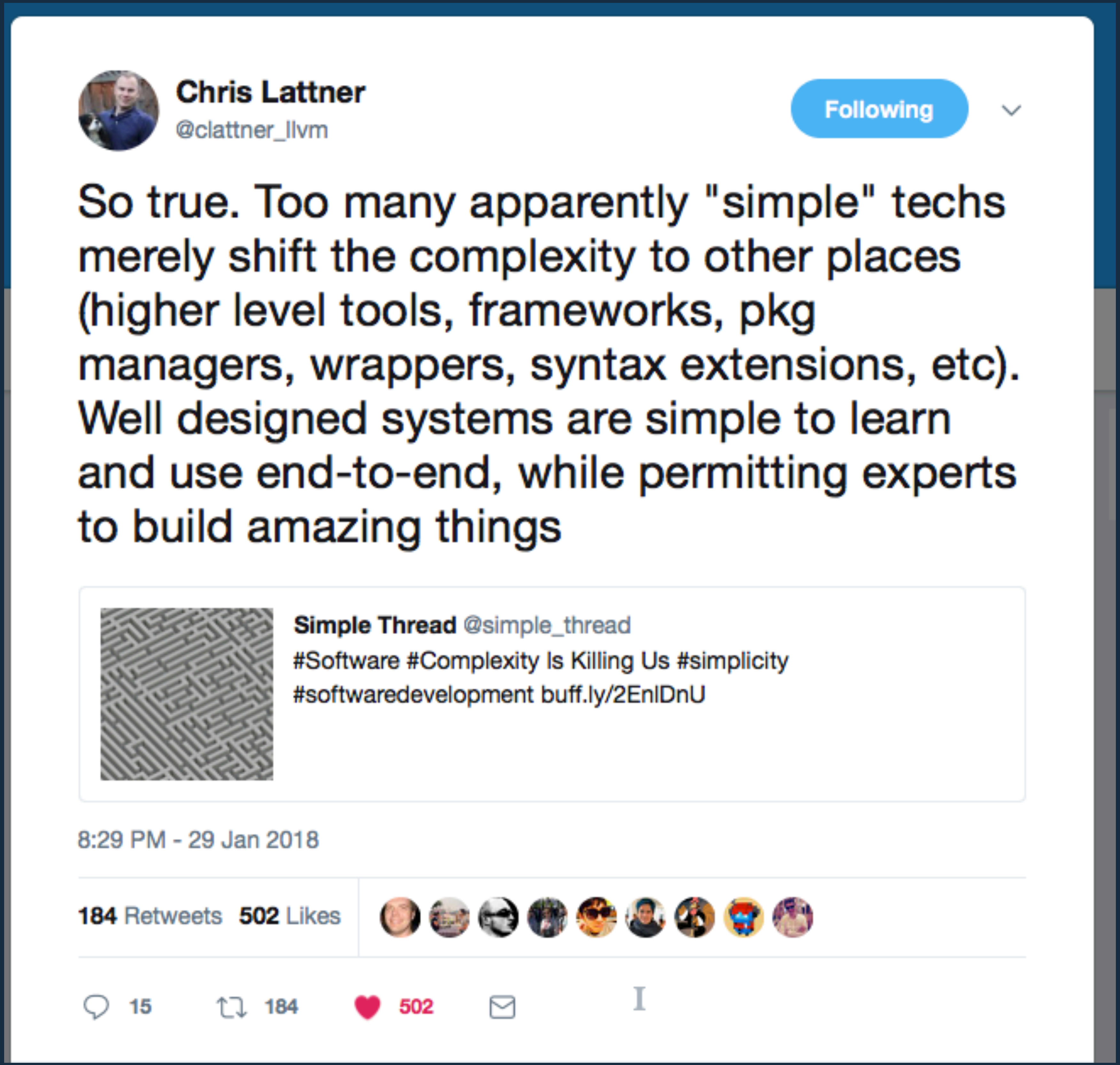

```swift
import Alamofire


class APIClient {


    var cache: [String: Any] = [:]

    var counter = 0 {
        didSet {
            // show indicator if counter is > 0, else hide
        }
    }


    func loggingRequest(request) {
        // log request
    }


    func loggingResponse(response) {
        // log response
    }


      func getFoos(@escaping completion: [Foo] -> Void) {

        counter += 1

        guard cacheCondition else {
            //call completion with cached value
        }

            let req = Alamofire.request("your URL/api/v1/foos/bars").validate().responseJSON {
                  response in

            counter -= 1

            loggingResponse(response)

            // store in cache

                  //do stuff
                  completion(result)
            }

        loggingRequest(req)
      }
}
```

🤕

Tests on Linux?

I'M DONE

It's not really architecture, it's about decisions and complexity



Chris Lattner
@clattner_llvm

Following

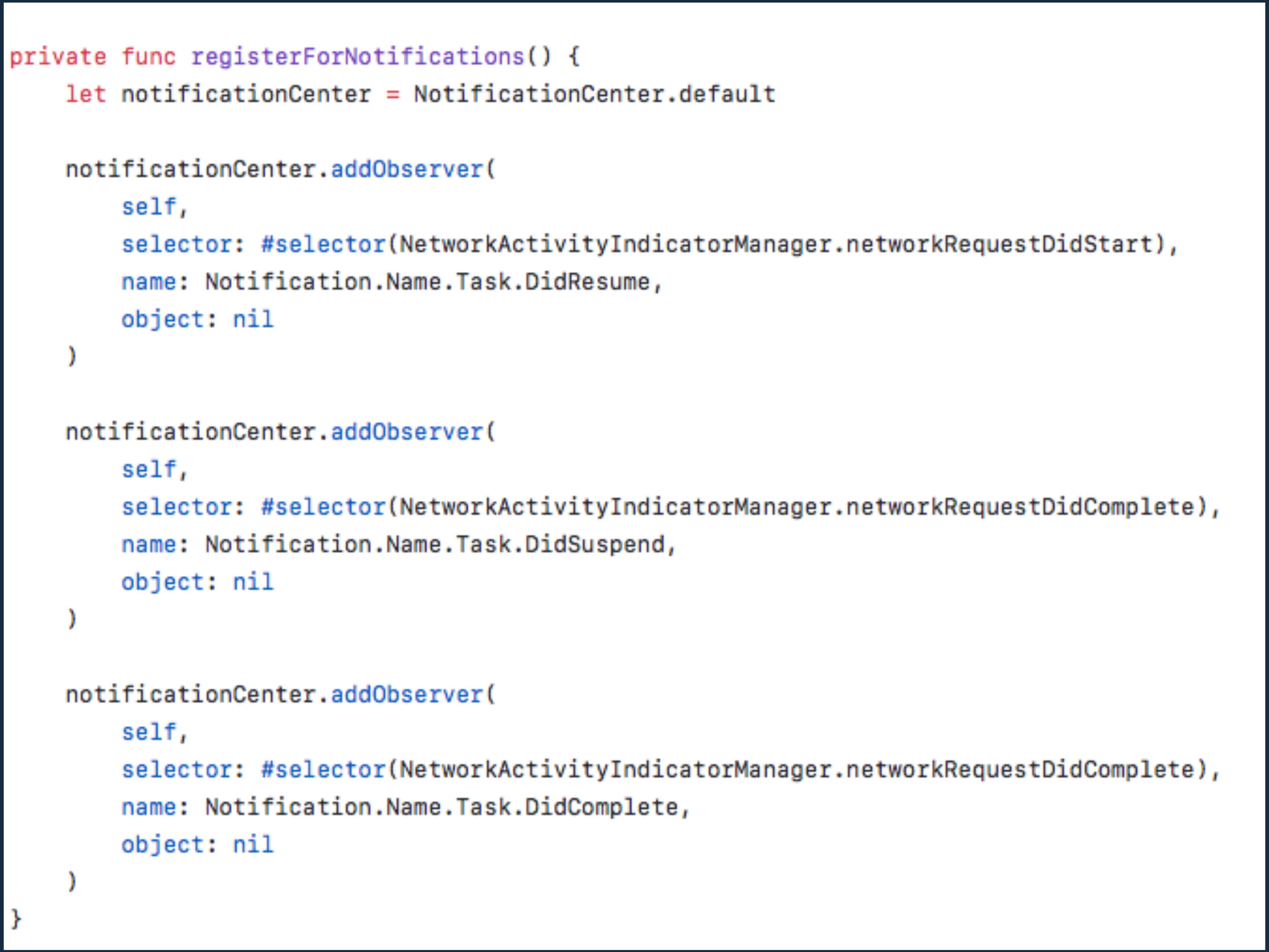
So true. Too many apparently "simple" techs merely shift the complexity to other places (higher level tools, frameworks, pkg managers, wrappers, syntax extensions, etc). Well designed systems are simple to learn and use end-to-end, while permitting experts to build amazing things

Simple Thread @simple_thread
#Software #Complexity Is Killing Us #simplicity #softwaredevelopment buff.ly/2EnlDnU

8:29 PM - 29 Jan 2018

184 Retweets 502 Likes

15 184 502

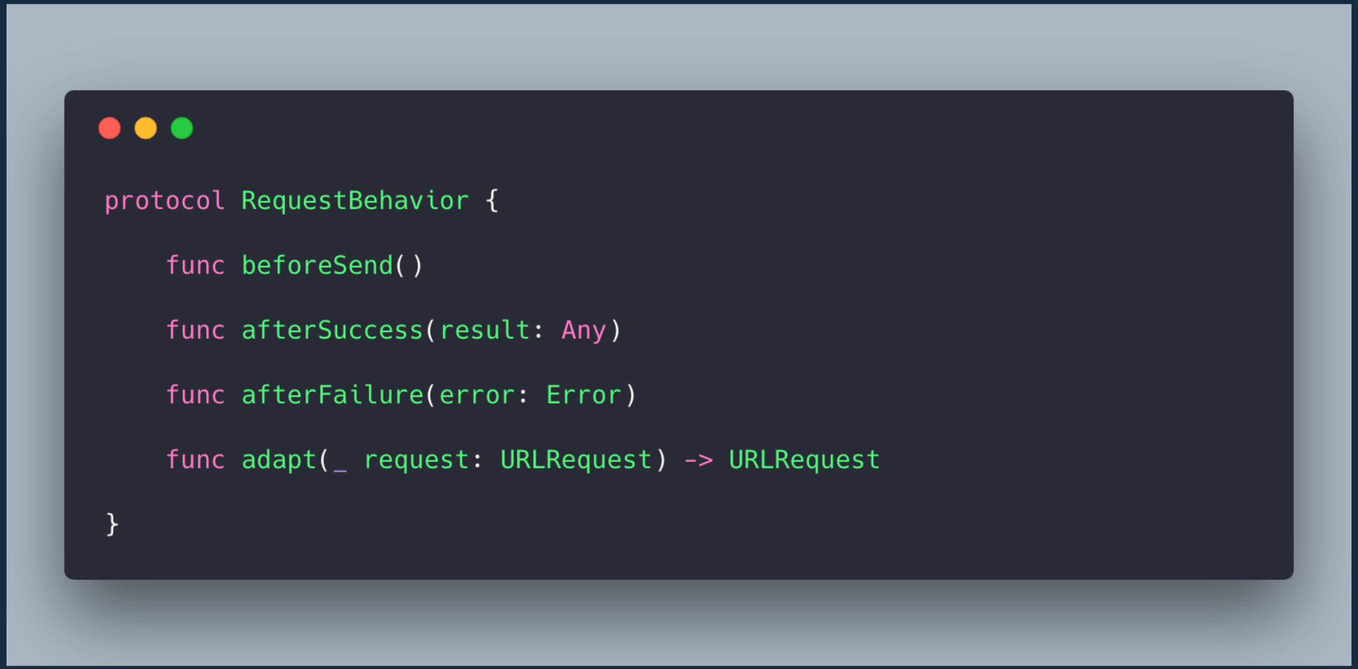


```swift
private func registerForNotifications() {
    let notificationCenter = NotificationCenter.default

    notificationCenter.addObserver(
        self,
        selector: #selector(NetworkActivityIndicatorManager.networkRequestDidStart),
        name: Notification.Name.Task.DidResume,
        object: nil
    )

    notificationCenter.addObserver(
        self,
        selector: #selector(NetworkActivityIndicatorManager.networkRequestDidComplete),
        name: Notification.Name.Task.DidSuspend,
        object: nil
    )

    notificationCenter.addObserver(
        self,
        selector: #selector(NetworkActivityIndicatorManager.networkRequestDidComplete),
        name: Notification.Name.Task.DidComplete,
        object: nil
    )
}
```

A Better Way™

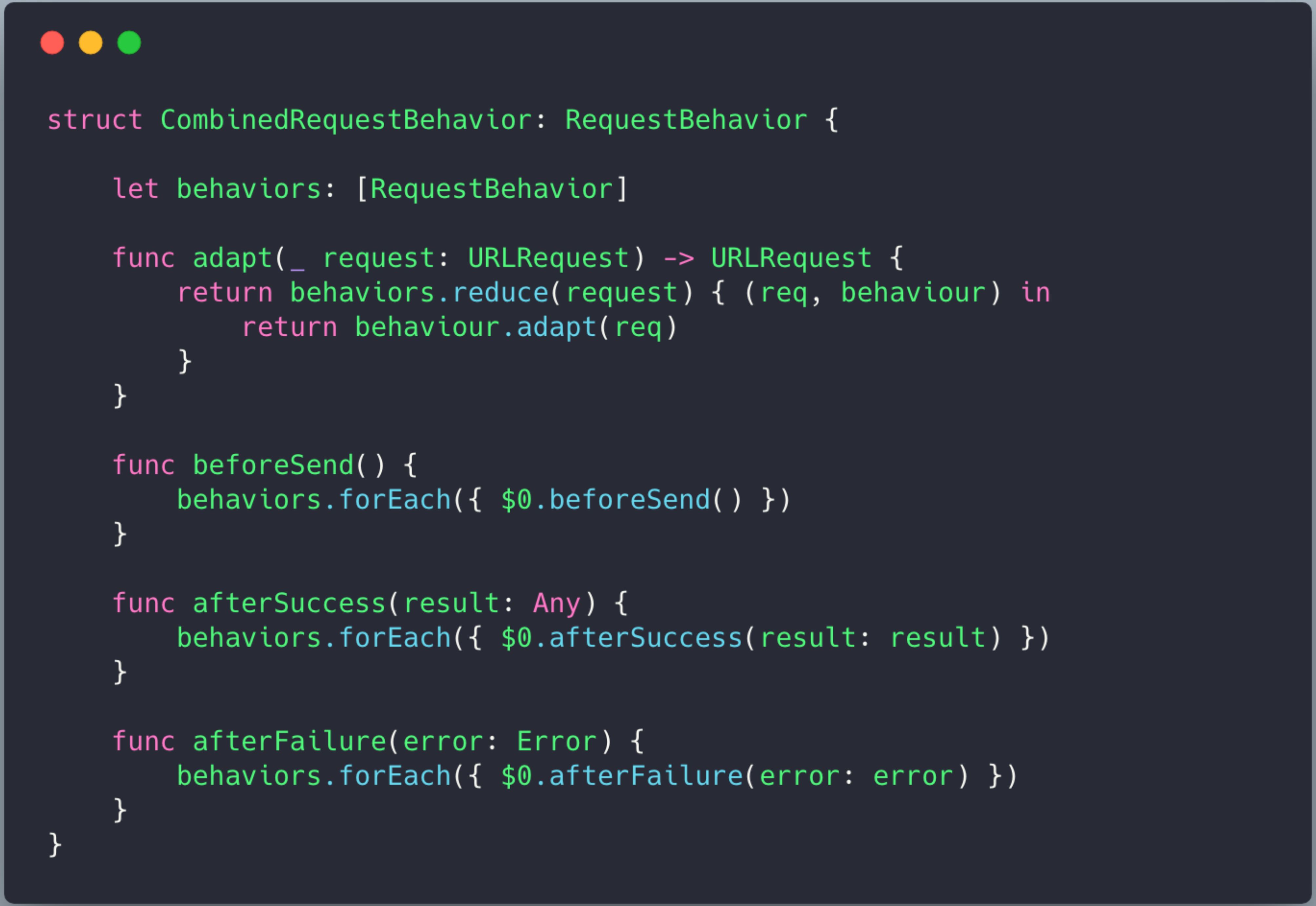
Request Behaviours

- Discovered this on Souroush Khanlou's blog[1]
- Make a protocol that wraps these underlying behaviours

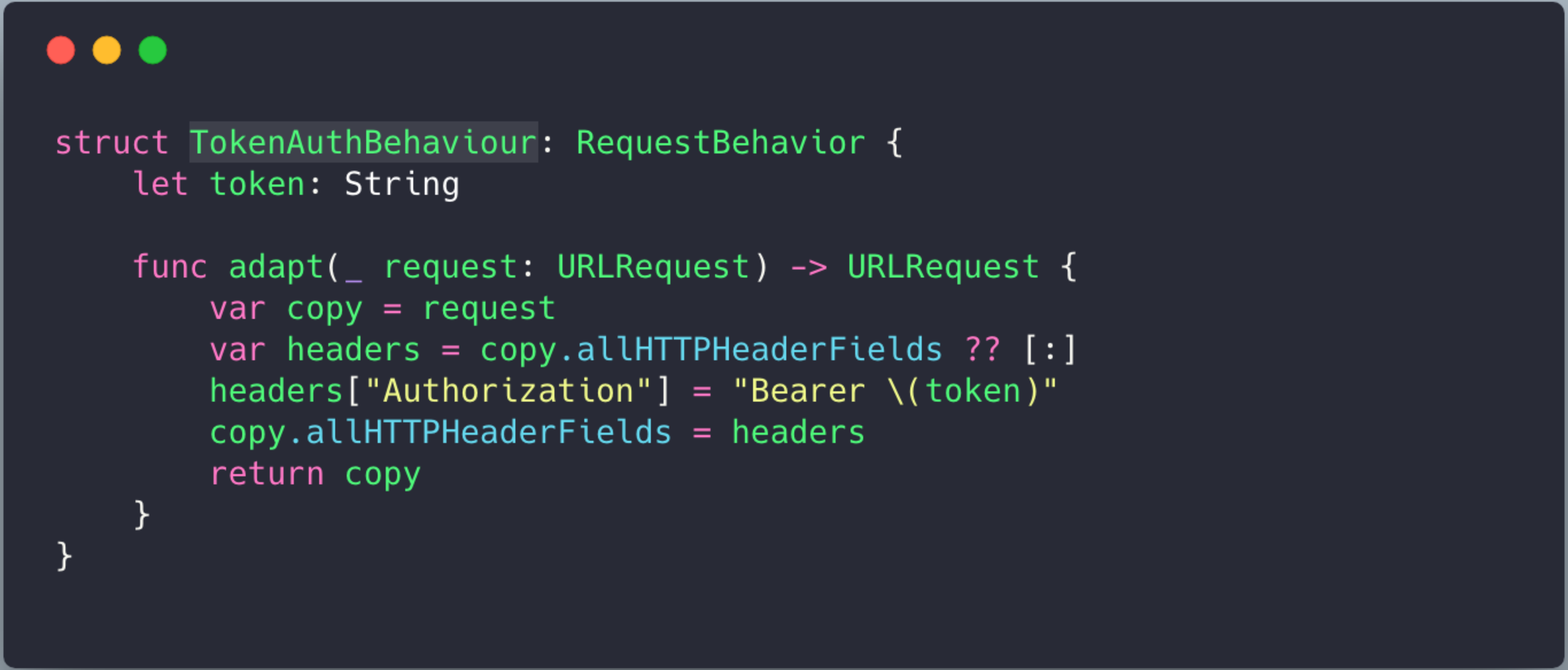


```swift
protocol RequestBehavior {
    func beforeSend()
    func afterSuccess(result: Any)
    func afterFailure(error: Error)
    func adapt(_ request: URLRequest) -> URLRequest
}
```

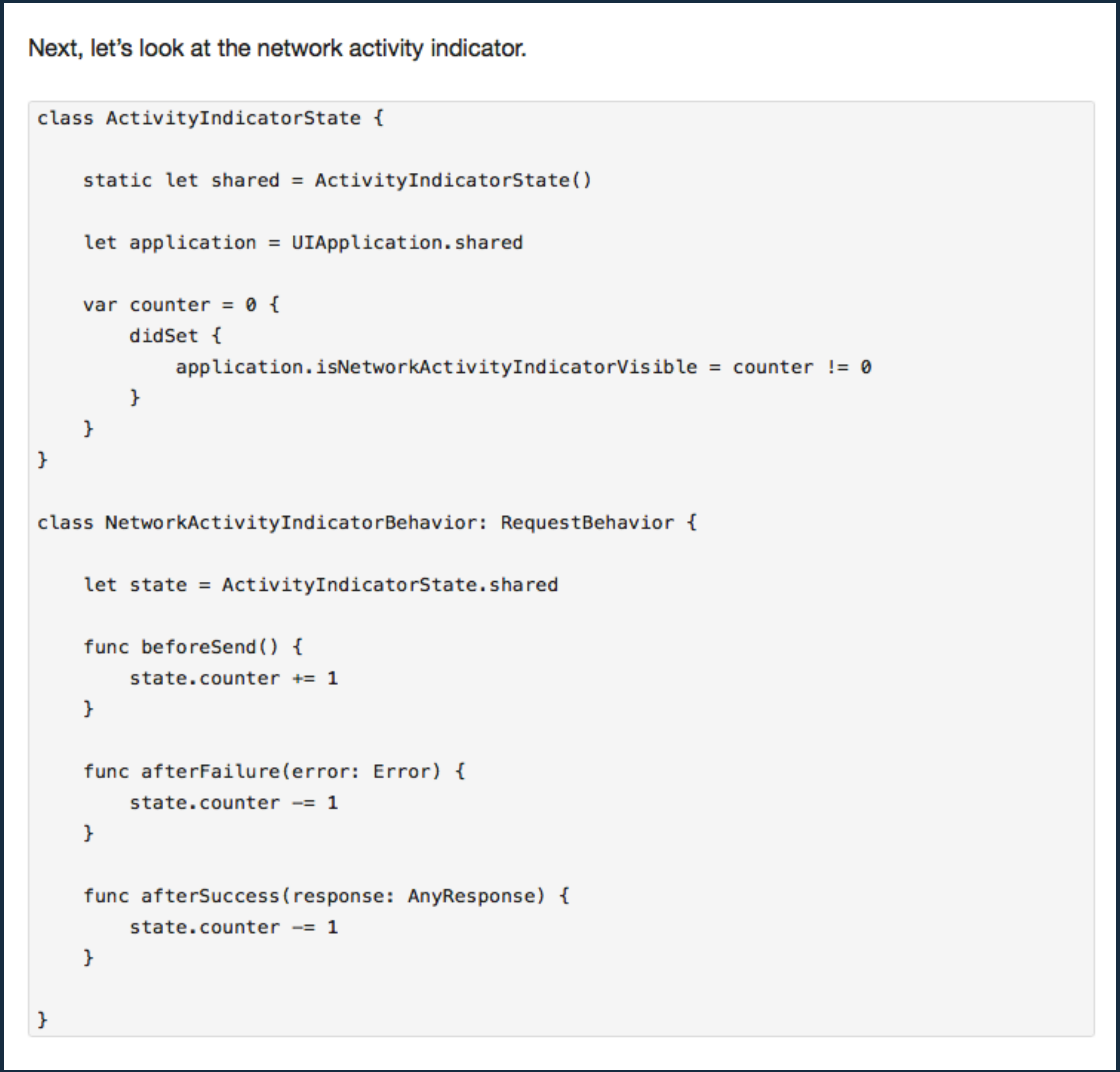
[1] http://khanlou.com/2017/01/request-behaviors/



```swift
struct CombinedRequestBehavior: RequestBehavior {

    let behaviors: [RequestBehavior]

    func adapt(_ request: URLRequest) -> URLRequest {
        return behaviors.reduce(request) { (req, behaviour) in
            return behaviour.adapt(req)
        }
    }

    func beforeSend() {
        behaviors.forEach({ $0.beforeSend() })
    }

    func afterSuccess(result: Any) {
        behaviors.forEach({ $0.afterSuccess(result: result) })
    }

    func afterFailure(error: Error) {
        behaviors.forEach({ $0.afterFailure(error: error) })
    }
}
```

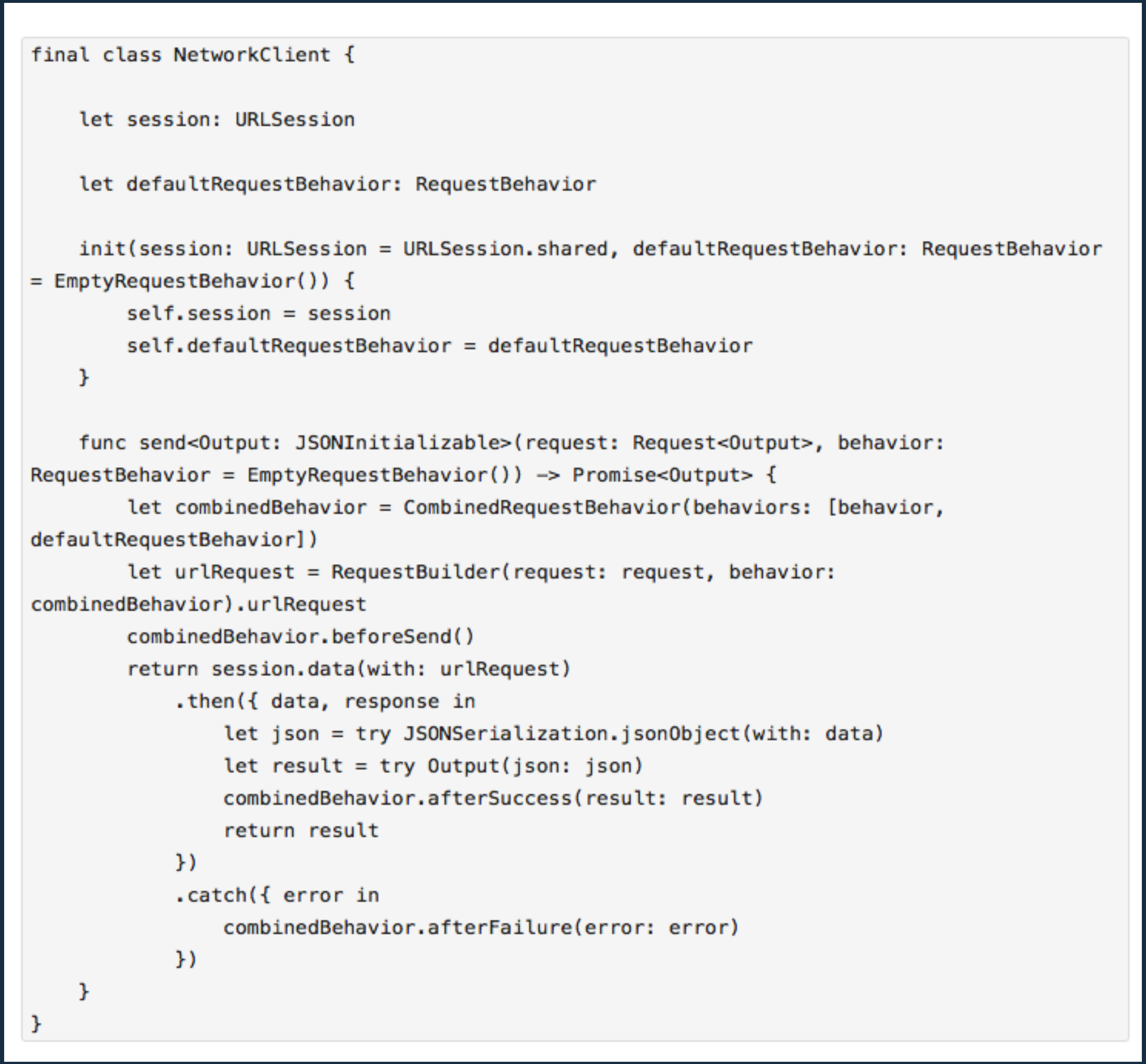


```swift
struct TokenAuthBehaviour: RequestBehavior {
    let token: String

    func adapt(_ request: URLRequest) -> URLRequest {
        var copy = request
        var headers = copy.allHTTPHeaderFields ?? [:]
        headers["Authorization"] = "Bearer \(token)"
        copy.allHTTPHeaderFields = headers
        return copy
    }
}
```

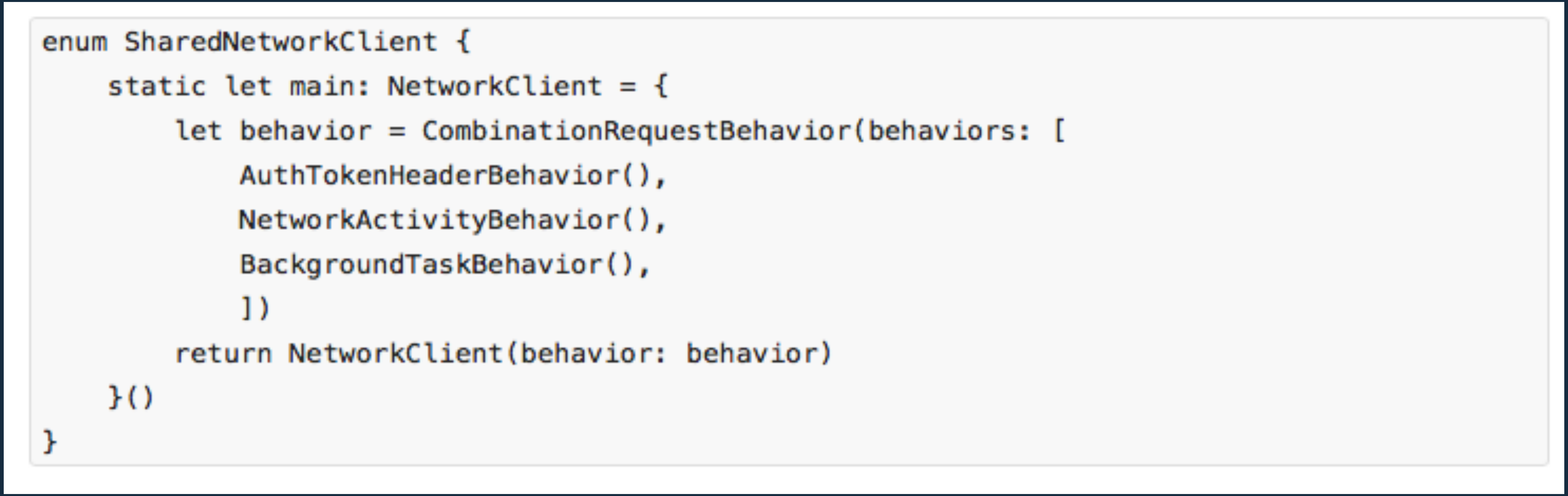


Next, let's look at the network activity indicator.

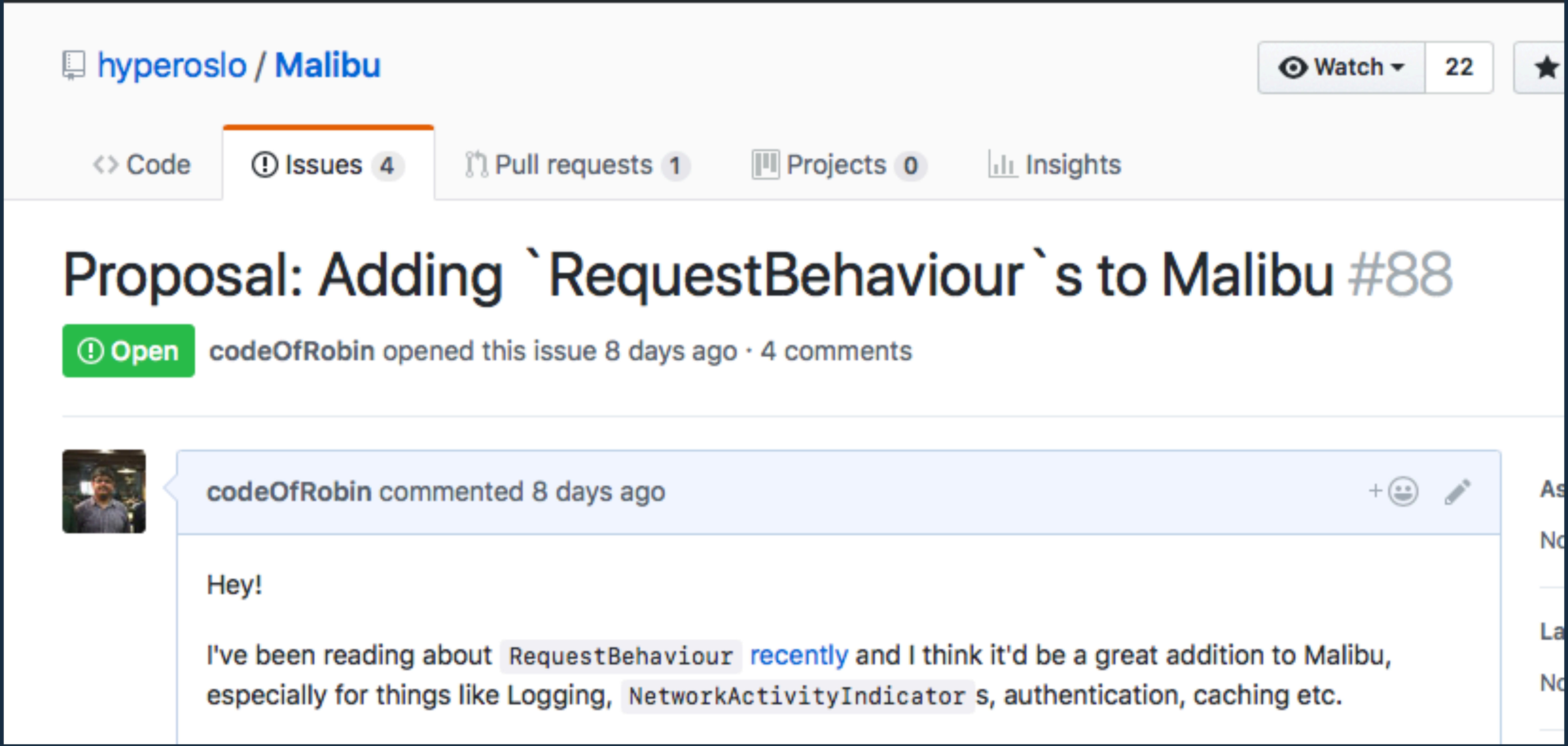
```
class ActivityIndicatorState {

    static let shared = ActivityIndicatorState()

    let application = UIApplication.shared

    var counter = 0 {
        didSet {
            application.isNetworkActivityIndicatorVisible = counter != 0
        }
    }
}

class NetworkActivityIndicatorBehavior: RequestBehavior {

    let state = ActivityIndicatorState.shared

    func beforeSend() {
        state.counter += 1
    }

    func afterFailure(error: Error) {
        state.counter -= 1
    }

    func afterSuccess(response: AnyResponse) {
        state.counter -= 1
    }

}
```



```
final class NetworkClient {

    let session: URLSession

    let defaultRequestBehavior: RequestBehavior

    init(session: URLSession = URLSession.shared, defaultRequestBehavior: RequestBehavior
= EmptyRequestBehavior()) {
        self.session = session
        self.defaultRequestBehavior = defaultRequestBehavior
    }

    func send<Output: JSONInitializable>(request: Request<Output>, behavior:
RequestBehavior = EmptyRequestBehavior()) -> Promise<Output> {
        let combinedBehavior = CombinedRequestBehavior(behaviors: [behavior,
defaultRequestBehavior])
        let urlRequest = RequestBuilder(request: request, behavior:
combinedBehavior).urlRequest
        combinedBehavior.beforeSend()
        return session.data(with: urlRequest)
            .then({ data, response in
                let json = try JSONSerialization.jsonObject(with: data)
                let result = try Output(json: json)
                combinedBehavior.afterSuccess(result: result)
                return result
            })
            .catch({ error in
                combinedBehavior.afterFailure(error: error)
            })
    }
}
```



```swift
enum SharedNetworkClient {
    static let main: NetworkClient = {
        let behavior = CombinationRequestBehavior(behaviors: [
            AuthTokenHeaderBehavior(),
            NetworkActivityBehavior(),
            BackgroundTaskBehavior(),
            ])
        return NetworkClient(behavior: behavior)
    }()
}
```


hyperoslo / Malibu

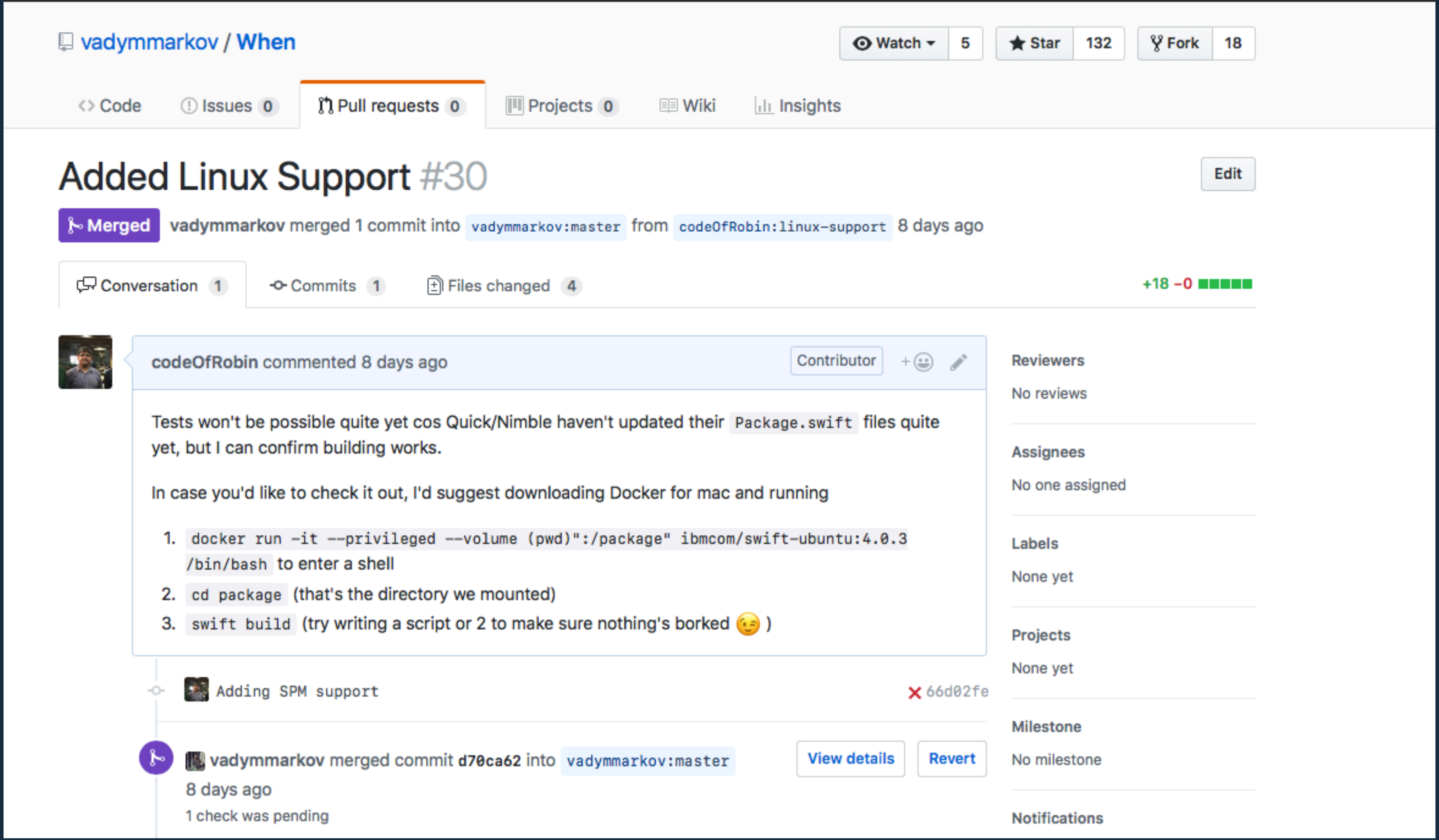
Proposal: Adding `RequestBehaviour`s to Malibu #88

Open codeOfRobin opened this issue 8 days ago · 4 comments

codeOfRobin commented 8 days ago

Hey!

I've been reading about `RequestBehaviour` recently and I think it'd be a great addition to Malibu, especially for things like Logging, `NetworkActivityIndicator`s, authentication, caching etc.



Added Linux Support #30

Merged vadymmarkov merged 1 commit into `vadymmarkov:master` from `codeOfRobin:linux-support` 8 days ago

codeOfRobin commented 8 days ago

Tests won't be possible quite yet cos Quick/Nimble haven't updated their `Package.swift` files quite yet, but I can confirm building works.

In case you'd like to check it out, I'd suggest downloading Docker for mac and running

1. `docker run -it --privileged --volume (pwd)":/package" ibmcom/swift-ubuntu:4.0.3 /bin/bash` to enter a shell
2. `cd package` (that's the directory we mounted)
3. `swift build` (try writing a script or 2 to make sure nothing's borked 😉)

`Adding SPM support` ✗ 66d02fe

vadymmarkov merged commit **d70ca62** into `vadymmarkov:master` 8 days ago

1 check was pending

Reviewers
No reviews

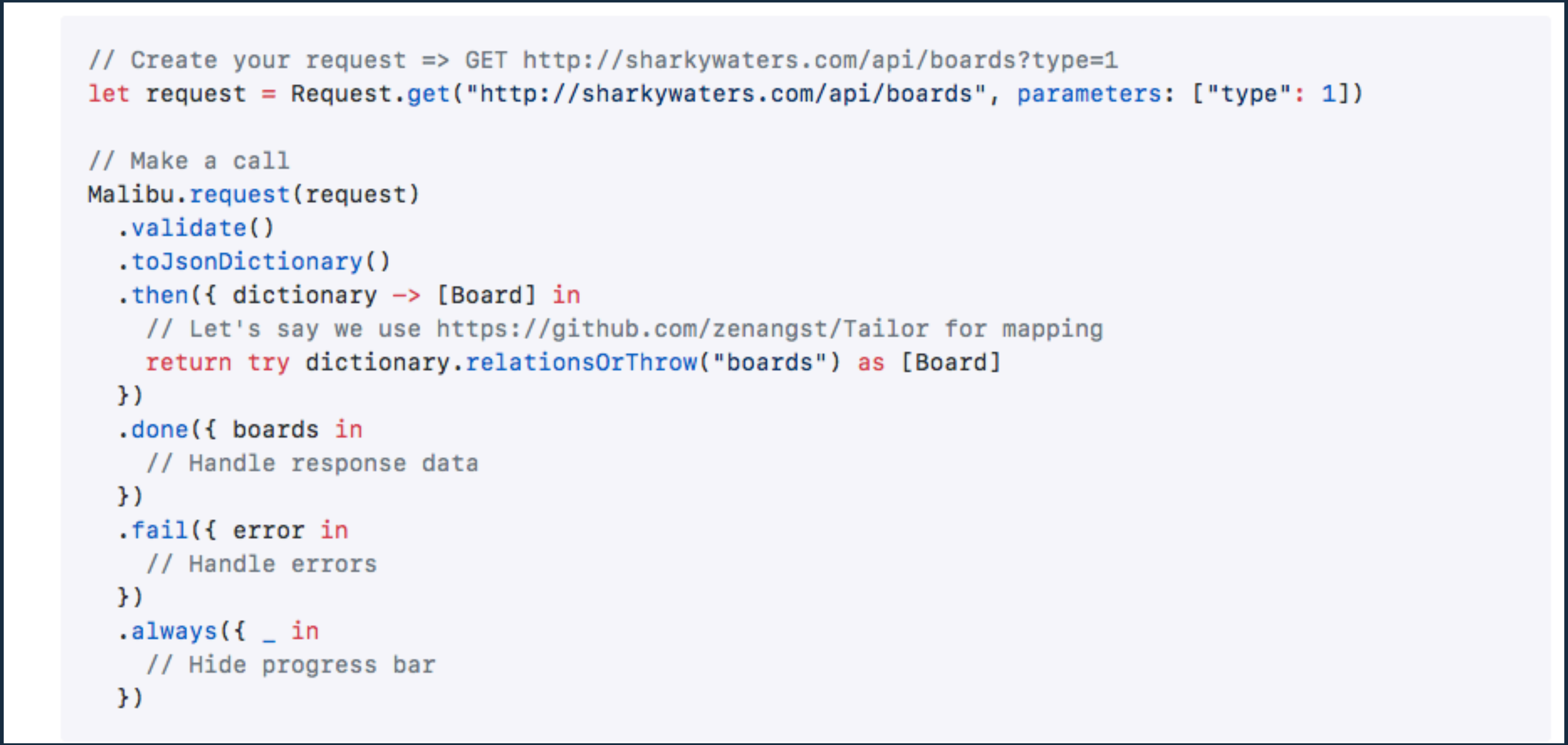
Assignees
No one assigned

Labels
None yet

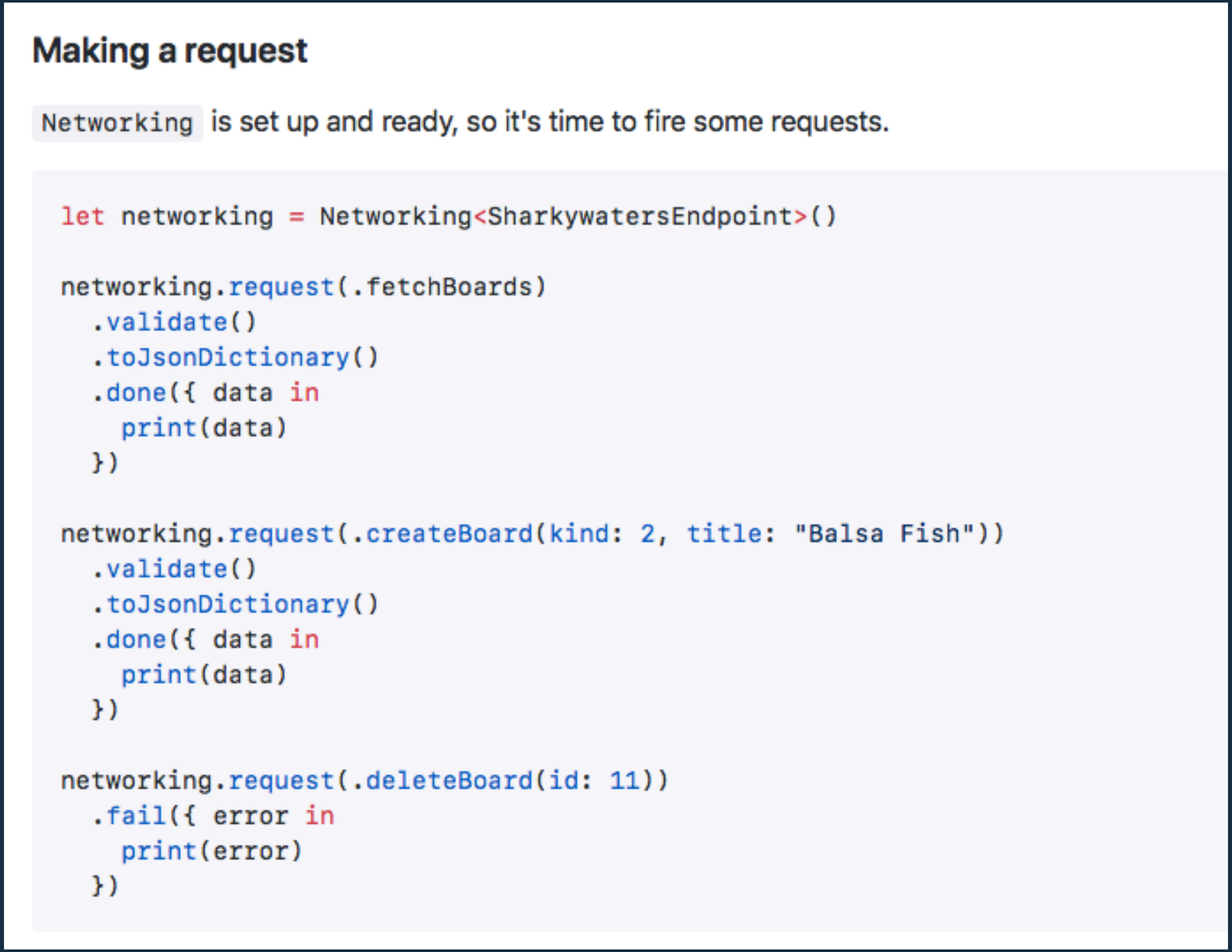
Projects
None yet

Milestone
No milestone

Notifications



```swift
// Create your request => GET http://sharkywaters.com/api/boards?type=1
let request = Request.get("http://sharkywaters.com/api/boards", parameters: ["type": 1])

// Make a call
Malibu.request(request)
  .validate()
  .toJsonDictionary()
  .then({ dictionary -> [Board] in
    // Let's say we use https://github.com/zenangst/Tailor for mapping
    return try dictionary.relationsOrThrow("boards") as [Board]
  })
  .done({ boards in
    // Handle response data
  })
  .fail({ error in
    // Handle errors
  })
  .always({ _ in
    // Hide progress bar
  })
```

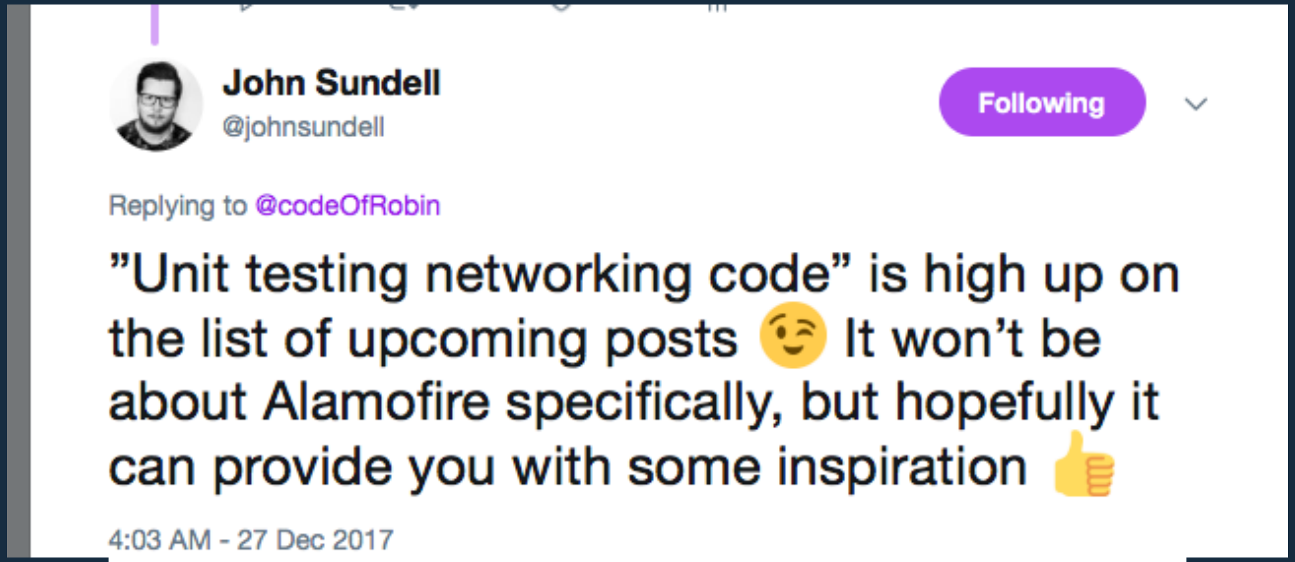


Making a request

`Networking` is set up and ready, so it's time to fire some requests.

```swift
let networking = Networking<SharkywatersEndpoint>()

networking.request(.fetchBoards)
  .validate()
  .toJsonDictionary()
  .done({ data in
    print(data)
  })

networking.request(.createBoard(kind: 2, title: "Balsa Fish"))
  .validate()
  .toJsonDictionary()
  .done({ data in
    print(data)
  })

networking.request(.deleteBoard(id: 11))
  .fail({ error in
    print(error)
  })
```

IF YOU'RE STILL NOT CONVINCED

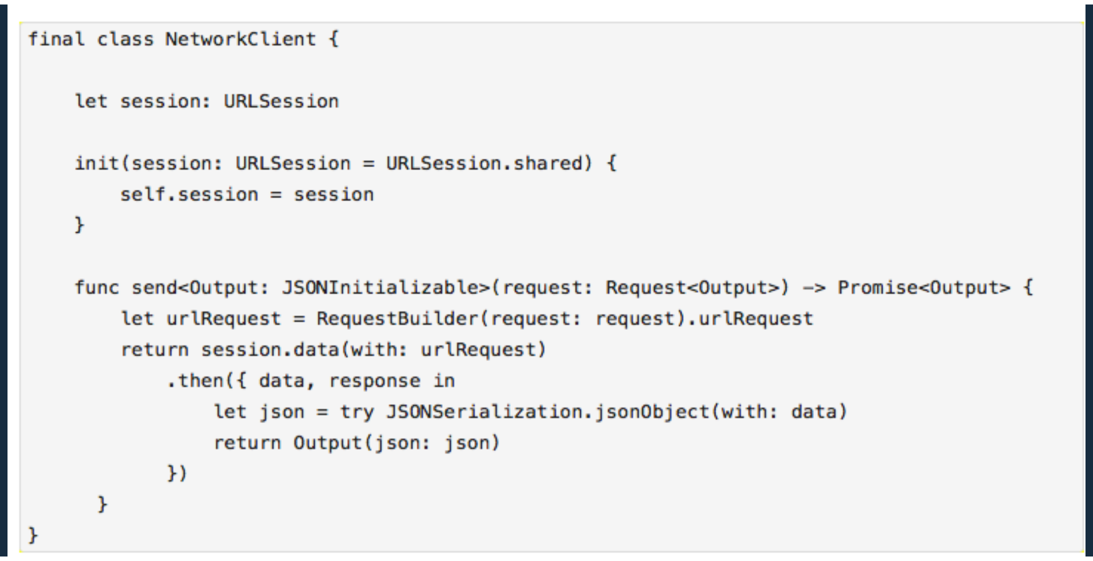


John Sundell
@johnsundell

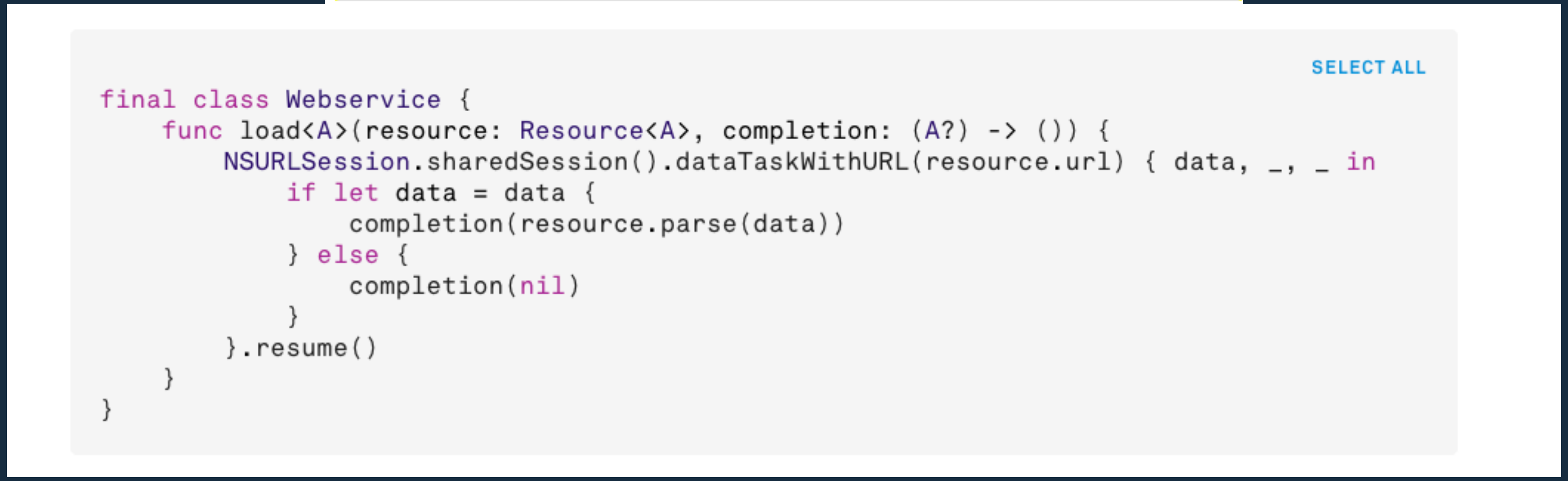
Replying to @codeOfRobin

”Unit testing networking code” is high up on the list of upcoming posts 😉 It won’t be about Alamofire specifically, but hopefully it can provide you with some inspiration 👍

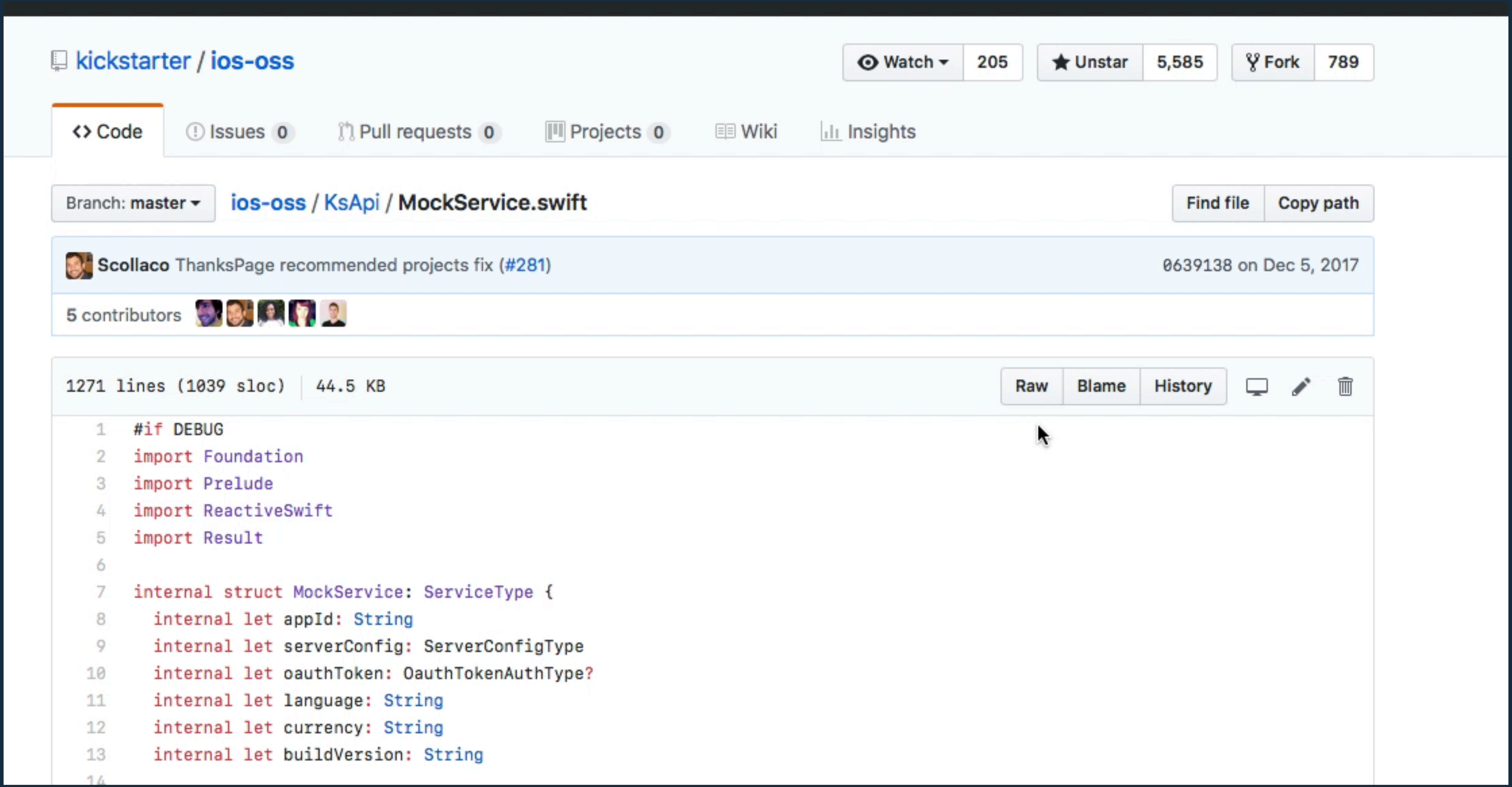
4:03 AM - 27 Dec 2017



```swift
final class NetworkClient {

    let session: URLSession

    init(session: URLSession = URLSession.shared) {
        self.session = session
    }

    func send<Output: JSONInitializable>(request: Request<Output>) -> Promise<Output> {
        let urlRequest = RequestBuilder(request: request).urlRequest
        return session.data(with: urlRequest)
            .then({ data, response in
                let json = try JSONSerialization.jsonObject(with: data)
                return Output(json: json)
            })
      }
}
```



```swift
final class Webservice {
    func load<A>(resource: Resource<A>, completion: (A?) -> ()) {
        NSURLSession.sharedSession().dataTaskWithURL(resource.url) { data, _, _ in
            if let data = data {
                completion(resource.parse(data))
            } else {
                completion(nil)
            }
        }.resume()
    }
}
```



ios-oss / KsApi / MockService.swift

Scollaco ThanksPage recommended projects fix (#281) 0639138 on Dec 5, 2017

5 contributors

1271 lines (1039 sloc) | 44.5 KB

```
#if DEBUG
import Foundation
import Prelude
import ReactiveSwift
import Result

internal struct MockService: ServiceType {
  internal let appId: String
  internal let serverConfig: ServerConfigType
  internal let oauthToken: OauthTokenAuthType?
  internal let language: String
  internal let currency: String
  internal let buildVersion: String
```

SINGLETON BASED ARCHITECTURE 🤮

THANKS FOLKS!

@codeOfRobin