Introduction to SwiftUI APIs

Girish Lukka

Topics

- API
- **Fetching Data**
- Completion handler
- Demos

API

- Definition – Application Programming Interface
- Application programming interfaces, or APIs, simplify software development and innovation by enabling applications to exchange data and functionality easily and securely.
- An API is a set of defined rules that explain how computers or applications communicate with one another. APIs sit between an application and the web server, acting as an intermediary layer that processes data transfer between systems.

API – mechanics IBM

- **A client application initiates an API call** to retrieve information—also known as a *request*. This request is processed from an application to the web server via the API's Uniform Resource Identifier (URI) and includes a request verb, headers, and sometimes, a request body.
- **After receiving a valid request**, the API makes a call to the external program or web server.
- **The server sends a *response*** to the API with the requested information.
- **The API transfers the data** to the initial requesting application.

Fetching Data sample apis

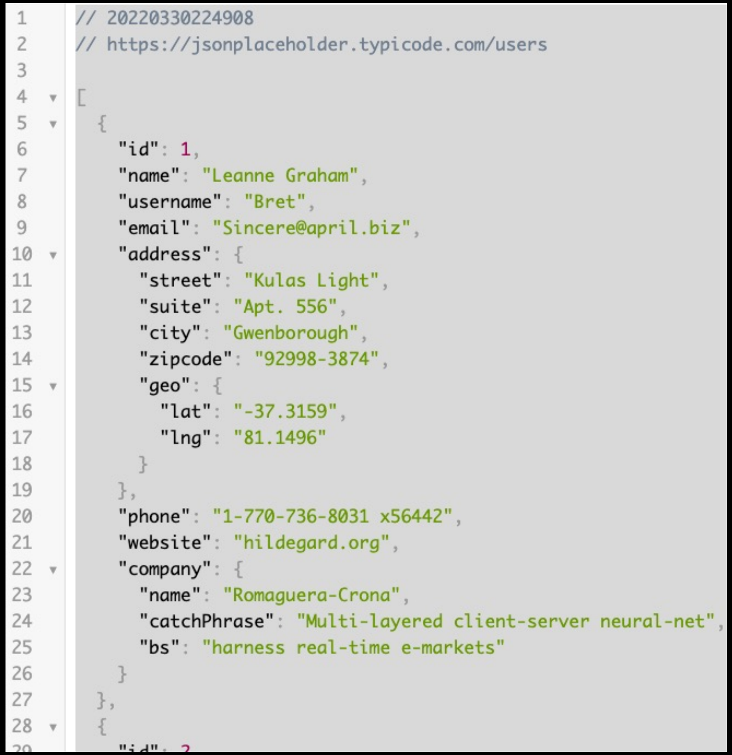
- Typicode
- Typicode Users
- Github Users
- openweather

Common terms relating to api

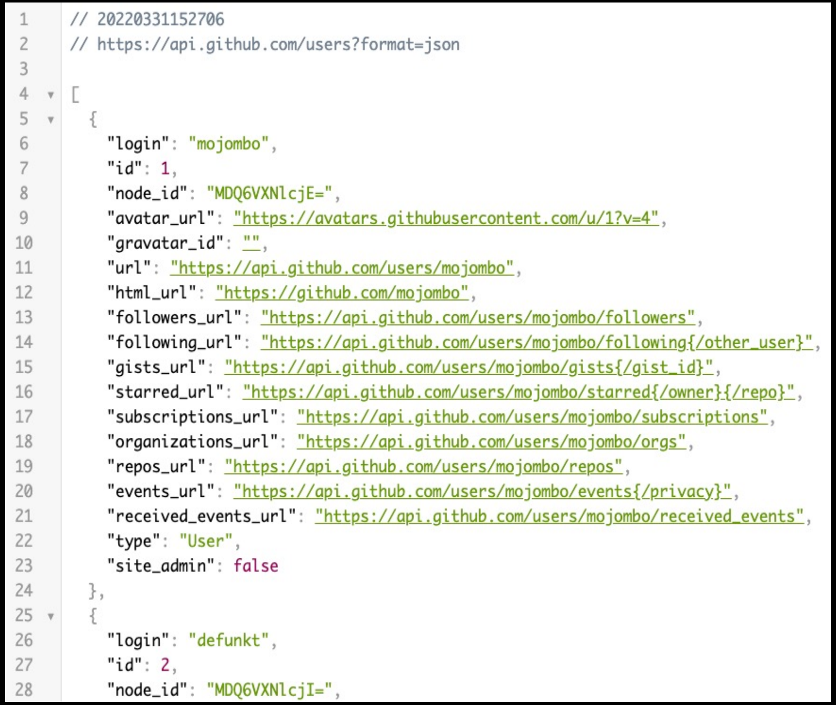
- API
- REST
- Endpoint
- Request
- Response
- Authentication
- Authorisation

Fetching Data sample apis

- https://jsonplaceholder.typicode.com/



```
// 20220330224908
// https://jsonplaceholder.typicode.com/users

[
  {
    "id": 1,
    "name": "Leanne Graham",
    "username": "Bret",
    "email": "Sincere@april.biz",
    "address": {
      "street": "Kulas Light",
      "suite": "Apt. 556",
      "city": "Gwenborough",
      "zipcode": "92998-3874",
      "geo": {
        "lat": "-37.3159",
        "lng": "81.1496"
      }
    },
    "phone": "1-770-736-8031 x56442",
    "website": "hildegard.org",
    "company": {
      "name": "Romaguera-Crona",
      "catchPhrase": "Multi-layered client-server neural-net",
      "bs": "harness real-time e-markets"
    }
  },
  {
    "id": 2
```



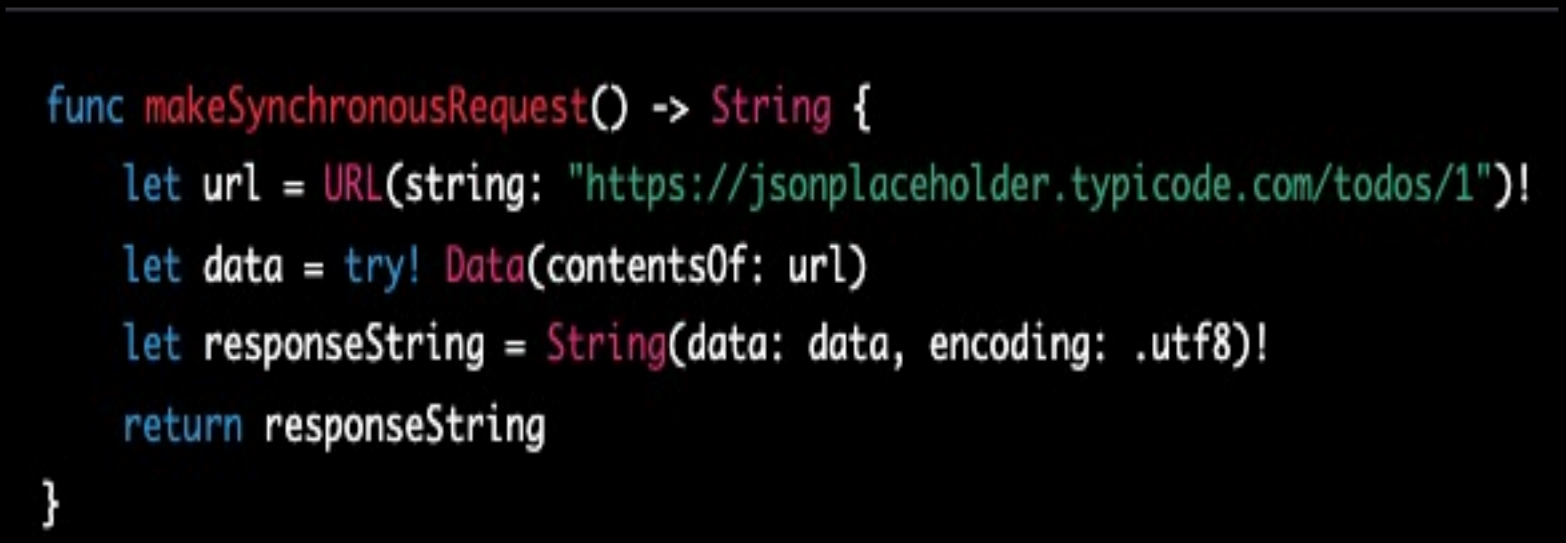
```
// 20220331152706
// https://api.github.com/users?format=json

[
  {
    "login": "mojombo",
    "id": 1,
    "node_id": "MDQ6VXNlcjE=",
    "avatar_url": "https://avatars.githubusercontent.com/u/1?v=4",
    "gravatar_id": "",
    "url": "https://api.github.com/users/mojombo",
    "html_url": "https://github.com/mojombo",
    "followers_url": "https://api.github.com/users/mojombo/followers",
    "following_url": "https://api.github.com/users/mojombo/following{/other_user}",
    "gists_url": "https://api.github.com/users/mojombo/gists{/gist_id}",
    "starred_url": "https://api.github.com/users/mojombo/starred{/owner}{/repo}",
    "subscriptions_url": "https://api.github.com/users/mojombo/subscriptions",
    "organizations_url": "https://api.github.com/users/mojombo/orgs",
    "repos_url": "https://api.github.com/users/mojombo/repos",
    "events_url": "https://api.github.com/users/mojombo/events{/privacy}",
    "received_events_url": "https://api.github.com/users/mojombo/received_events",
    "type": "User",
    "site_admin": false
  },
  {
    "login": "defunkt",
    "id": 2,
    "node_id": "MDQ6VXNlcjI=",
```

Network layer

- As in web requests there can be many pitfalls to retrieving data
- Connection problems
- Data corruption
- Applications have to manage all these possibilities if the app is not to crash.
- HTTP Status Codes:
- 1XX, 2XX, 3XX, 4XX, 5XX - w3schools.com

Network requests

- Synchronous API call:
- Request is made and the main thread of the app is blocked until the response is received.
- Example code:



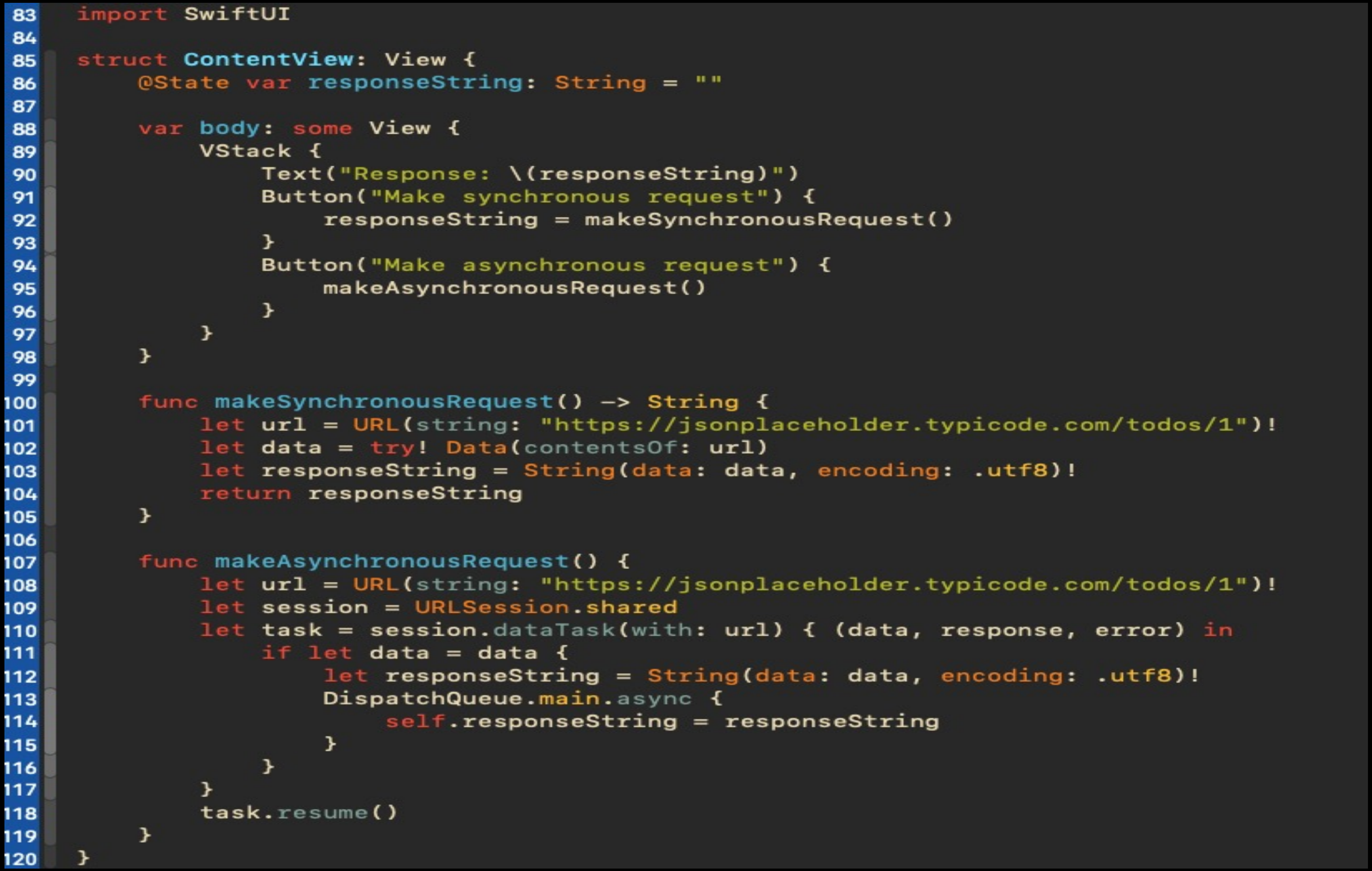
```swift
func makeSynchronousRequest() -> String {
    let url = URL(string: "https://jsonplaceholder.typicode.com/todos/1")!
    let data = try! Data(contentsOf: url)
    let responseString = String(data: data, encoding: .utf8)!
    return responseString
}
```

Network requests

- Asynchronous API call:
- Request is made and response is fetched in the background, no blocking main thread leading better performance.
- Example code:

- See code example later

Network requests – example code



```swift
import SwiftUI

struct ContentView: View {
    @State var responseString: String = ""

    var body: some View {
        VStack {
            Text("Response: \(responseString)")
            Button("Make synchronous request") {
                responseString = makeSynchronousRequest()
            }
            Button("Make asynchronous request") {
                makeAsynchronousRequest()
            }
        }
    }

    func makeSynchronousRequest() -> String {
        let url = URL(string: "https://jsonplaceholder.typicode.com/todos/1")!
        let data = try! Data(contentsOf: url)
        let responseString = String(data: data, encoding: .utf8)!
        return responseString
    }

    func makeAsynchronousRequest() {
        let url = URL(string: "https://jsonplaceholder.typicode.com/todos/1")!
        let session = URLSession.shared
        let task = session.dataTask(with: url) { (data, response, error) in
            if let data = data {
                let responseString = String(data: data, encoding: .utf8)!
                DispatchQueue.main.async {
                    self.responseString = responseString
                }
            }
        }
        task.resume()
    }
}
```

Managing API response

- App developer is responsible for correct management of API request
- Data returned is usually in JSON format (flat and nested) but it is possible to seek it as XML.
- JSON data objects must be mapped to Swift structs and then it becomes a matter of displaying the information.
- It is usual to follow MVVM pattern – Model, ViewModel, View

Swift terms and objects to manage api call

- URL – a class that represents the URL of the API endpoint.
  - `let url = URL(string: "https://api.example.com/data")`

URLRequest: is a class in Swift that represents an HTTP request, including its URL, HTTP method, and headers.

```
var request = URLRequest(url: url)
request.httpMethod = "POST"
request.addValue("application/json", forHTTPHeaderField: "Content-Type")
request.httpBody = jsonData
```

Swift terms and objects to manage api call

URLSession: is a class in Swift that provides an interface for making HTTP requests and handling responses.

```
let session = URLSession(configuration: .default)
let task = session.dataTask(with: request) { data, response, error in

// handle the response

}
task.resume()
```

Swift terms and objects to manage api call

- DataTask: URLSessionDataTask is a subclass of URLSessionTask that represents a data task, which is used for fetching data from a URL
- create a data task by calling the dataTask(with:) method on a URLSession object, passing in a URLRequest object.

```
      let task = session.dataTask(with: request) { data, response, error in
  // handle the response
  }
 task.resume()
```

- JSONDecoder

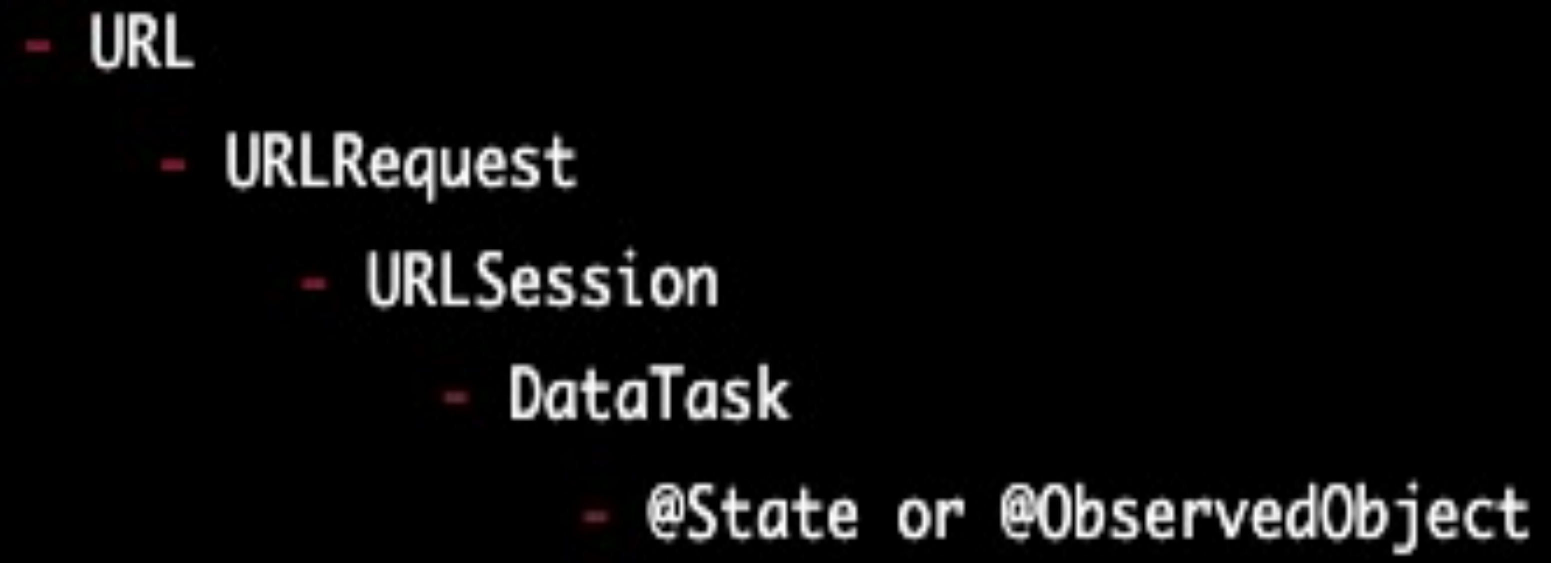
Swift terms and objects to manage api call

- Shared: shared is a static property on URLSession that returns a shared session object.

```
let session = URLSession.shared let task = session.dataTask(with: request) {
data, response, error in

// handle the response

}
 task.resume()
```

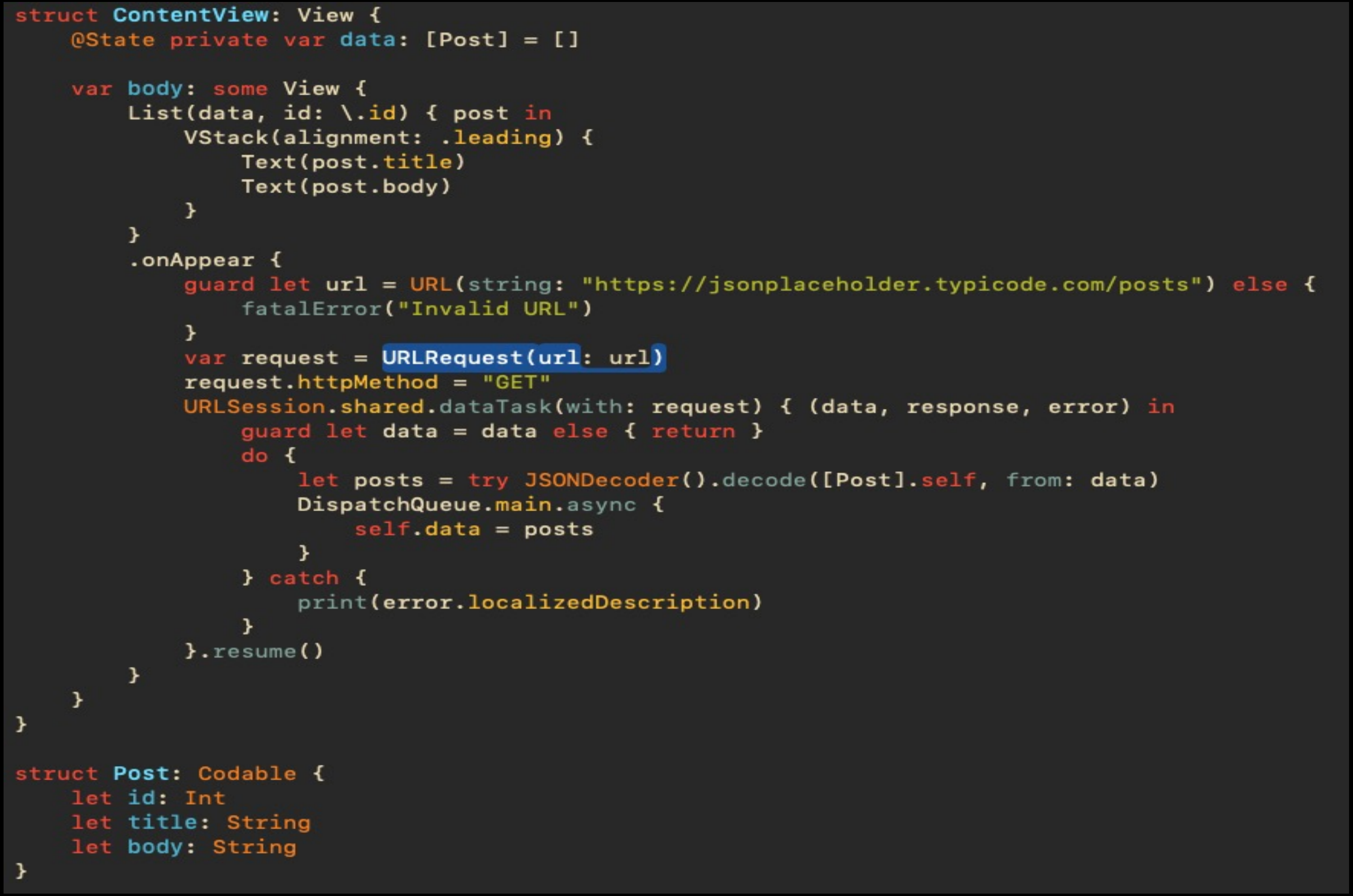
- JSONDecoder

Overview



- URL
    - URLRequest
        - URLSession
            - DataTask
                - @State or @ObservedObject

API call example



```swift
struct ContentView: View {
    @State private var data: [Post] = []

    var body: some View {
        List(data, id: \.id) { post in
            VStack(alignment: .leading) {
                Text(post.title)
                Text(post.body)
            }
        }
        .onAppear {
            guard let url = URL(string: "https://jsonplaceholder.typicode.com/posts") else {
                fatalError("Invalid URL")
            }
            var request = URLRequest(url: url)
            request.httpMethod = "GET"
            URLSession.shared.dataTask(with: request) { (data, response, error) in
                guard let data = data else { return }
                do {
                    let posts = try JSONDecoder().decode([Post].self, from: data)
                    DispatchQueue.main.async {
                        self.data = posts
                    }
                } catch {
                    print(error.localizedDescription)
                }
            }.resume()
        }
    }
}

struct Post: Codable {
    let id: Int
    let title: String
    let body: String
}
```

Completion Handlers

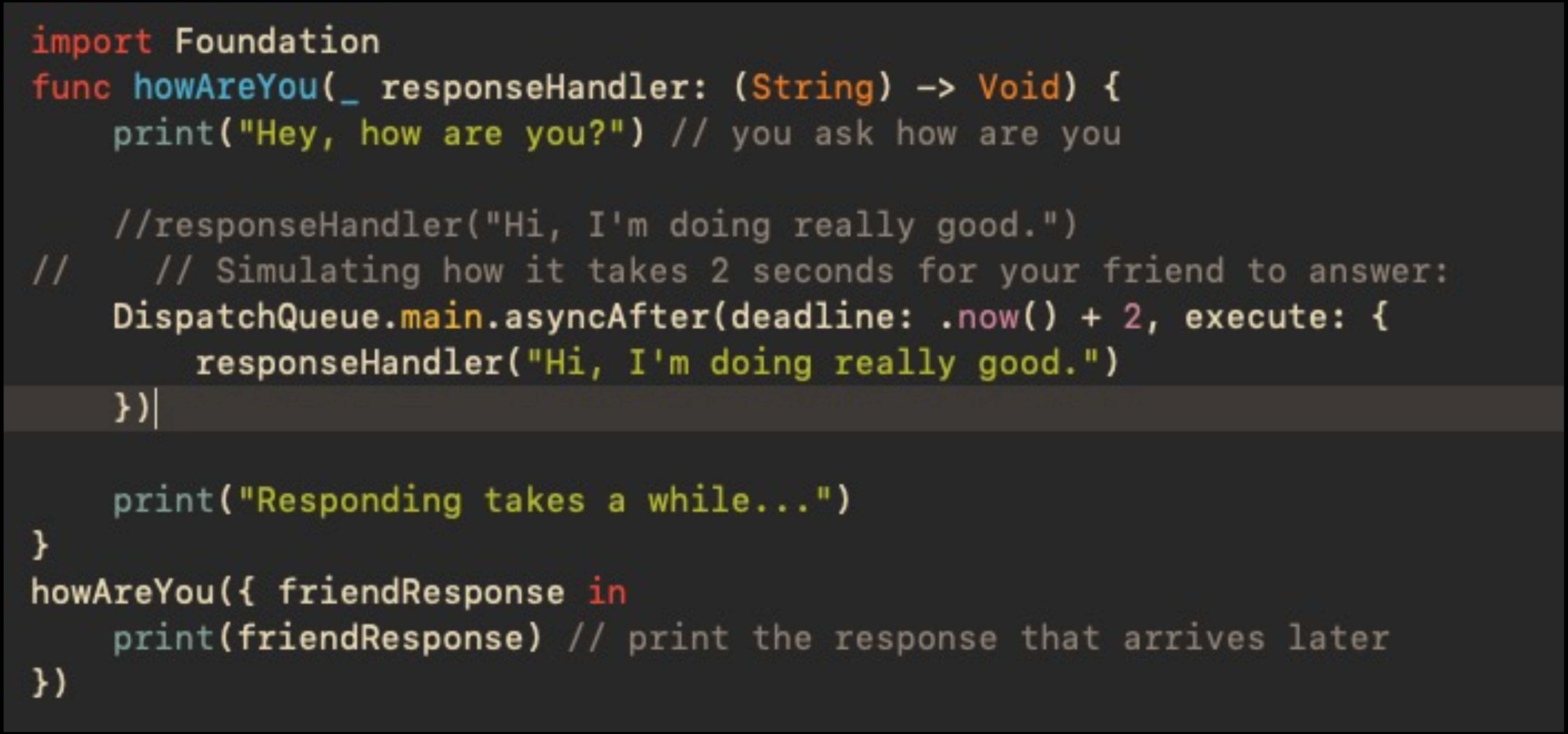
- A completion handler in Swift is a function to handle data and any errors that occur during the request
- A completion handler is a closure that is called when the API request completes, and it provides a way to handle the response data asynchronously.
- This allows the rest of the app to continue executing while the API request is being made.
- Synchronous API calls, on the other hand, block the current thread until the API request is complete.

Completion Handlers – non escaping

- A non-escaping completion handler is a closure that's called synchronously before the function returns. The closure is executed immediately and cannot be stored or used outside of the function.

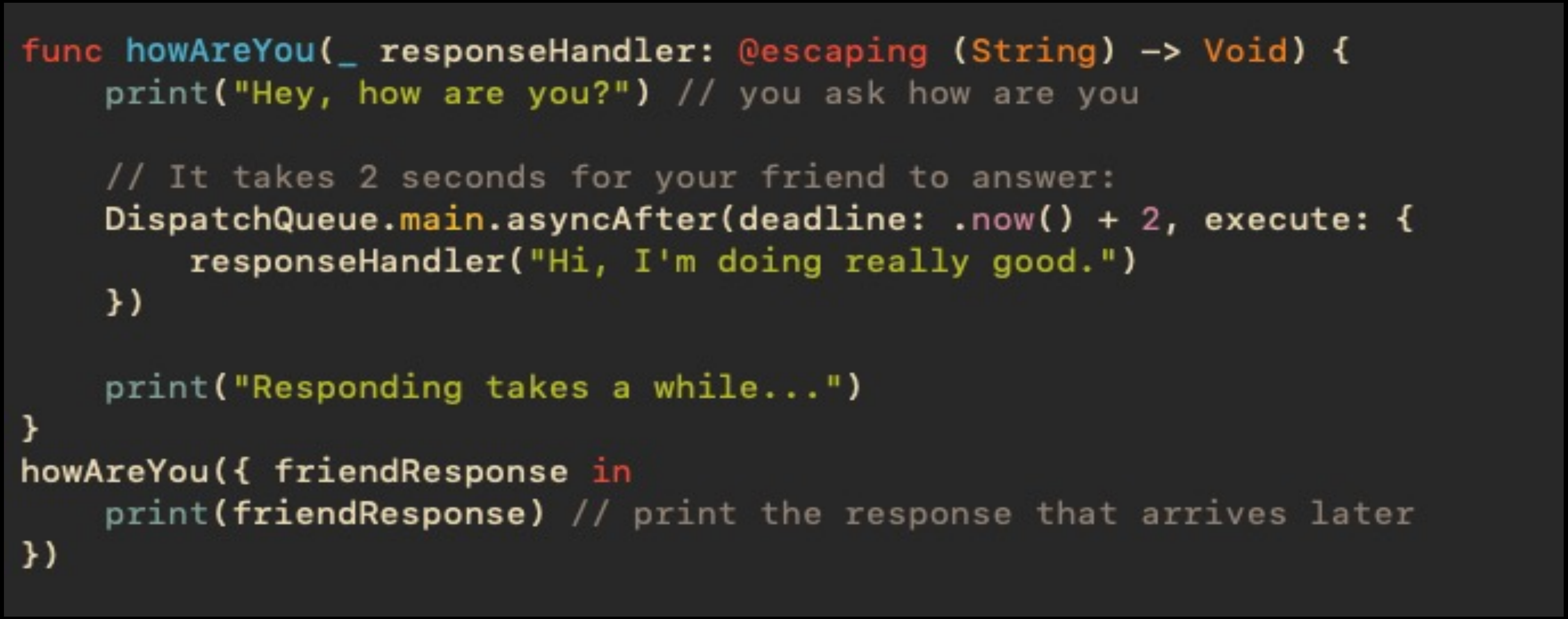
```
func calculateSum(_ a: Int, _ b: Int, completionHandler: (Int) -> Void) {
 let sum = a + b
completionHandler(sum)
}
calculateSum(3, 4) { sum in
print("The sum of 3 and 4 is \(sum)")
}
```

Completion Handlers – non escaping – demo1



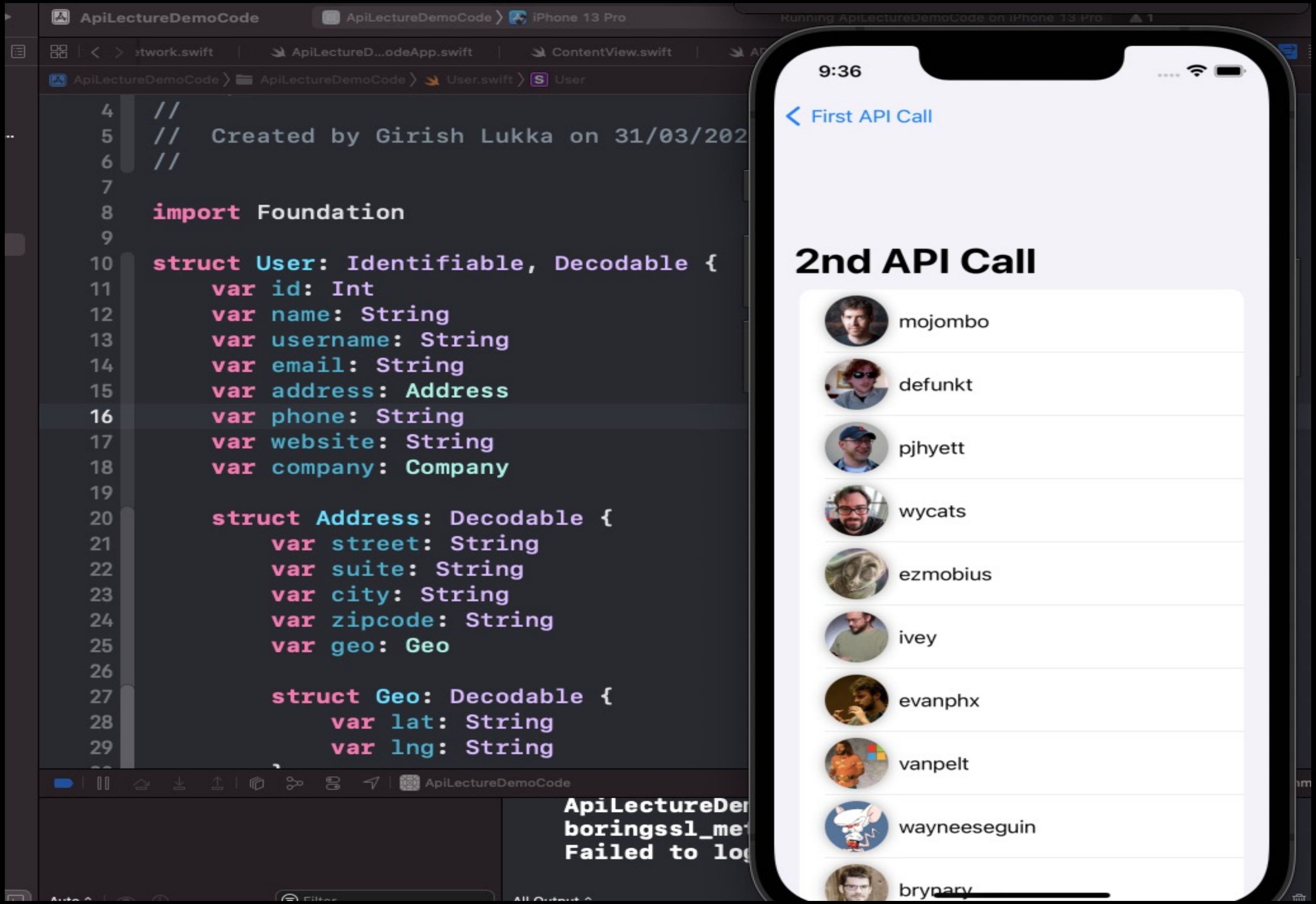
```
import Foundation
func howAreYou(_ responseHandler: (String) -> Void) {
    print("Hey, how are you?") // you ask how are you

    //responseHandler("Hi, I'm doing really good.")
//    // Simulating how it takes 2 seconds for your friend to answer:
    DispatchQueue.main.asyncAfter(deadline: .now() + 2, execute: {
        responseHandler("Hi, I'm doing really good.")
    })

    print("Responding takes a while...")
}
howAreYou({ friendResponse in
    print(friendResponse) // print the response that arrives later
})
```

Completion Handlers – escaping



```swift
func howAreYou(_ responseHandler: @escaping (String) -> Void) {
    print("Hey, how are you?") // you ask how are you

    // It takes 2 seconds for your friend to answer:
    DispatchQueue.main.asyncAfter(deadline: .now() + 2, execute: {
        responseHandler("Hi, I'm doing really good.")
    })

    print("Responding takes a while...")
}
howAreYou({ friendResponse in
    print(friendResponse) // print the response that arrives later
})
```

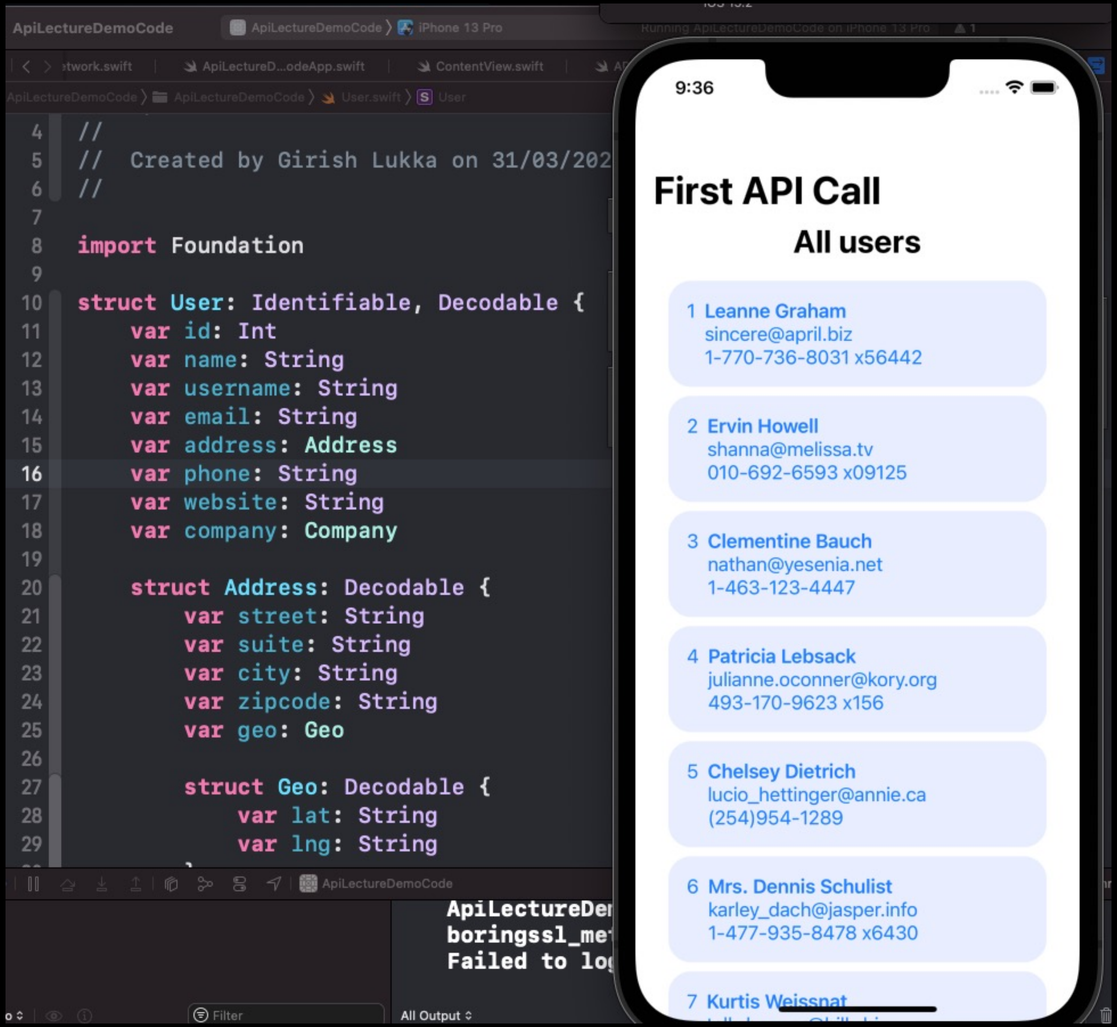
Demo 1 – GitHub Users



```swift
//
//  Created by Girish Lukka on 31/03/202
//

import Foundation

struct User: Identifiable, Decodable {
    var id: Int
    var name: String
    var username: String
    var email: String
    var address: Address
    var phone: String
    var website: String
    var company: Company

    struct Address: Decodable {
        var street: String
        var suite: String
        var city: String
        var zipcode: String
        var geo: Geo

        struct Geo: Decodable {
            var lat: String
            var lng: String
```

```
ApiLectureDe
boringssl_me
Failed to log
```

Demo 2 – typicode users


ApiLectureDemoCode
ApiLectureDemoCode 〉 iPhone 13 Pro
Running ApiLectureDemoCode on iPhone 13 Pro
twork.swift
ApiLectureD...odeApp.swift
ContentView.swift
ApiLectureDemoCode 〉 ApiLectureDemoCode 〉 User.swift 〉 User
ApiLectureDemoCode
Filter
All Output

```swift
//
//  Created by Girish Lukka on 31/03/202
//

import Foundation

struct User: Identifiable, Decodable {
    var id: Int
    var name: String
    var username: String
    var email: String
    var address: Address
    var phone: String
    var website: String
    var company: Company

    struct Address: Decodable {
        var street: String
        var suite: String
        var city: String
        var zipcode: String
        var geo: Geo

        struct Geo: Decodable {
            var lat: String
            var lng: String
```

```
ApiLectureDer
boringssl_met
Failed to log
```

9:36

First API Call

All users

1 Leanne Graham
sincere@april.biz
1-770-736-8031 x56442

2 Ervin Howell
shanna@melissa.tv
010-692-6593 x09125

3 Clementine Bauch
nathan@yesenia.net
1-463-123-4447

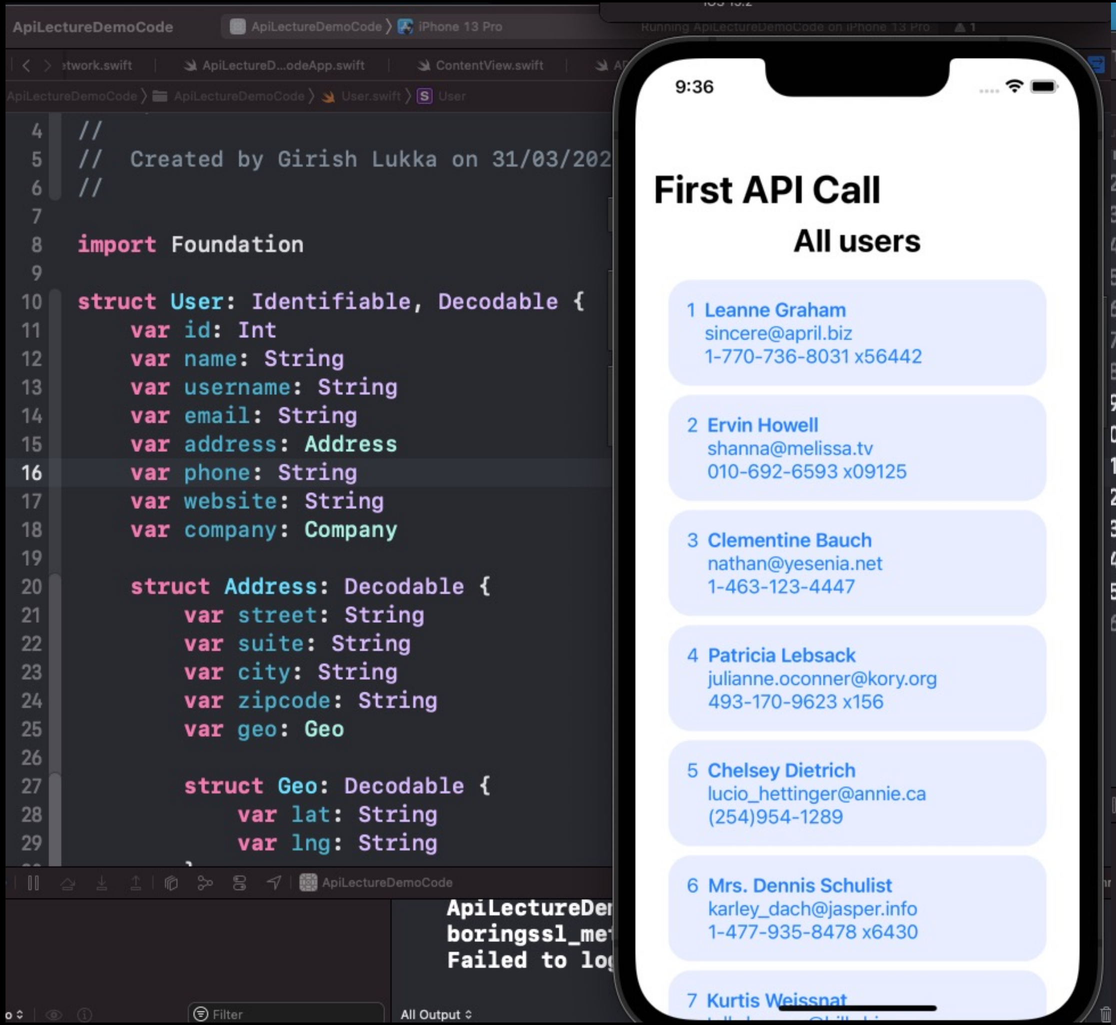
4 Patricia Lebsack
julianne.oconner@kory.org
493-170-9623 x156

5 Chelsey Dietrich
lucio_hettinger@annie.ca
(254)954-1289

6 Mrs. Dennis Schulist
karley_dach@jasper.info
1-477-935-8478 x6430

7 Kurtis Weissnat

ComDemo 3 – Single App with 2 API calls


ApiLectureDemoCode
ApiLectureDemoCode 〉 iPhone 13 Pro
etwork.swift
ApiLectureD...odeApp.swift
ContentView.swift
ApiLectureDemoCode 〉 ApiLectureDemoCode 〉 User.swift 〉 User
ApiLectureDemoCode
Filter
All Output

```swift
//
//  Created by Girish Lukka on 31/03/202
//

import Foundation

struct User: Identifiable, Decodable {
    var id: Int
    var name: String
    var username: String
    var email: String
    var address: Address
    var phone: String
    var website: String
    var company: Company

    struct Address: Decodable {
        var street: String
        var suite: String
        var city: String
        var zipcode: String
        var geo: Geo

        struct Geo: Decodable {
            var lat: String
            var lng: String
```

```
ApiLectureDem
boringssl_me
Failed to log
```

9:36

First API Call

All users

1 Leanne Graham
sincere@april.biz
1-770-736-8031 x56442

2 Ervin Howell
shanna@melissa.tv
010-692-6593 x09125

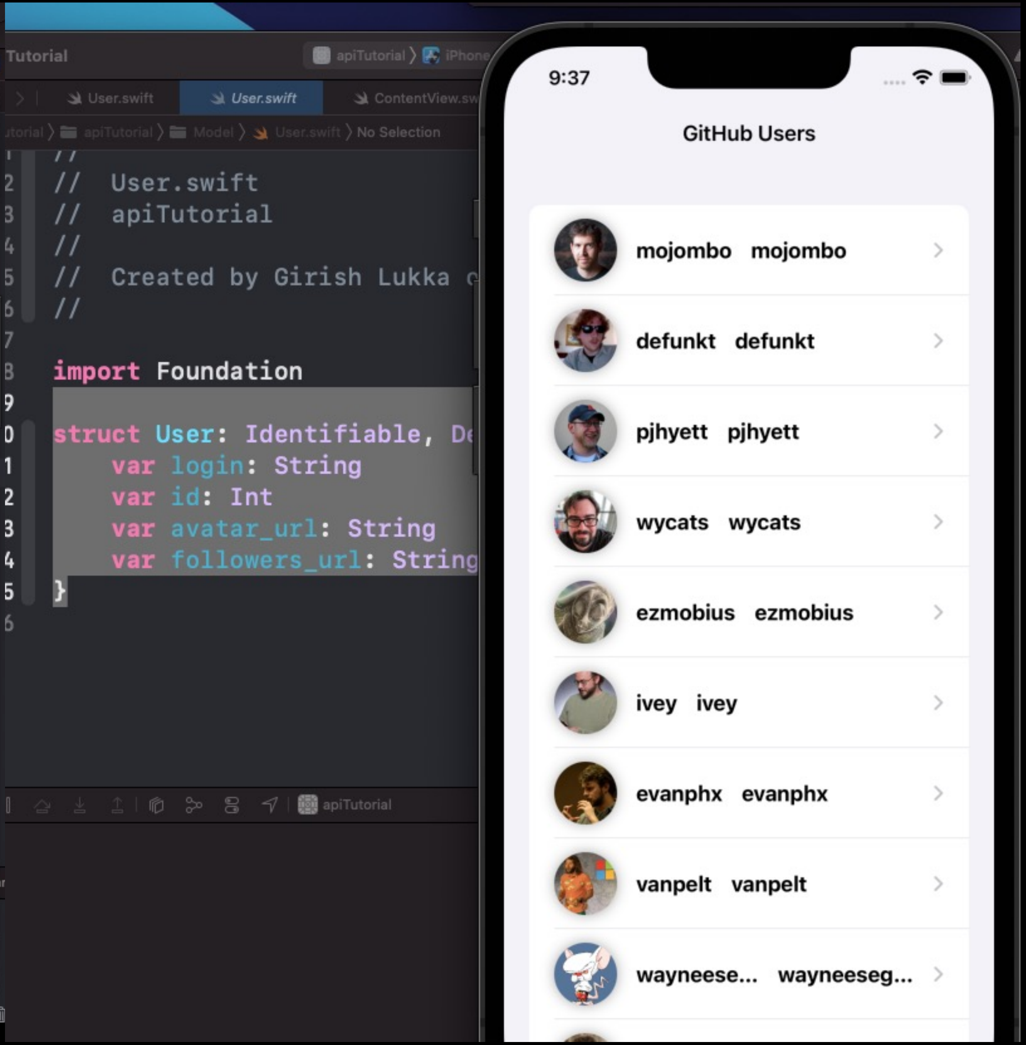
3 Clementine Bauch
nathan@yesenia.net
1-463-123-4447

4 Patricia Lebsack
julianne.oconner@kory.org
493-170-9623 x156

5 Chelsey Dietrich
lucio_hettinger@annie.ca
(254)954-1289

6 Mrs. Dennis Schulist
karley_dach@jasper.info
1-477-935-8478 x6430

7 Kurtis Weissnat



```swift
//
//  User.swift
//  apiTutorial
//
//  Created by Girish Lukka c
//

import Foundation

struct User: Identifiable, De
    var login: String
    var id: Int
    var avatar_url: String
    var followers_url: String
}
```

9:37

GitHub Users

mojombo	mojombo
defunkt	defunkt
pjhyett	pjhyett
wycats	wycats
ezmobius	ezmobius
ivey	ivey
evanphx	evanphx
vanpelt	vanpelt
wayneese...	wayneeseg...