Chapter 3: Frameworks

In this session, we are going to use frameworks to download some data from a server to show them in our app.

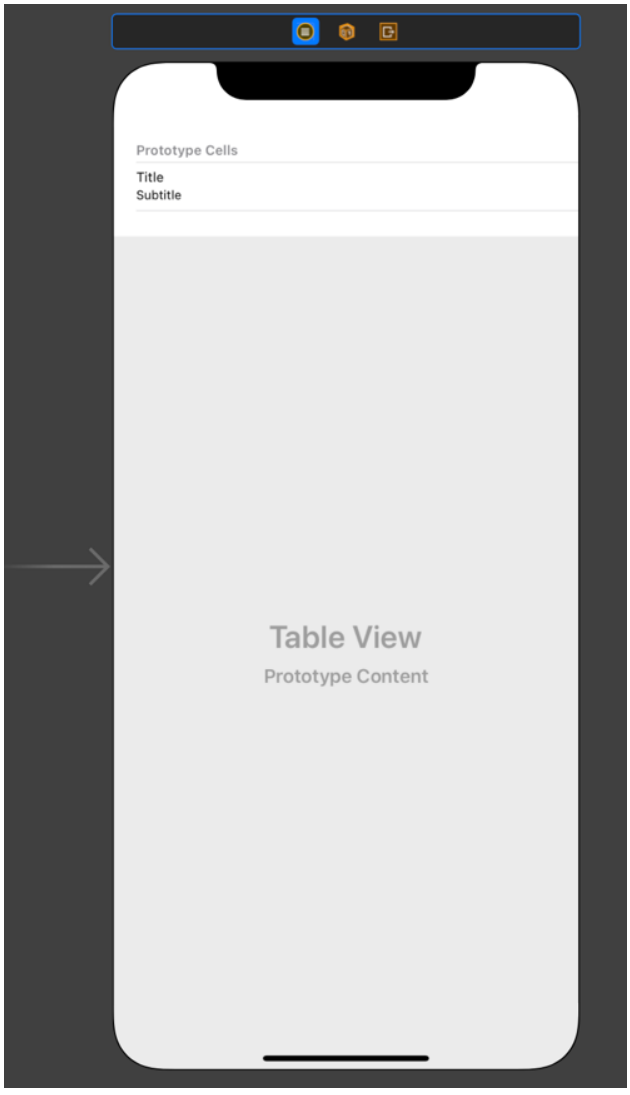
1. Project & UI Creation

Create a new iOS app named GameCritics.

Remove the file ViewController.swift and his ViewController on the Main.storyboard, we are not going to use them.

Add a TableViewController to your Storyboard and set the Prototype cell style to Subtitle

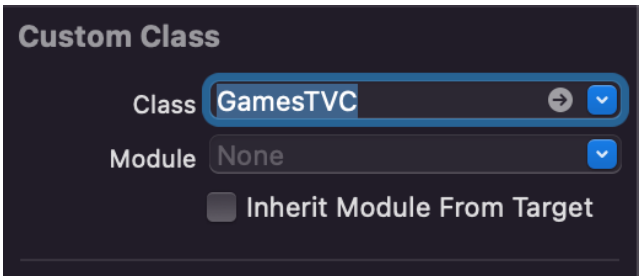
Don't forget to give a reuse identifier to your cell.



2. The Table View Controller

Now, you can create a new Cocoa Touch Class file called *GamesTVC.swift*, subclass of UITableViewController.

Then select the TableViewController you created on your Storyboard and on the *identity inspector,* fill the custom class's class field with GameTVC



3. Create the model

Add a new "swift file" named "Game.swift" and create a Struct "Game" that implement the "Decodable" protocol.

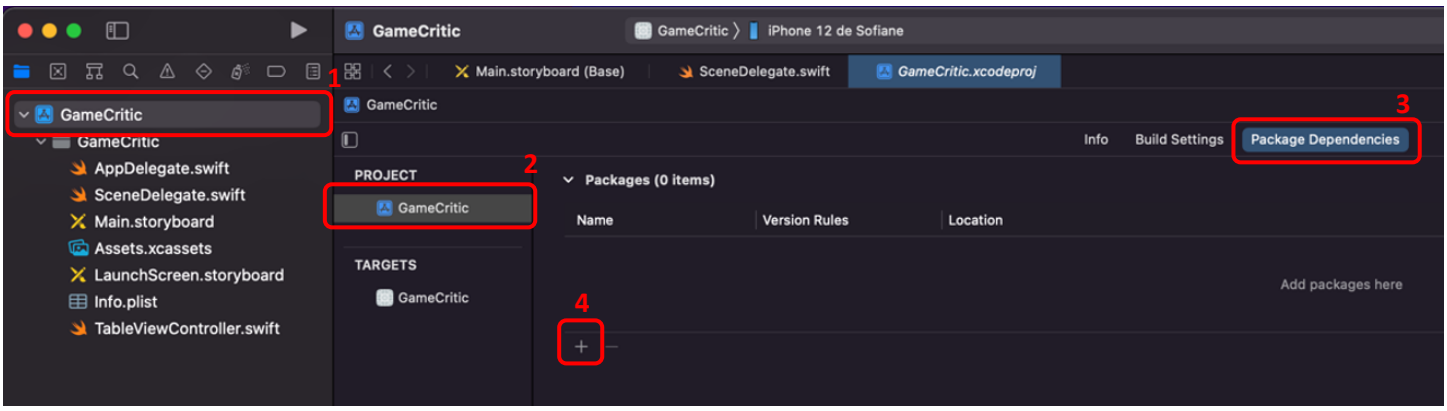
Add the necessary fields according to the JSON file (https://education.3ie.fr/ios/StarterKit/GameCritic/GameCritics.json)

4. Package Managers & Frameworks

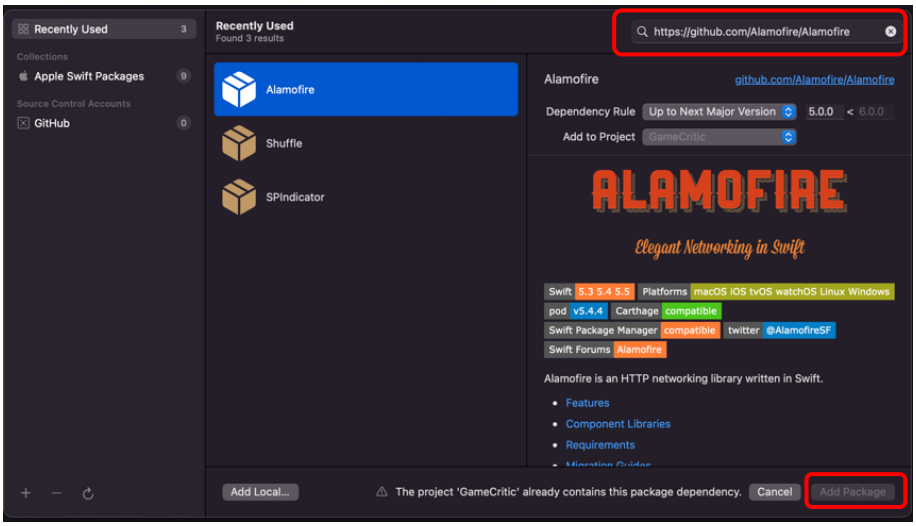
Now, it's time to add some frameworks to your app.

For that, we will use Swift Package Manager.

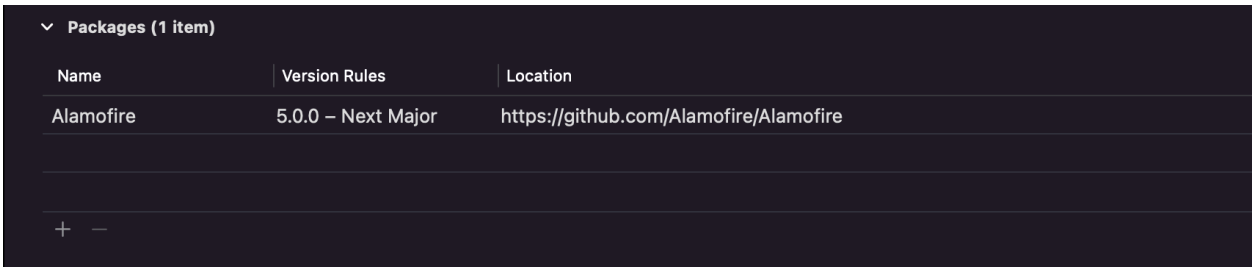
Go into your Project, select Package Dependencies and click on the "+" to add the frameworks.



Then, on the search bar, type the url of the framework you want to add (https://github.com/Alamofire/Alamofire) and click on “Add Package”.



If everything went well, you should see Alamofire on the packages list.



Now, do the same to add

- AlamofireImage (https://github.com/Alamofire/AlamofireImage), dependency to next major version

5. Making requests

Now, to get the data, return into GamesTVC.swift.

Create an empty array of Game and import Alamofire & AlamofireImage.

Downloading a json is easy with Alamofire. This is just to show you, don't put it in your code.

```
let url = "https://education.3ie.fr/ios/StarterKit/GameCritic/GameCritics.json"
AF.request(url).response { response in
    guard let data = response.data else { return }
}
```

Once you have your data, you could decode the json the same way as before.

```
if let games = try? JSONDecoder().decode([Game].self, from: data) {
    print(games)
}
```

But since we are using Alamofire we can do something with less boilerplate code. We are going to use the "responseDecodable" method so that we don't even have to use JSONDecoder ourselves. Then we use the associated value in the "response" enum to get the list of games.

```
AF.request(url).responseDecodable(of:[Game].self) { response in
    switch response.result {
    case .success(let games):
        print(games)
        //TODO: store your games
        self.tableView.reloadData()
    case .failure(_):
        return
    }
}
```

Now put that code in a method in your GameTVC and call it in "viewDidLoad". Check that everything works and that you can print the list of games. Don't forget to store the games.

Because Alamofire is making requests on the mainThread, you don't have to switch threads manually but it's not very efficient, you are still parsing on the main thread. You can try it if you want, just be sure to use the "DispatchQueue.main.sync" method to reload your tableView.

```
AF.request(url).responseDecodable(of:[Game].self, queue: DispatchQueue.global()) {
    //TODO: your code
}
```

6. Filling your tableView

At this point, if you run your app, nothing will be shown on your screen. If you remember, you must add some code to tell your tableView what to display.

To do so, override the numberOfSections, numberOfRowsInSection and cellForRowAtIndexPath methods of your tableView and give them the data contained into your games array.

```
override func numberOfSections(in tableView: UITableView)
override func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
override func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath)
```

Tip: To set the subtitle text from your prototype cell, use cell.detailTextLabel.text

And … Voila ! Your app now displays the games ! 🥳

A. Images

To download and display the images on our tableView, we'll use ImageView extension provided by AlamofireImage.

Open the GamesTVC.swift file and add this code to your cellForRowAtIndexPath method. You must provide the image Url.

```
cell.imageView?.af.setImage(withURL: urlString, completion: { _ in
        cell.setNeedsLayout()
})
```

Now, Build and Run your app. You should see the images!

Challenge 1

Use the UserDefaults interface to store the JSON data (not the images) into your app.
https://www.hackingwithswift.com/example-code/system/how-to-save-user-settings-using-userdefaults

Limit your “cache” availability to 1mn. You can do that by saving the current (saving) date to the user defaults and at each app launch, comparing the current date to the date you saved