Home assignment #3

Due date: 20.07.2022, 23:59

Home assignment will be consisted of 3 separate assignements:

1. Creating a new view controller: `HomeViewController` and navigating to it
2. Implementing API calls for login and creating an account and then again navigating to `HomeViewController`.

1. assignment - HomeViewController

- Create a new `HomeViewController` (class + storyboard)
- Embedd current `LoginViewController` in `UINavigationController` (in storyboard)
- Using approach from code push `HomeViewController` from `LoginViewController` :
  - on `Login` button press (Touch Up Inside)
  - on `Register` button press (Touch Up Inside)

2. assignment - API calls

- Implement bot API calls for register and login
- API documentation: https://tv-shows.infinum.academy/api/v1/docs/index.html
- `pod "Alamofire"`
- use `.responseDecodable(of: ...)` method for invoking API call
- take a look at demo project how parsing should be done
- **Register API call**:
  - on `Create an account` button press (Touch Up Inside)
  - `POST https://tv-shows.infinum.academy/users`
  - Check the API documentation for request/response ;)
  - Before invoking API call check if email and password exists (`isEmpty` on `String`)
    - You should have `UITextField` for email and password, and it has property `emailField.text`
  - Show spinner before invoking call
  - Hide spinner when API call finishes: `MBProgressHUD.hide(for: self.view, animated: true)`
  - On success:
    - parse JSON data to `UserResponse` Codable model and save it as private property inside `LoginViewController`
    - check the API documentation for `UserResponse` model properties ;)
    - try to extract the headers from response (there is in a demo project how to, but please try first on your own!) (they'll be used in next home assignment)
    - push `HomeViewController`

    - On failure:
        - print error in console
- **Login API call**:
    - on `Login` button press (Touch Up Inside)
    - `POST https://tv-shows.infinum.academy/users/sign_in`
    - Check the API documentation for request/response :)
    - Show spinner before invoking call
    - Hide spinner when API call finishes: `MBProgressHUD.hide(for: self.view, animated: true)`
    - On success:
        - parse JSON data to `UserResponse` Codable model and save it as private property inside `LoginViewController`
        - check the API documentation for `UserResponse` model properties ;)
        - try to extract the headers from response (there is in a demo project how to, but please try first on your own!) (they'll be used in next home assignment)
        - push `HomeViewController`
    - On failure:
        - print error in console

Extra assignment

Having a big dependency like Alamofire inside a view controller is not the best practice in development. Reason why: for example, if Alamofire team decides to ditch the development of it in future we'll have to remove it and update the networking in every single view controller. In production apps you'll have more than 100 of them...

So rather to import Alamofire in each of our view controllers we should extract Alamofire behind some service/manager which will provide interface for our app towards the Alamofire.

Instead of using `AF.request...` directly inside view controller you should create your own API singleton manager which will have its own Alamofire session instance (`AF.` is just a shorthand to access `public let AF = Session.default` which uses default `Session` singleton) encapsulated, and exposed methods for creating request and providing a callback closure. Ideally you shouldn't have to have `import Alamofire` inside view controller (more in links).

Also, having to write url and method each time could be improved via using Router pattern in Alamofire (more in links).

Links:

- Creating Custom `Session` Instances
- Router pattern with Alamofire
- Alamofire API Manager

Appendix

> **Take a look at tutorials attached to the lecture #4 and accompanying documentation.**

Podfile

- you can use this as your podfile

```
platform :ios, '13.0'
use_frameworks!

target 'TVShows' do
  pod "Alamofire"
  pod "MBProgressHUD"
end
```

UserResponse and User

```
struct UserResponse: Codable {
    let user: User
}

struct User: Codable {
    let email: String
    let imageUrl: String?
    let id: String

    enum CodingKeys: String, CodingKey {
        case email
        case imageUrl = "image_url"
        case id
    }
}
```

Alamofire example

```
MBProgressHUD.showAdded(to: view, animated: true)

let parameters: [String: String] = [
    "email": email,
    "password": password
]

AF.request(
    "https://tv-shows.infinum.academy/users/sign_in",
    method: .post,
    parameters: parameters,
    encoder: JSONParameterEncoder.default
)
.validate() // Status code in 200 ..< 300 range
.responseDecodable(of: UserResponse.self) { [weak self] response in
```

```
    guard let self = self else { return }
    MBProgressHUD.hide(for: self.view, animated: true)
    switch response.result {
    case .success(let response):
        print("Success: \(response)")
    case .failure(let error):
        print("Failure: \(error)")
    }
}
```