Readme

Telechargement de project depuis github

```
git clone https://github.com/tommypeps/workshop.git
```

Explication cocoapods, usage et configuration

```
touch Podfile
nano Podfile
```

Fichier Podfile:

```
platform :ios, '8.0'

use_frameworks!

target 'Workshop' do
pod 'AFNetworking'
pod 'ObjectMapper', '~> 3.4'
pod 'AlamofireObjectMapper', '~> 5.2'
pod 'Kingfisher', '4.10.1'

end
```

Installation de dépendance:

```
pod install
```

Démarer le project:

```
open 'Workshop'.xcworkspace
```

Definir l'importance du service et commencer la application depuis :

- implémenter LaunchDTO

- implémenter Launch
- implémenter DTOConversor
- implémenter LaunchService
- implémenter LaunchDTO:

```
import ObjectMapper

class LaunchDTO: Mappable {

    var flightNumber = 0
    var launchSuccess = false
    var details = ""
    var missionName = ""
    var missionPathUrl = ""

    required init?(map: Map) {}

    func mapping(map: Map) {
        flightNumber <- map["flight_number"]
        launchSuccess <- map["launch_success"]
        details <- map["details"]
        missionName <- map["mission_name"]
        missionPathUrl <- map["links.mission_patch"]
    }
}
```

- implémenter LaunchModel:

```
struct LaunchModel {
    var flightNumber = 0
    var launchSuccess = false
    var details = ""
    var missionName = ""
    var missionPathUrl = ""
}
```

- implémener DTOConversor:

```
class LaunchDTOConversor {
    static func from(launchDTO: LaunchDTO) -> LaunchModel {
```

```
        var model = LaunchModel()
        model.flightNumber = launchDTO.flightNumber
        model.launchSuccess = launchDTO.launchSuccess
        model.details = launchDTO.details
        model.missionName = launchDTO.missionName
        model.missionPathUrl = launchDTO.missionPathUrl
        return model
    }
}
```

- implémentater LaunchService:

```
import Alamofire
import AlamofireObjectMapper


class LaunchService {
    static func getLaunchs(success: (([LaunchModel]) -> Void)?, failure: ((Error)
        let url = "https://api.spacexdata.com/v3/launches"
        Alamofire.request(url, method: .get).responseArray { (response: DataRespo
            switch response.result {
                case .success(let lauchModelResponses):
                let models = lauchModelResponses.map({ LaunchDTOConversor.from(la
                success?(models)
            case .failure(let error):
                failure?(error)
            }
        }
    }
}
```

- Tester que le service marche:

```
LaunchService.getLaunchs(success: { models in
    print(models)
}) { error in
    print(error)
}
```

- Integration vue:

```
import Kingfisher

class LaunchCell: UITableViewCell {

    @IBOutlet var pathImage: UIImageView!
    @IBOutlet var missionLabel: UILabel!
    @IBOutlet var detailLabel: UILabel!

    func configureCell(model: LaunchModel) {
        if !model.missionPathUrl.isEmpty {
        pathImage.kf.setImage(with: model.missionPathUrl.url)
        }
        missionLabel.text = model.missionName
        detailLabel.text = model.details
    }

    override func prepareForReuse() {
        super.prepareForReuse()
        pathImage.kf.cancelDownloadTask()
    }
}
```

- Definition vue du cell:

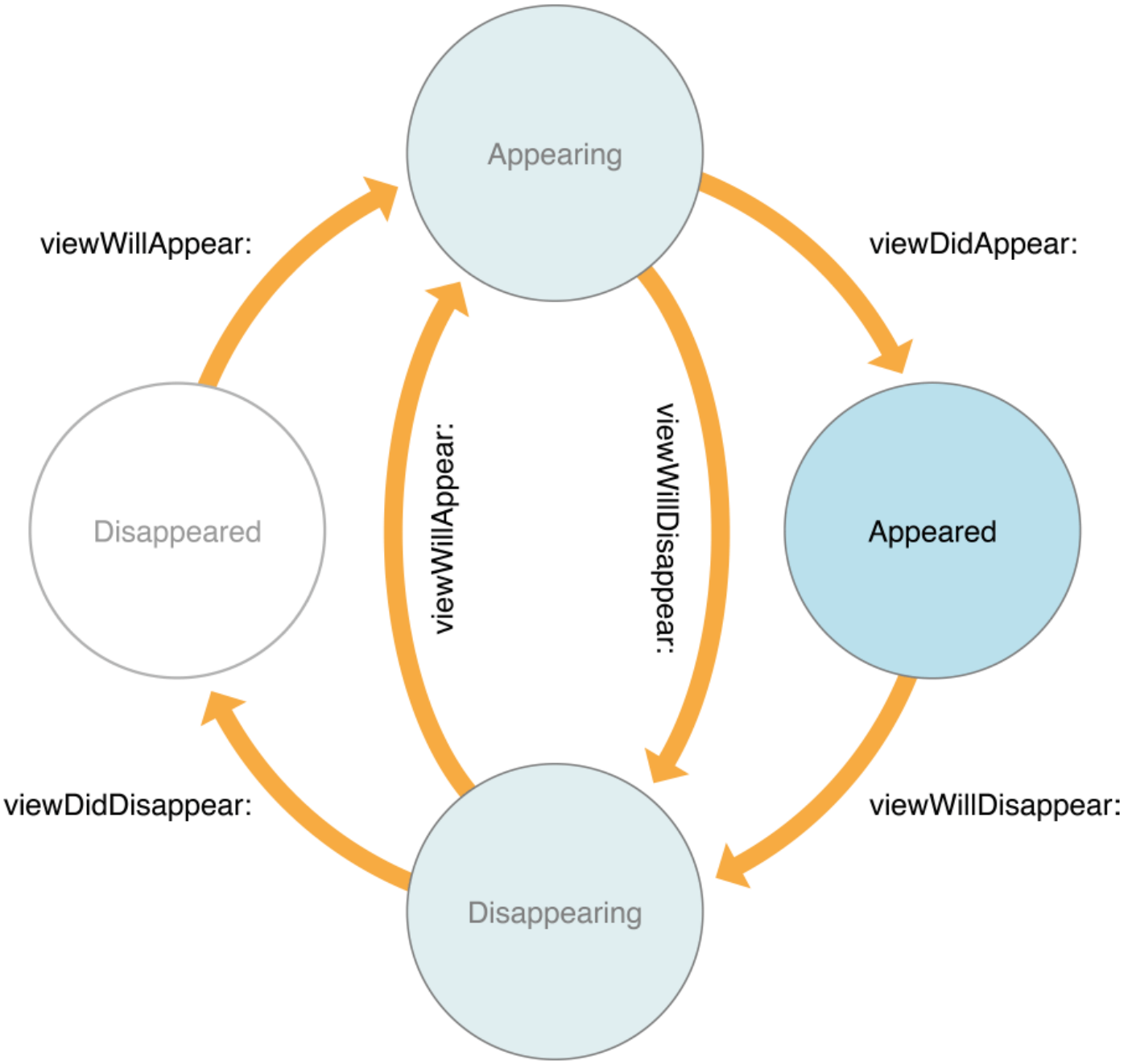
- Definition struct Module:

```
struct Launch {
    static let storyBoardName = "LaunchStorybord"

    struct Constant {
        static let alertTitle = "**alertTitle**"
        static let alertBody = "**alertBody**"
        static let confirmationButton = "**confirmationButton**"
    }

    enum LaunchCell: String, CaseIterable {
        case launchCell = "LaunchCell"
    }
}
```

- Charge storyboard dans l'app:

```
let launchStorybord = UIStoryboard(name: Launch.storyBoardName, bundle: nil)
if let letLaunchViewController =  launchStorybord.instantiateInitialViewControlle
    self.window = UIWindow(frame: UIScreen.main.bounds)
    self.window?.rootViewController = letLaunchViewController
    self.window?.makeKeyAndVisible()
}
```

- Definition storybard cell:
- Definiction ViewController:



- Configuration Cell
- Configuration TableView
- Registration cell
- assignation delegate
- Configuration Datasource
- Configuration alert
- Call service
- Show alert/ reload info

```
import UIKit

class LaunchViewController: UIViewController {

    @IBOutlet var tableView: UITableView!

    var model = [LaunchModel]()
    var alertController: UIAlertController?

    override func viewDidLoad() {
        super.viewDidLoad()
        loadInformation()
        configureTableView()
    }
}

extension LaunchViewController {

    private func configureTableView() {
        // Register cell
        let nib = UINib(nibName: Launch.LaunchCell.launchCell.rawValue, bundle: B
        tableView.register(nib, forCellReuseIdentifier: Launch.LaunchCell.launchC
        // Configue datasource
        tableView.dataSource = self
        tableView.delegate = self
        tableView.estimatedRowHeight = 44.0
        tableView.rowHeight = UITableView.automaticDimension
    }

    private func loadInformation() {
        LaunchService.getLaunchs(success: { [weak self] launchModels in
            self?.model = launchModels
            self?.tableView.reloadData()
```

```
        }) { [weak self] error in
            self?.showError()
        }
    }

    private func showError() {
        let alertAction = UIAlertAction(title: Launch.Constant.confirmationButton
            self.alertController?.dismiss(animated: true, completion: nil)
        }
        alertController = UIAlertController(title: Launch.Constant.alertTitle, me
        alertController?.addAction(alertAction)
        guard let alertController = alertController else { return }
        present(alertController, animated: true)
    }
}

extension LaunchViewController: UITableViewDataSource {

    func tableView(_ tableView: UITableView, numberOfRowsInSection section: Int)
        return self.model.count
    }

    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -
        if let cell = tableView.dequeueReusableCell(withIdentifier: Launch.Launch
            cell.configureCell(model: model[indexPath.row])
            return cell
        }
        return UITableViewCell()
    }
}

extension LaunchViewController: UITableViewDelegate {
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath)
        let modelSelected = model[indexPath.row]
        //TODO: Navigate to Next View
    }
}
```

- Integration vue: [UIViewController](https://developer.apple.com/library/archive/referencelibrary/GettingStarted/DevelopiOSAppsSwift/WorkWithViewControllers.html)

Bonus: migration du code à MVP architecture