Xib Practices

TO-DO

1. Implementar UITabBarController com duas tabs;

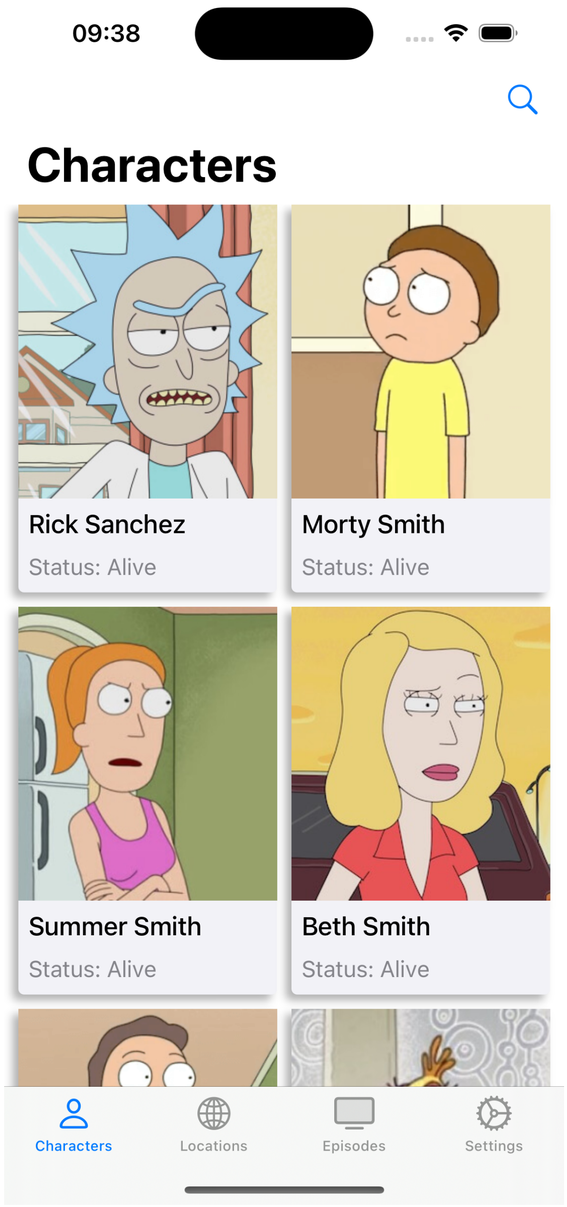
2. Implementar a tela de Episodes;

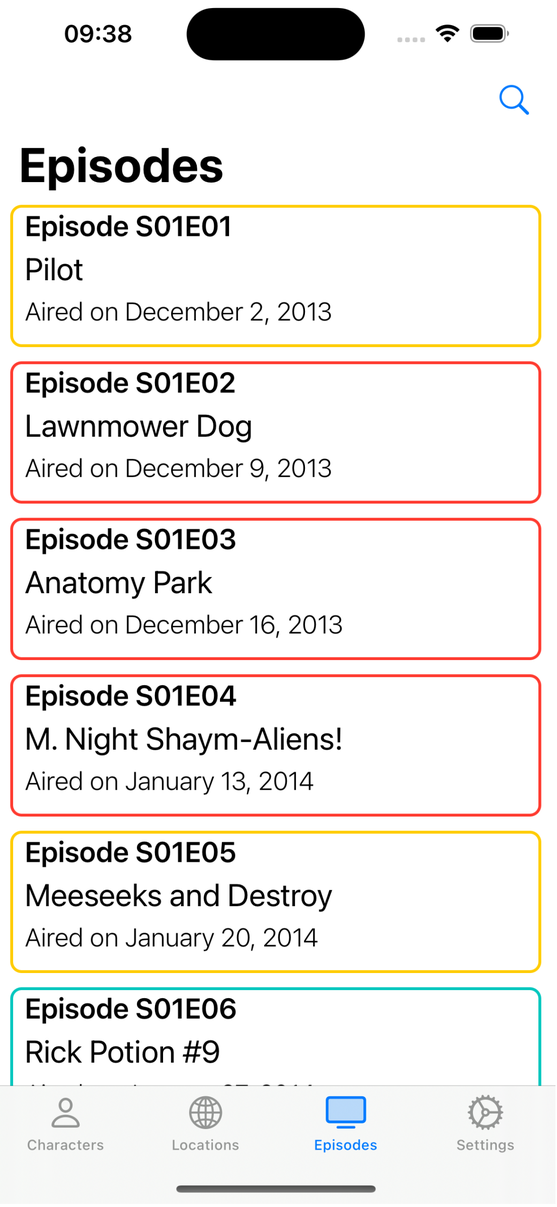
3. Implementar a tela de Characters;

4. Implementar a tela de CharactersDetails (aparece ao clicar num elemento da tela de Characters) - propagar os dados.

Detalhes da implementação

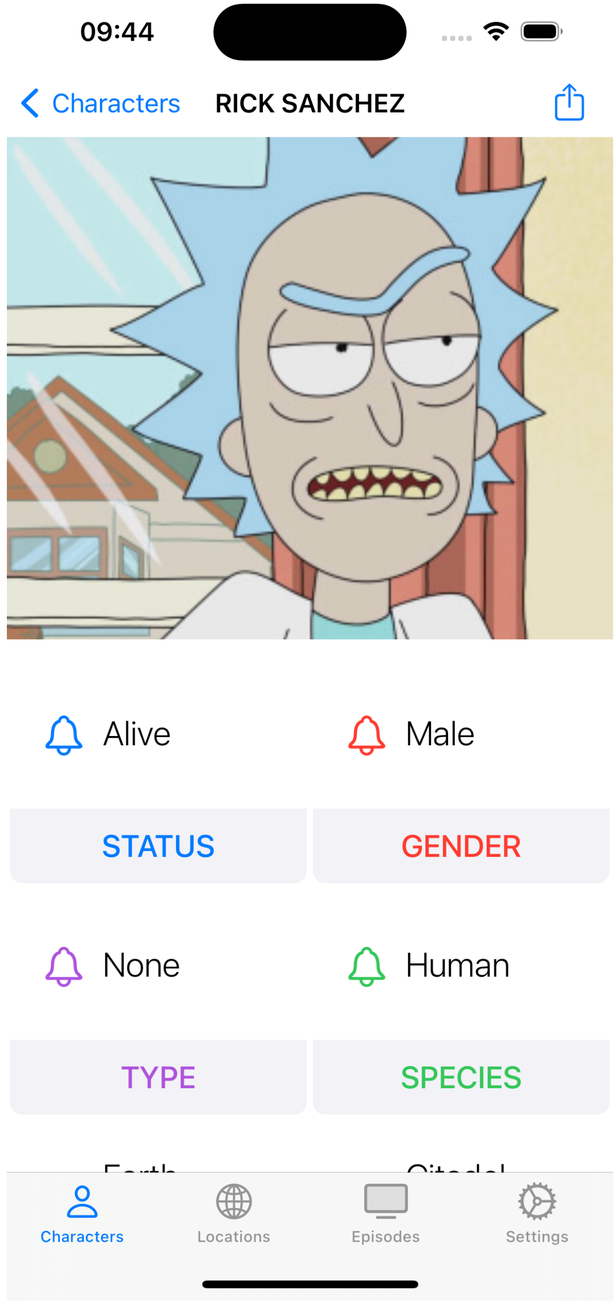
- 2 tabs: Characters e Episodes
- Endpoints para recuperar as informações:
  - GET https://rickandmortyapi.com/api/character
  - GET https://rickandmortyapi.com/api/episode
- Os models já se encontram no código
- Documentação: https://rickandmortyapi.com/documentation/





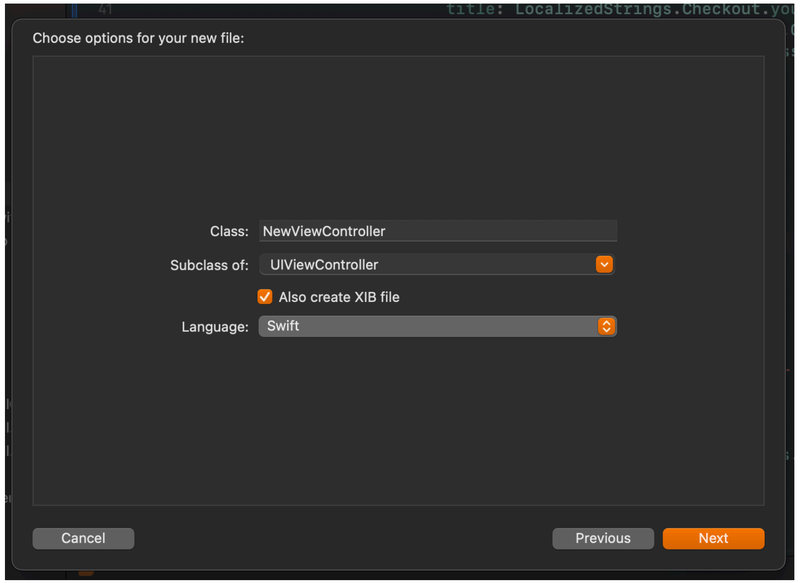
Characters

- Ao clicar em um elemento da tela Characters, apresentar uma UIScrollView parecida com a próxima:



UIViewController com .xib

- Criar uma CocoaTouchClass com .xib
- Adicionar o nibName na inicialização




```swift
class NewViewController: UIViewController {

    // MARK: - Initialization
    public init() {
        super.init(
            nibName: String(describing: NewViewController.self),
            bundle: nil
        )
    }

    @available(*, unavailable)
    required init?(coder decoder: NSCoder) {
        preconditionFailure("init(coder:) has not been implemented")
    }

    // MARK: - Overrides
    override func viewDidLoad() {
        super.viewDidLoad()
    }
}
```

UITableView e UICollectionView com .xib

- Criar uma CocoaTouchClass com .xib para as células
- Registrar o nibName para o tipo de listagem escolhido

```swift
private func configureCollectionView() {
    collectionView.delegate = self
    collectionView.dataSource = self
    let nib = UINib(nibName: "CharacterCollectionViewCell", bundle: nil)
    collectionView.register(nib, forCellWithReuseIdentifier: "characterCell")
}
```

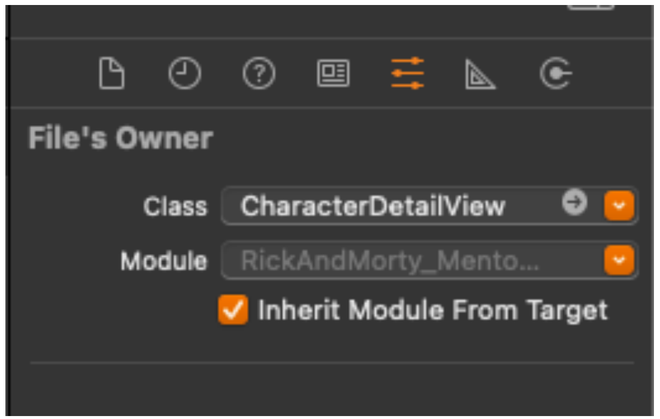
UIView com .xib

- Herdar a classe XibView na sua .swift
- Adicionar o File’s Owner na sua .xib
- Usar o commonXibInit como a função de inicialização

```swift
class CharacterDetailView: XibView {

    @IBOutlet var imageView: UIImageView!
    @IBOutlet var valueLabel: UILabel!
    @IBOutlet var typeView: UIView!
    @IBOutlet var typeLabel: UILabel!

    // MARK: - Init
    override func commonXibInit() {
        let viewColor = UIColor.random()
        configureTypeView()
        configureTypeLabel(with: viewColor)
        configureImageView(with: viewColor)
    }
```



Dicas

- Usar a .xib para implementar o “esqueleto” da tela:
  - Adicione os elementos visuais
  - Adicione constraints
- Deixe textos, cores e outras formas de customização para serem implementadas no código.