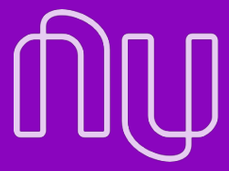


Evoluindo a Arquitetura iOS do Nubank

Eduardo Pinto
@edulpn

Piera Marchesini
@pieramarchesini

08/12/2018

Show of hands

Quem aqui:

- É cliente do Nubank

O que é Nubank?

O que é Nubank?

- Mais de 1200 funcionários
- Diversidade
- Mais de 5M de clientes
- App first
- Cultura de qualidade e testes

História

Linha do tempo

Setembro 2014

Lançamento do aplicativo

Linha do tempo

Setembro 2014 — Agosto 2017

Lançamento do aplicativo

Rewards

Linha do tempo

Setembro 2014

Agosto 2017

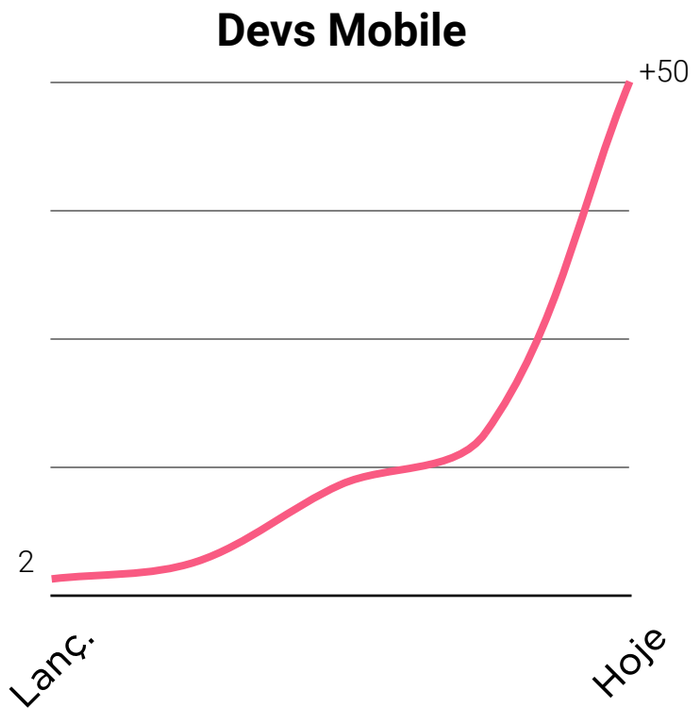
Outubro 2017

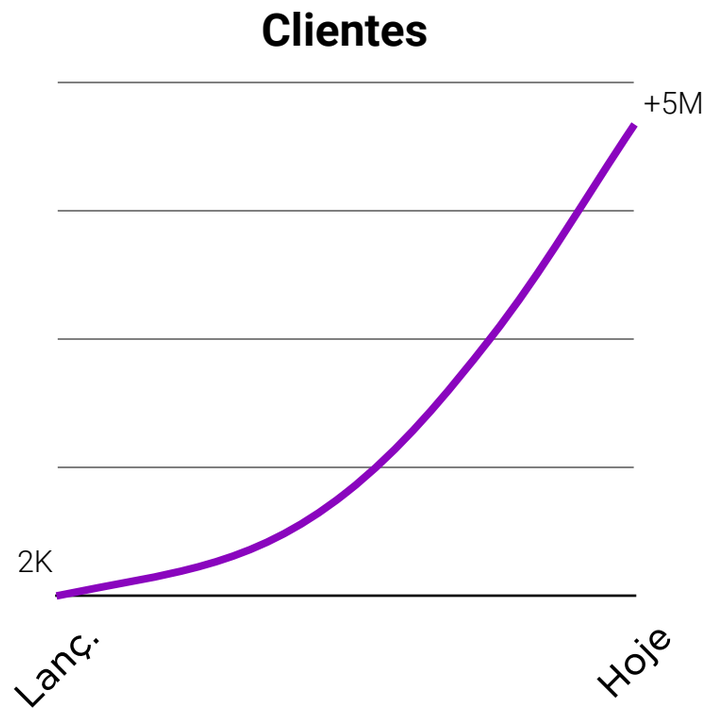
Lançamento do aplicativo

Rewards

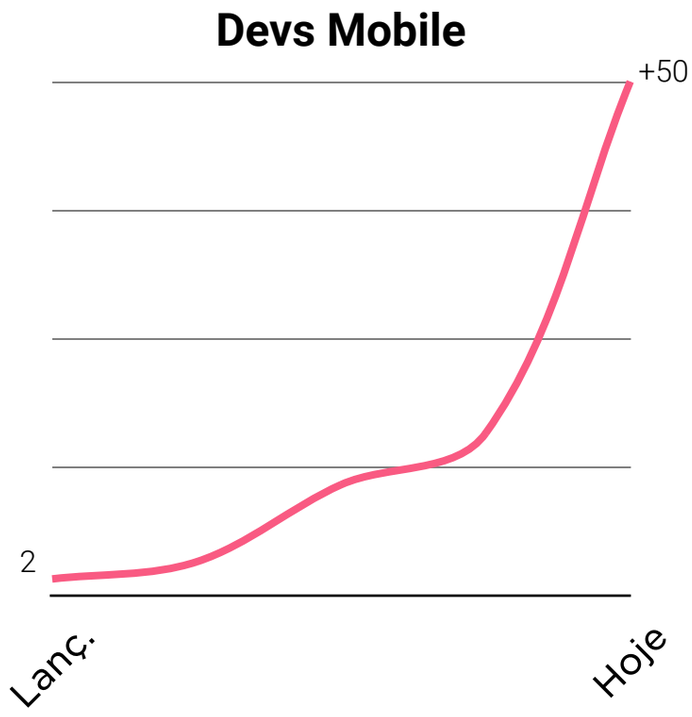
NuConta

O que enfrentamos?





O que enfrentamos?



Nós **escalamos**, nossa arquitetura não nos acompanhou

- Nossa equipe aumentou **exponencialmente**
- Necessário continuar criando e iterando com **agilidade e qualidade**

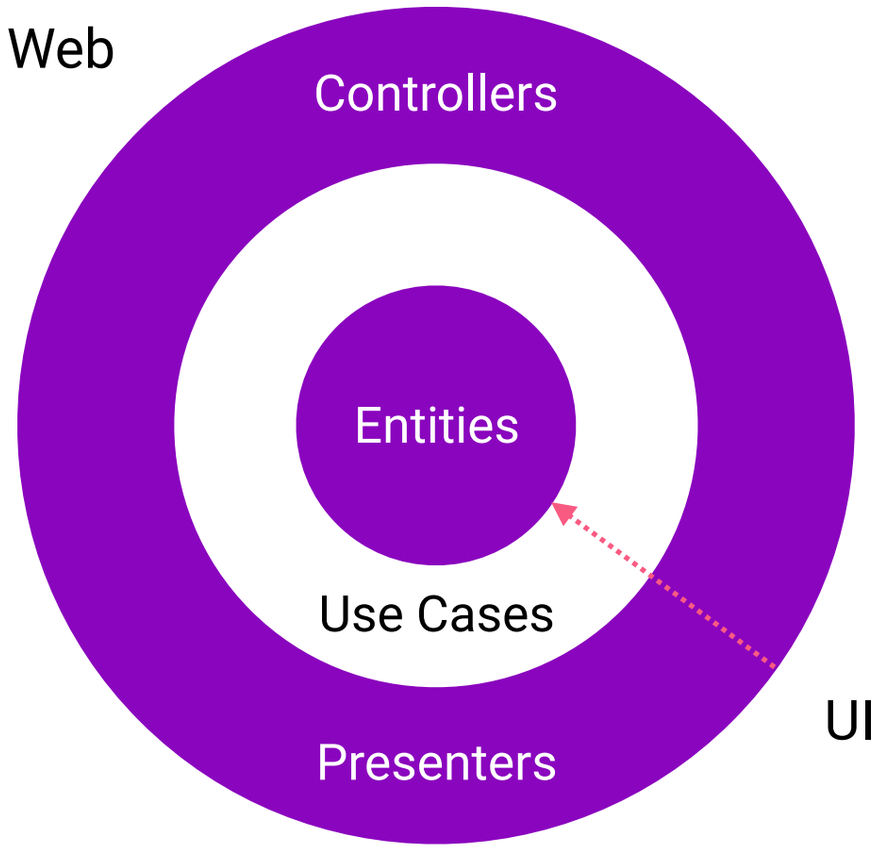
Swift e Kotlin

Decisão de migração em 2016

- Esta decisão foi o ponto de partida para estudar soluções de arquitetura
- Intenção de adotar uma arquitetura para ambas as plataformas

Planejamento

Inspiração

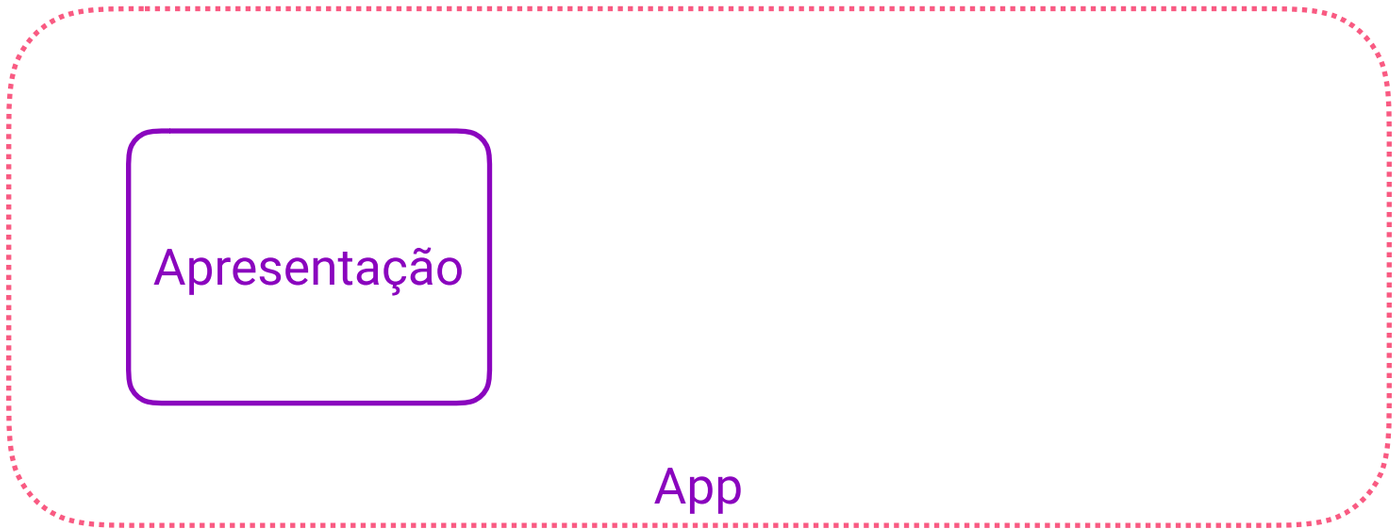


Uncle Bob's Clean Architecture

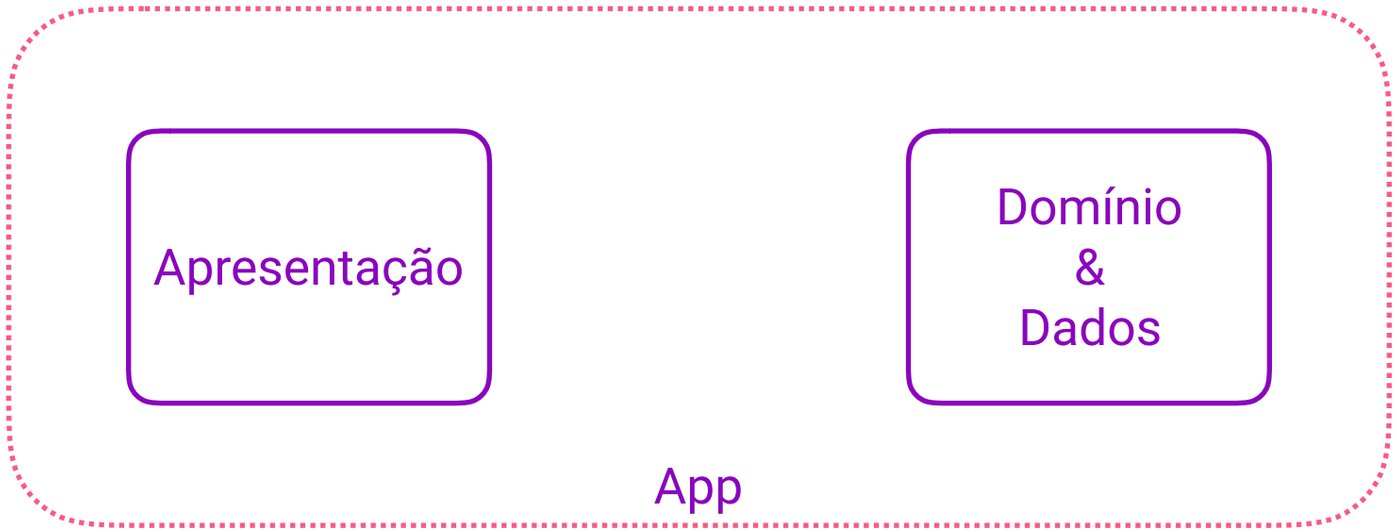
- Modularização e reuso
- Padronização
- Testes e manutenção

Como resolver?

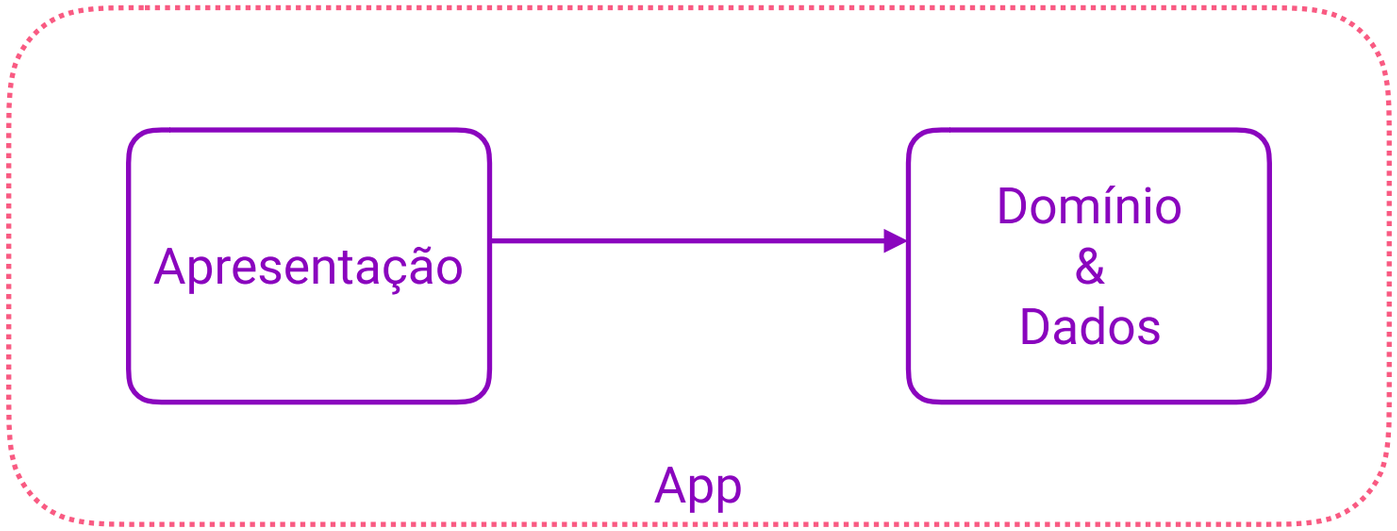
Como resolver?



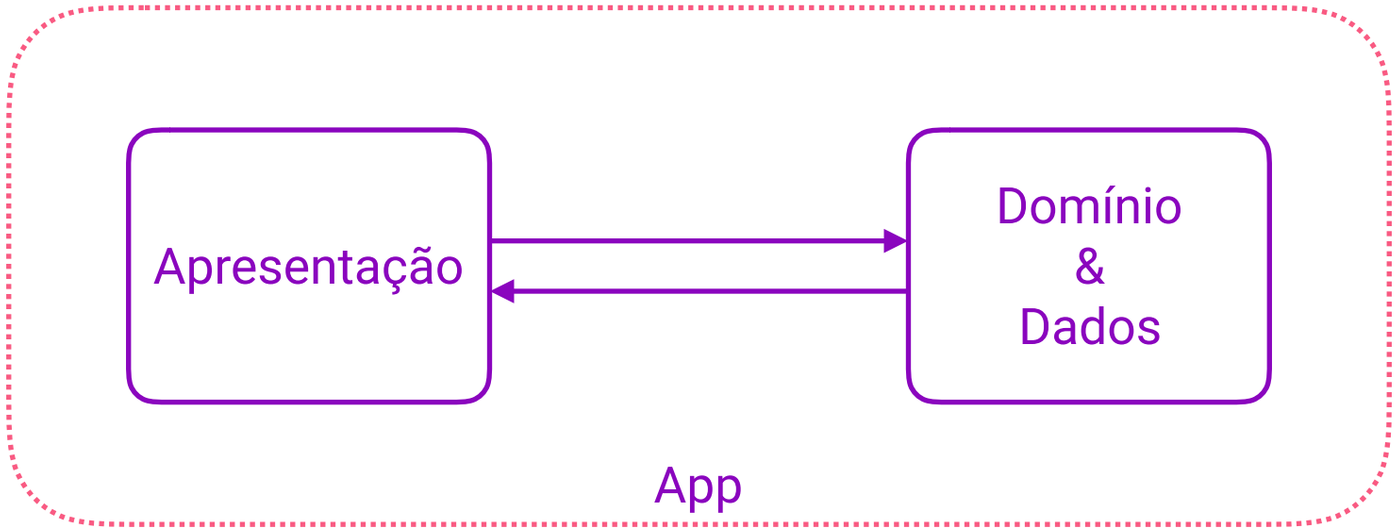
Como resolver?



Como resolver?



Como resolver?



Arquitetura de apresentação

Abstrações

UIKit nos oferece duas classes: **UIView** e **UIViewController**

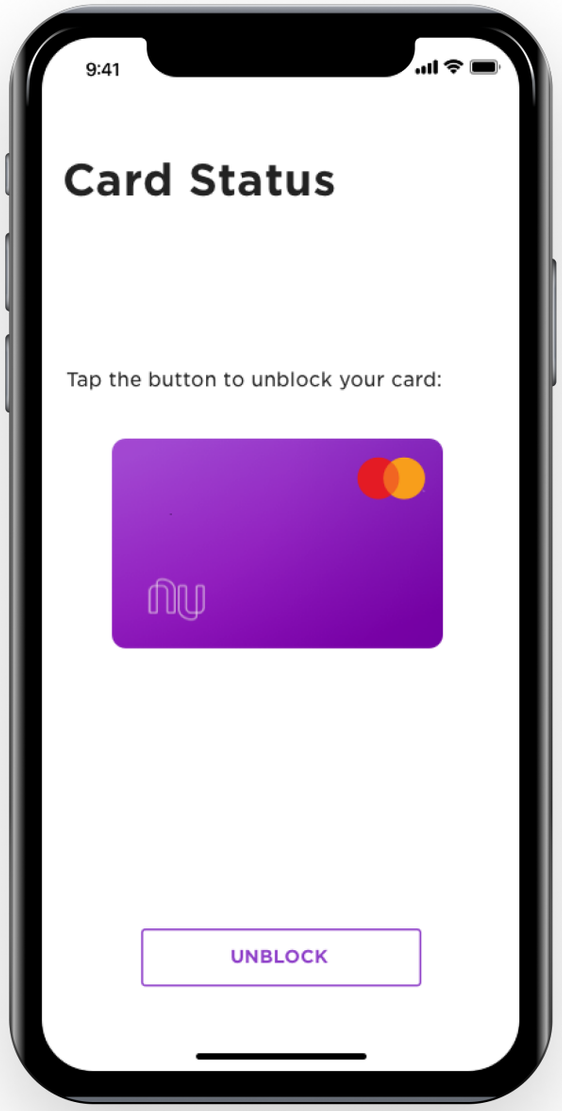
- **Responsabilidades** pouco definidas
- Necessário mais camadas de **abstração**

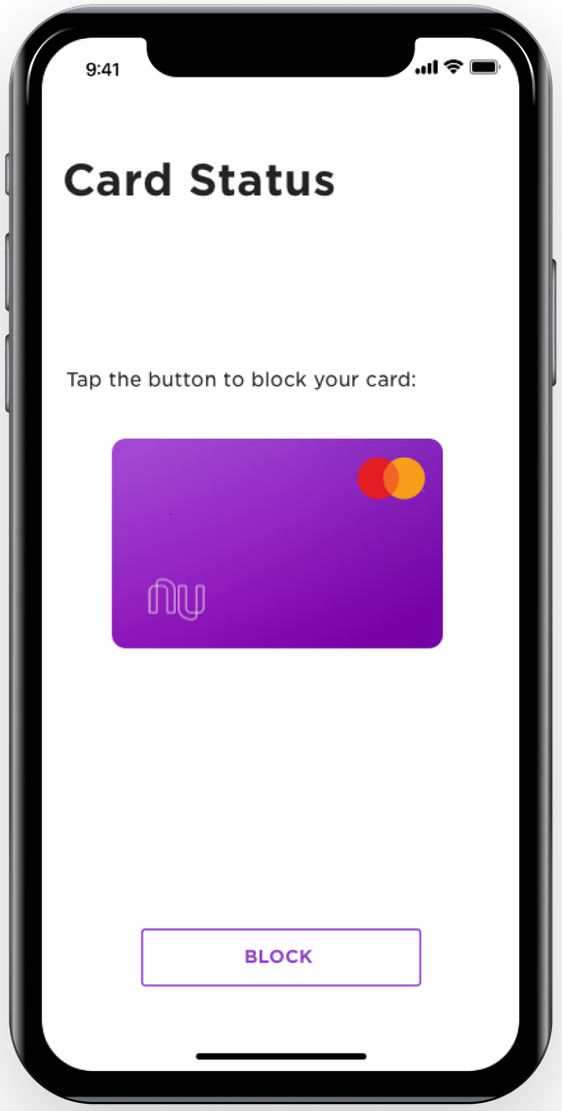
Abstrações

- Controller
- ViewController
- View
- ViewModel

Protótipo

Bloqueio e desbloqueio de cartão

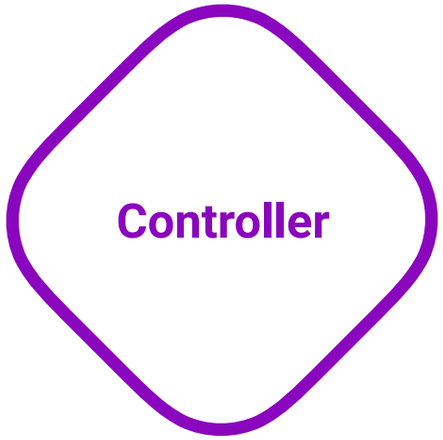


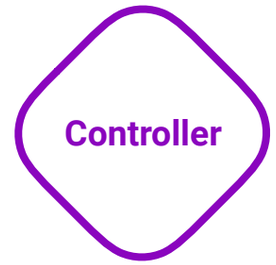


Protocolos

```
enum CardStatus {
    case active
    case blocked
}

protocol CardManagerProtocol {
    func toggleCardStatus() -> Completable
}
```

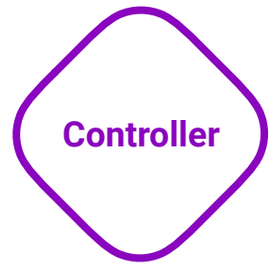
Controller





Controller

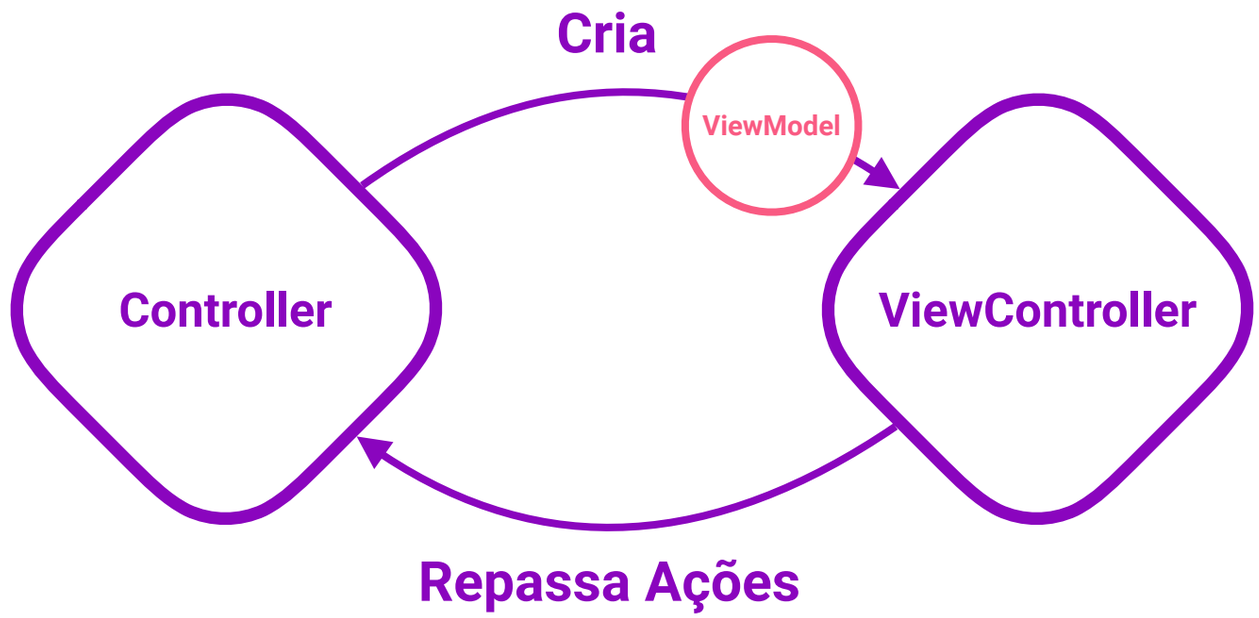
```
class Controller {
    enum Action {
        case toggleCardStatus
    }
    let viewController: ViewController
    let cardManager: CardManager
    lazy var action: Observable<Action> = {
        return self.viewController.toggleCardStatus.map { .toggleCardStatus }
    }()

    init(status: Observable<CardStatus>,
         cardManager: CardManager = .init(),
         viewController: ViewController = .init()) {
        self.viewController = viewController
        self.cardManager = cardManager

        subscribe(toStatus: status)
        subscribe(toAction: action)
    }
```

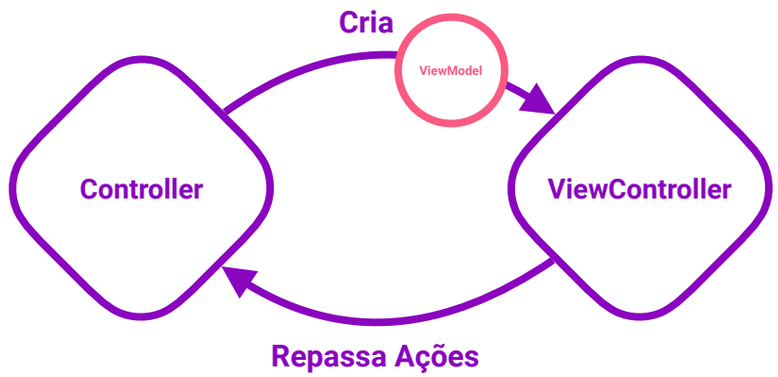


Controller

```
    func subscribe(toStatus cardStatus: Observable<CardStatus>) {
        cardStatus
            .subscribe(onNext: { [viewController] in
                let viewModel = ViewModel(cardStatus: $0)
                viewController.bind(viewModel)
            }).disposed(by: disposeBag)
    }

    func subscribe(toAction action: Observable<Action>) {
        action
            .flatMap { [cardManager] action -> Completable in
                switch action {
                case .toggleCardStatus: return cardManager.toggleCardStatus()
                }
            }.subscribe()
            .disposed(by: disposeBag)
    }
    let disposeBag = DisposeBag()
}
```

ViewController

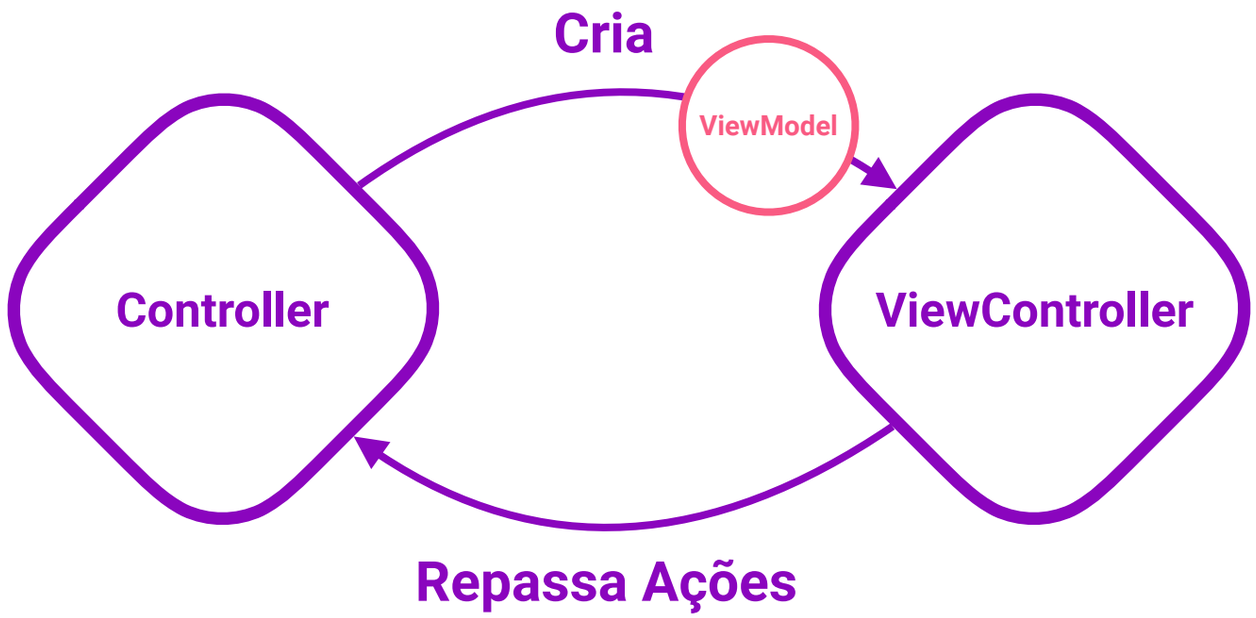


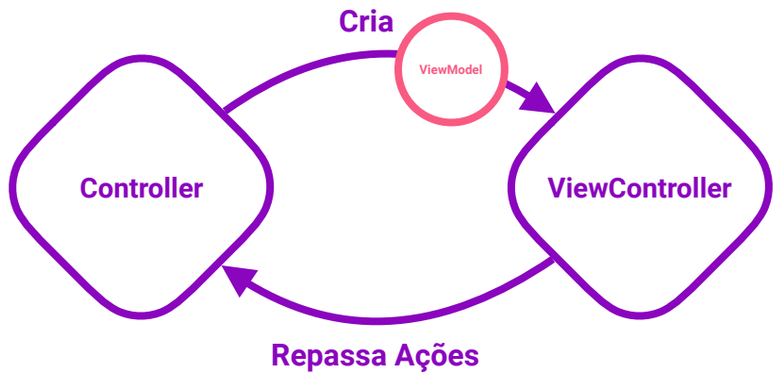


ViewController

```
class ViewController: UIViewController {
    lazy var typedView: View = {
        return view as! View
    }()

    lazy var toggleCardStatus: ControlEvent<Void> = {
        return typedView.button.rx.tap
    }()

    func bind(_ viewModel: ViewModel) {
        typedView.button.setTitle(viewModel.buttonTitle, for: .normal)
        typedView.titleLabel.text = viewModel.title
        typedView.hintLabel.text = viewModel.hint
        typedView.cardImageView.image = viewModel.cardImage
    }
}
```

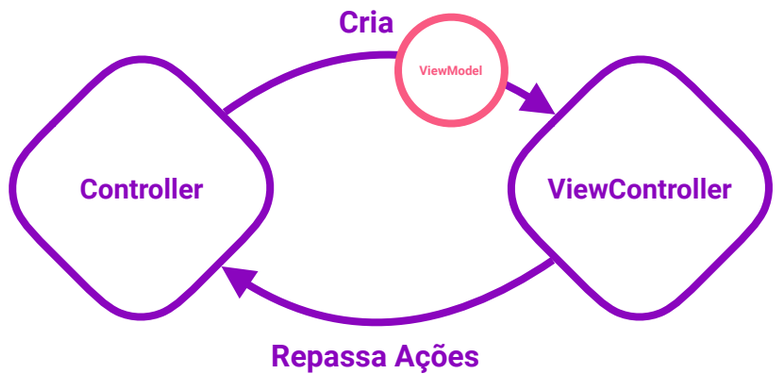
ViewModel





ViewModel

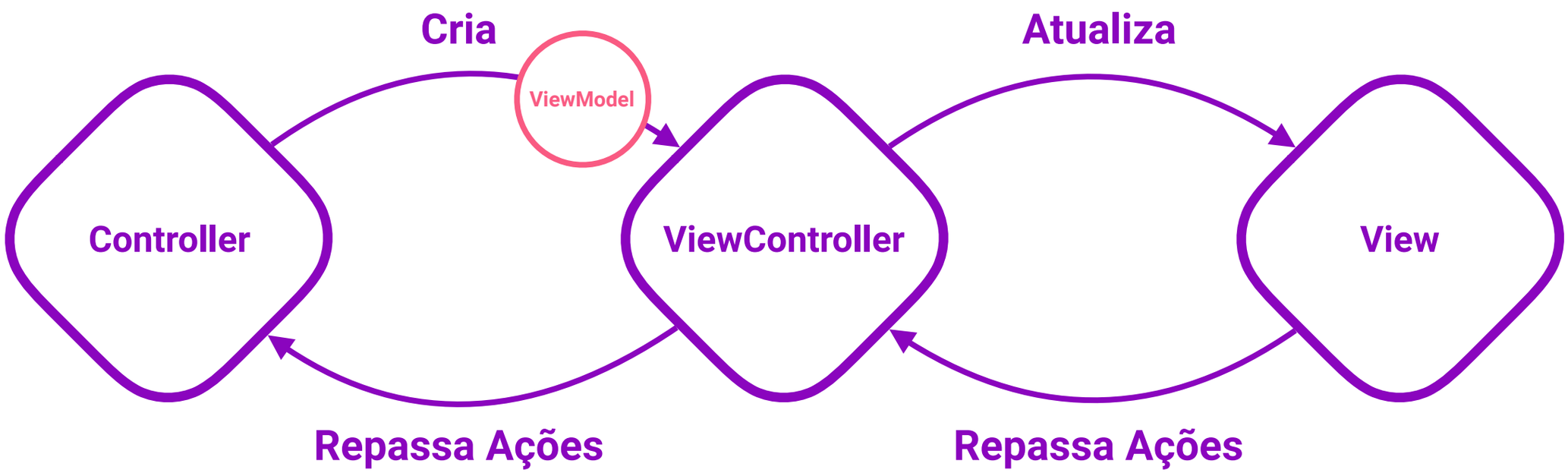
```
struct ViewModel: Equatable {
    let title: String = "Card Status"
    let buttonTitle: String
    let hint: String?
    let cardImage: UIImage?
}
```

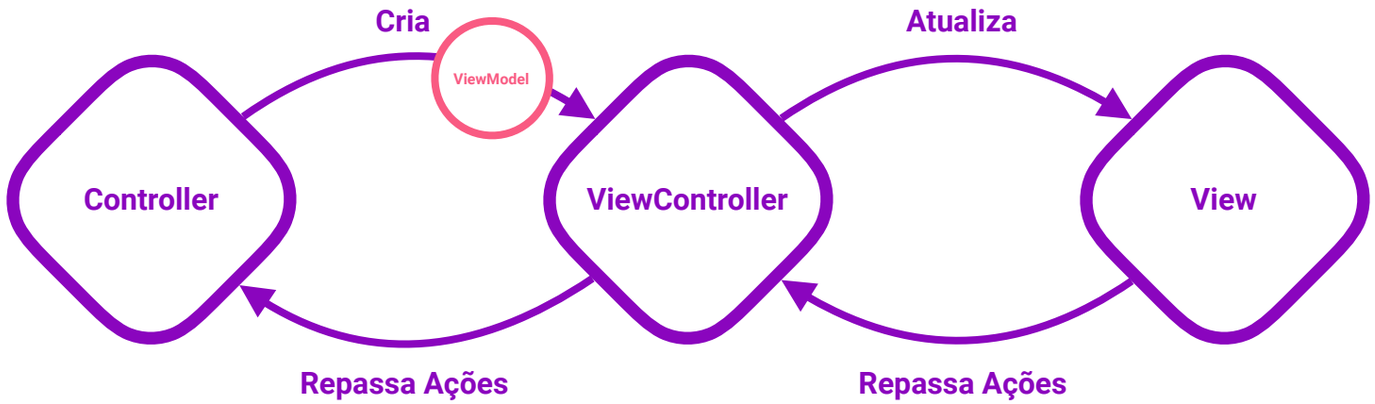


ViewModel

```
extension ViewModel {
    init(cardStatus: CardStatus) {
        cardImage = UIImage(named: "newCard")
        switch cardStatus {
        case .active:
            hint = "Tap the button to block your card"
            buttonTitle = "Block"
        case .blocked:
            hint = "Tap the button to unblock your card"
            buttonTitle = "Unblock"
        }
    }
}
```

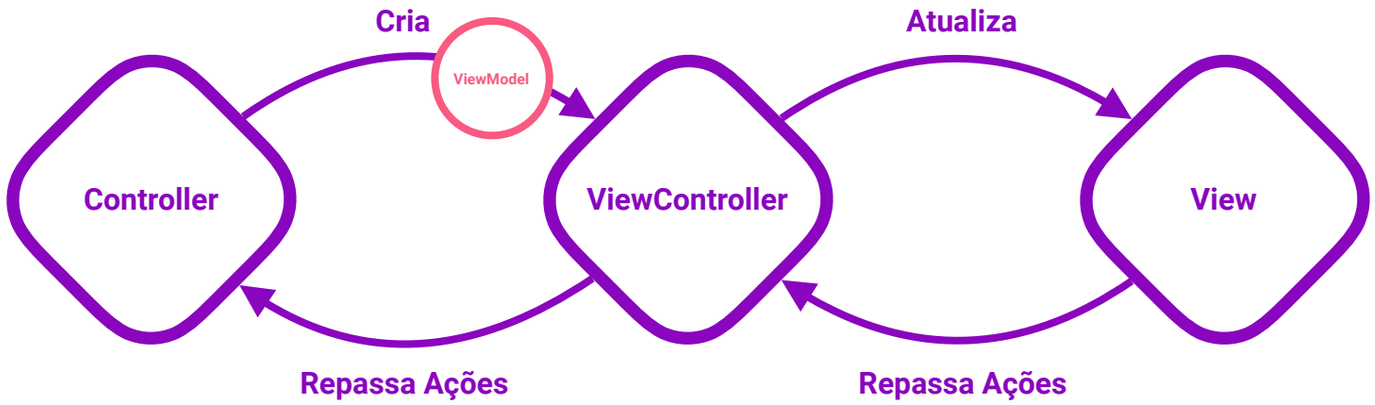
View


Cria
Atualiza
ViewModel
Controller
ViewController
View
Repassa Ações
Repassa Ações



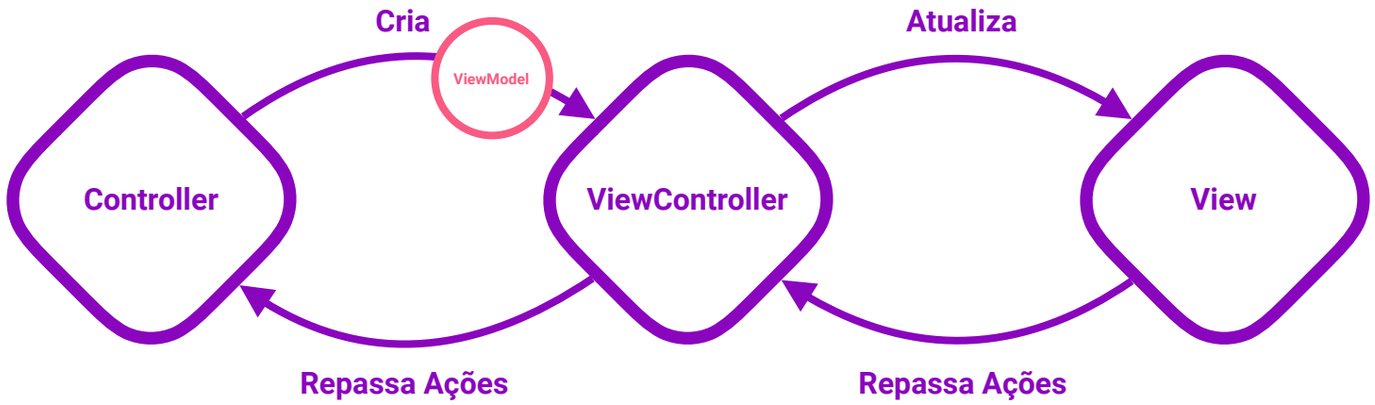
View

```
class View: BaseView {
    let button: UIButton = {
        let button = UIButton()
        button.backgroundColor = .purple
        button.setTitleColor(.white, for: .normal)
        button.setTitleColor(.lightGray, for: .disabled)
        return button
    }()
    let titleLabel: UILabel = {
        let label = UILabel()
        label.font = .boldSystemFont(ofSize: 24)
        return label
    }()
    let hintLabel: UILabel = {
        let label = UILabel()
        label.numberOfLines = 0
        return label
    }()
```



View

```
let cardImageView: UIImageView = {
    let imageView = UIImageView()
    imageView.contentMode = .scaleAspectFit
    return imageView
}()

override func initialize() {
    backgroundColor = .white
    addSubview(button)
    addSubview(titleLabel)
    addSubview(hintLabel)
    addSubview(cardImageView)
}

override func installConstraints() {
    button.snp.makeConstraints {
        $0.left.bottom.right.equalToSuperview().inset(20)
    }
```

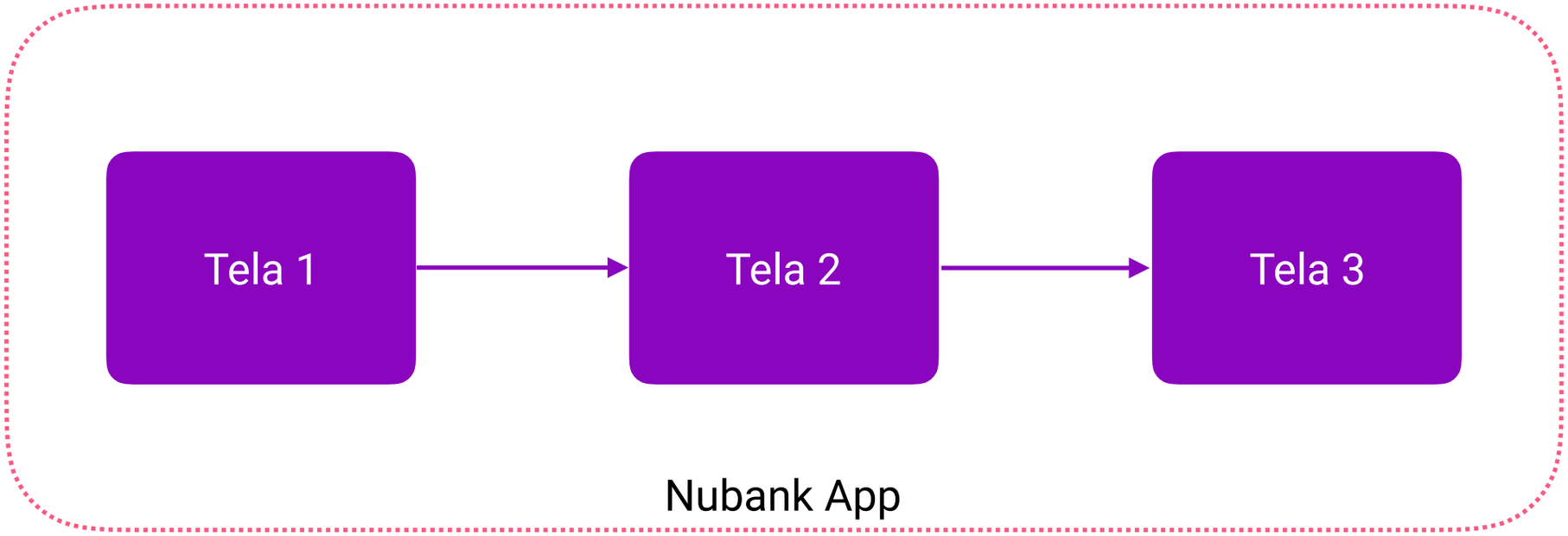


View

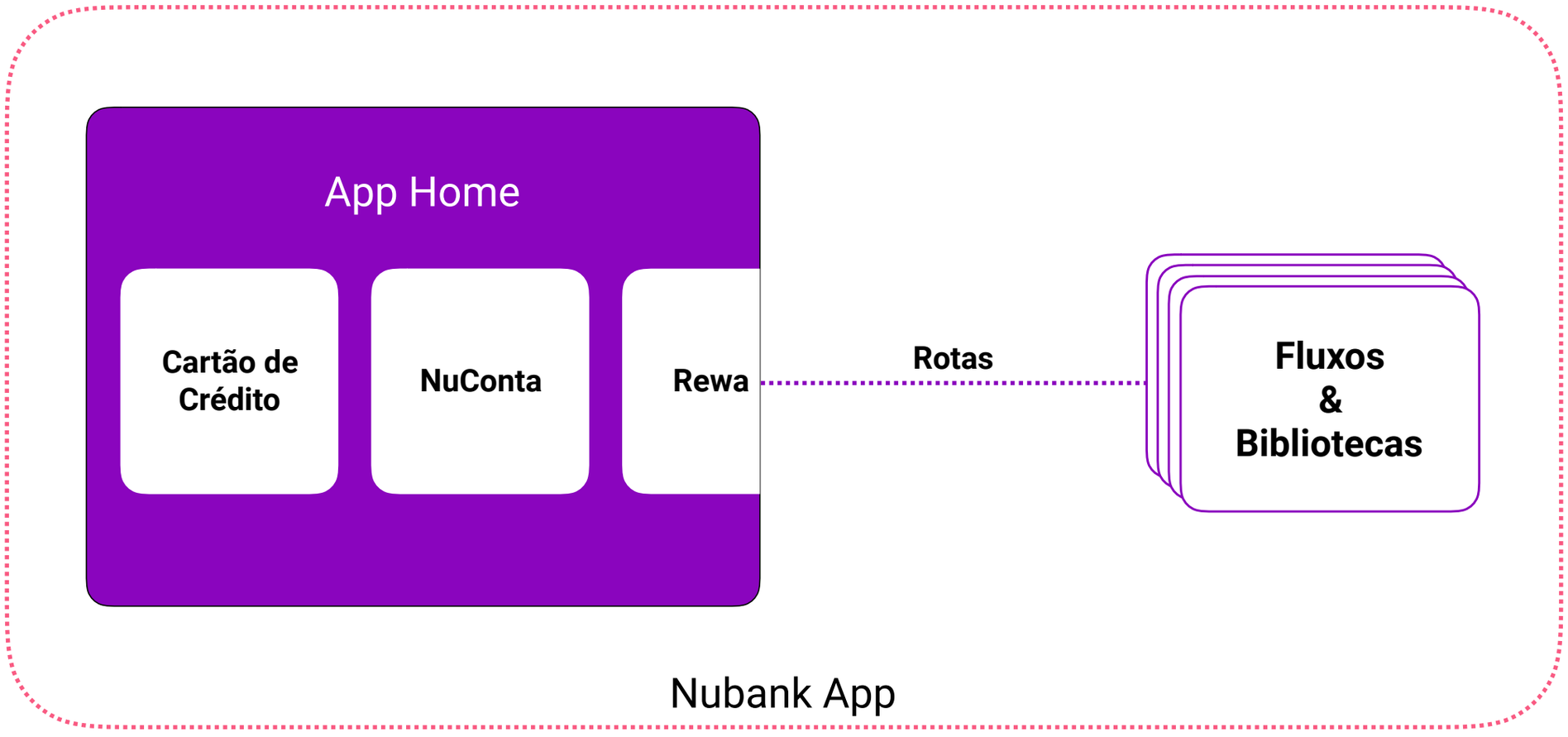
```
        titleLabel.snp.makeConstraints {
            $0.left.right.equalToSuperview().inset(20)
            $0.top.equalToSuperview().inset(20)
        }

        hintLabel.snp.makeConstraints {
            $0.left.right.equalTo(titleLabel)
            $0.top.equalTo(titleLabel.snp.bottom).offset(20)
        }

        cardImageView.snp.makeConstraints {
            $0.top.equalTo(hintLabel.snp.bottom).offset(20)
            $0.left.right.equalToSuperview().inset(20)
        }
    }
}
```

Navegação

Rotas e Coordinators



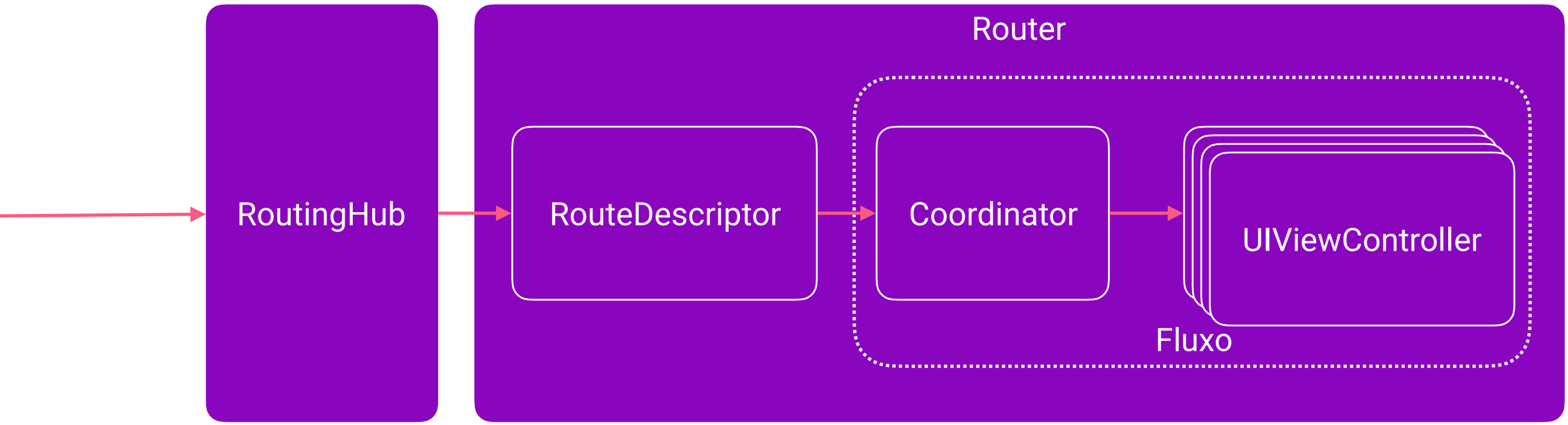
Rotas e Coordinators



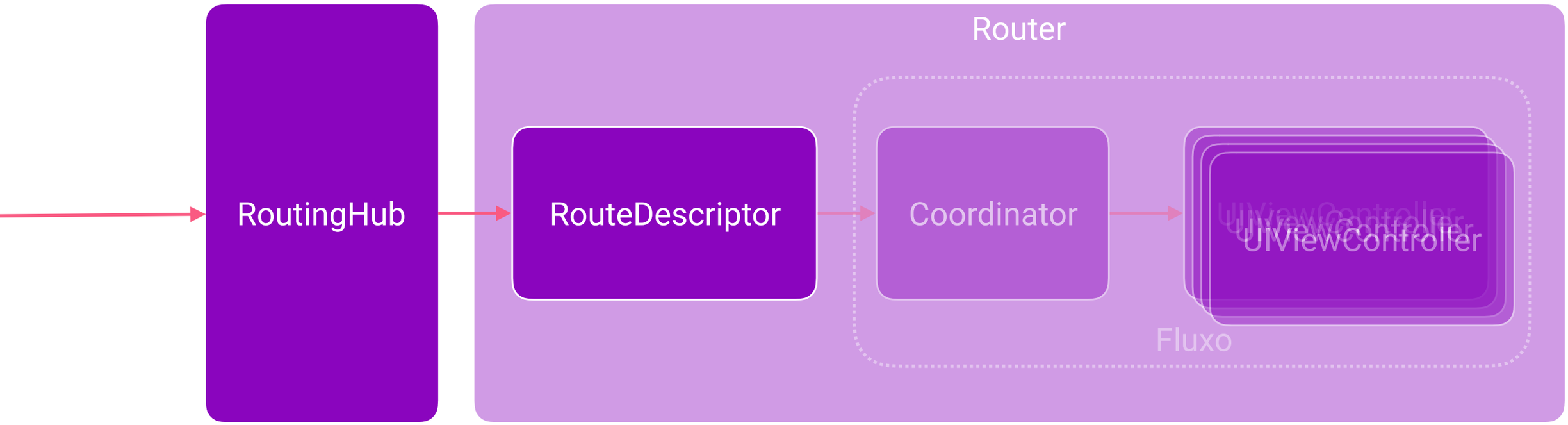
Rotas e Coordinators

- Um único ponto de entrada
- URLs para identificar fluxos
- Manter composição

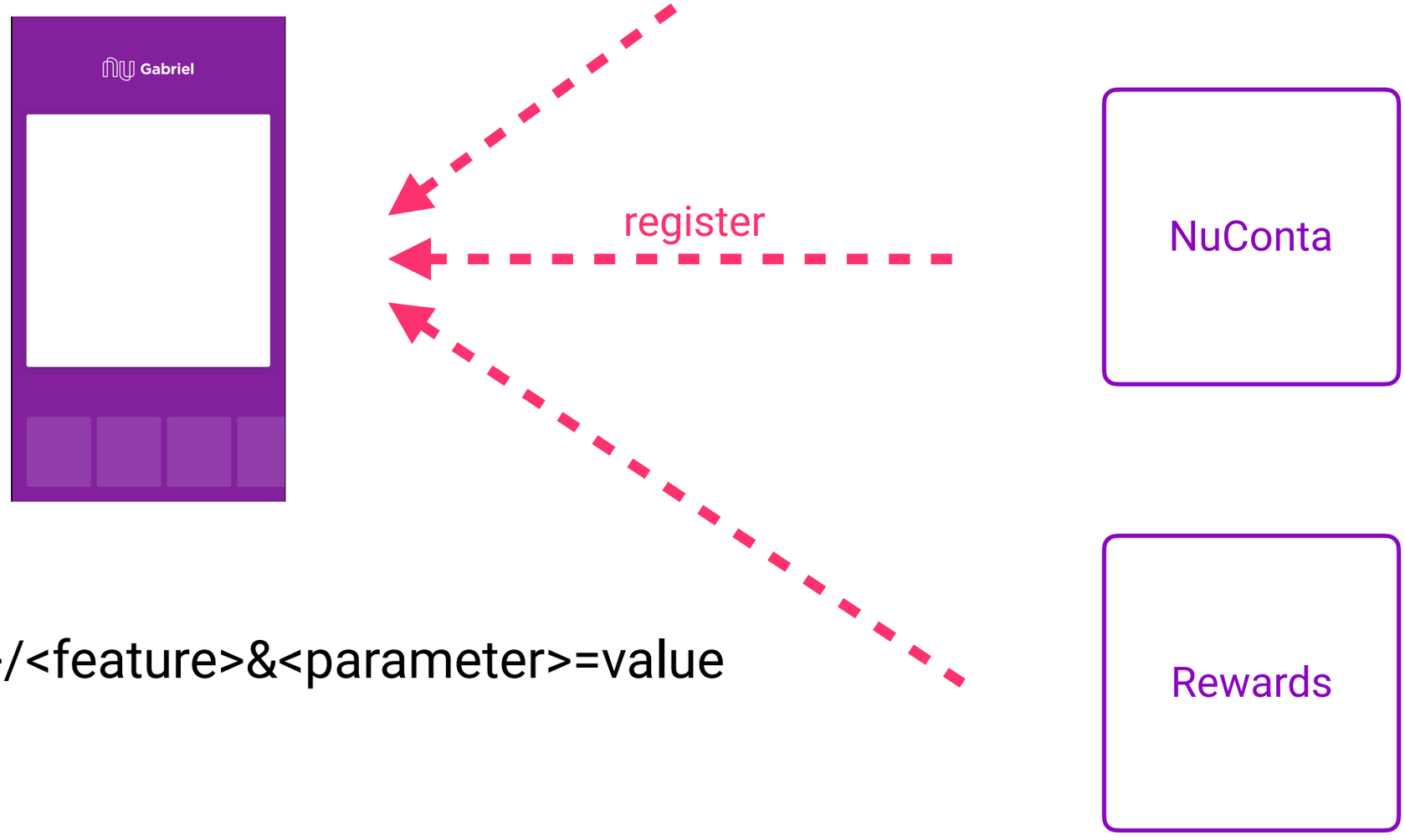
Rotas e Coordinators



Rotas

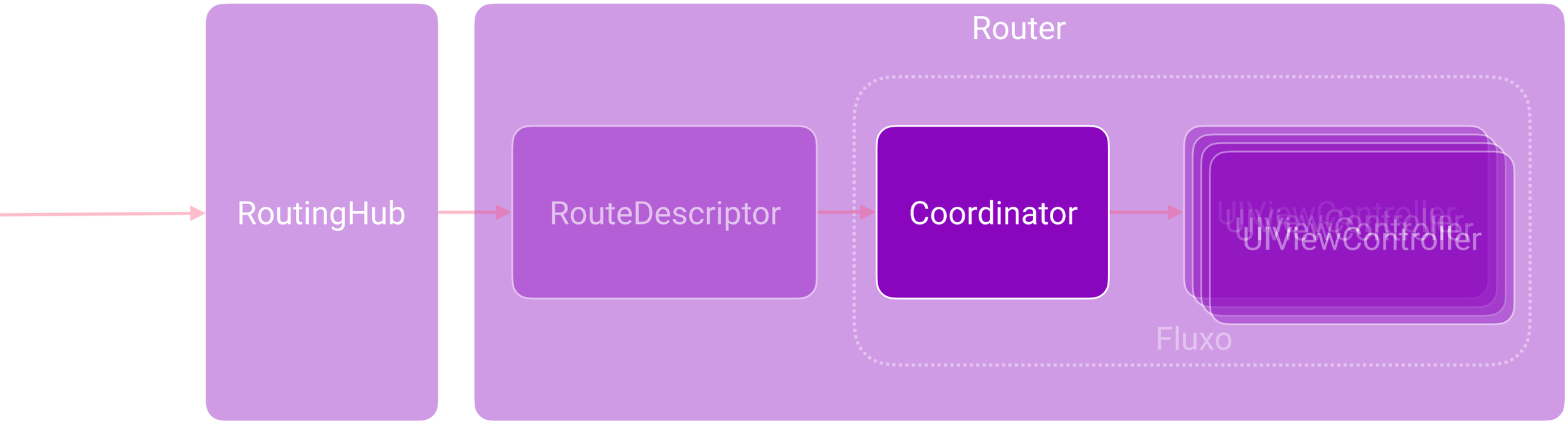

RoutingHub
Router
RouteDescriptor
Coordinator
UIViewController
UIViewController
UIViewController
Fluxo

Rotas



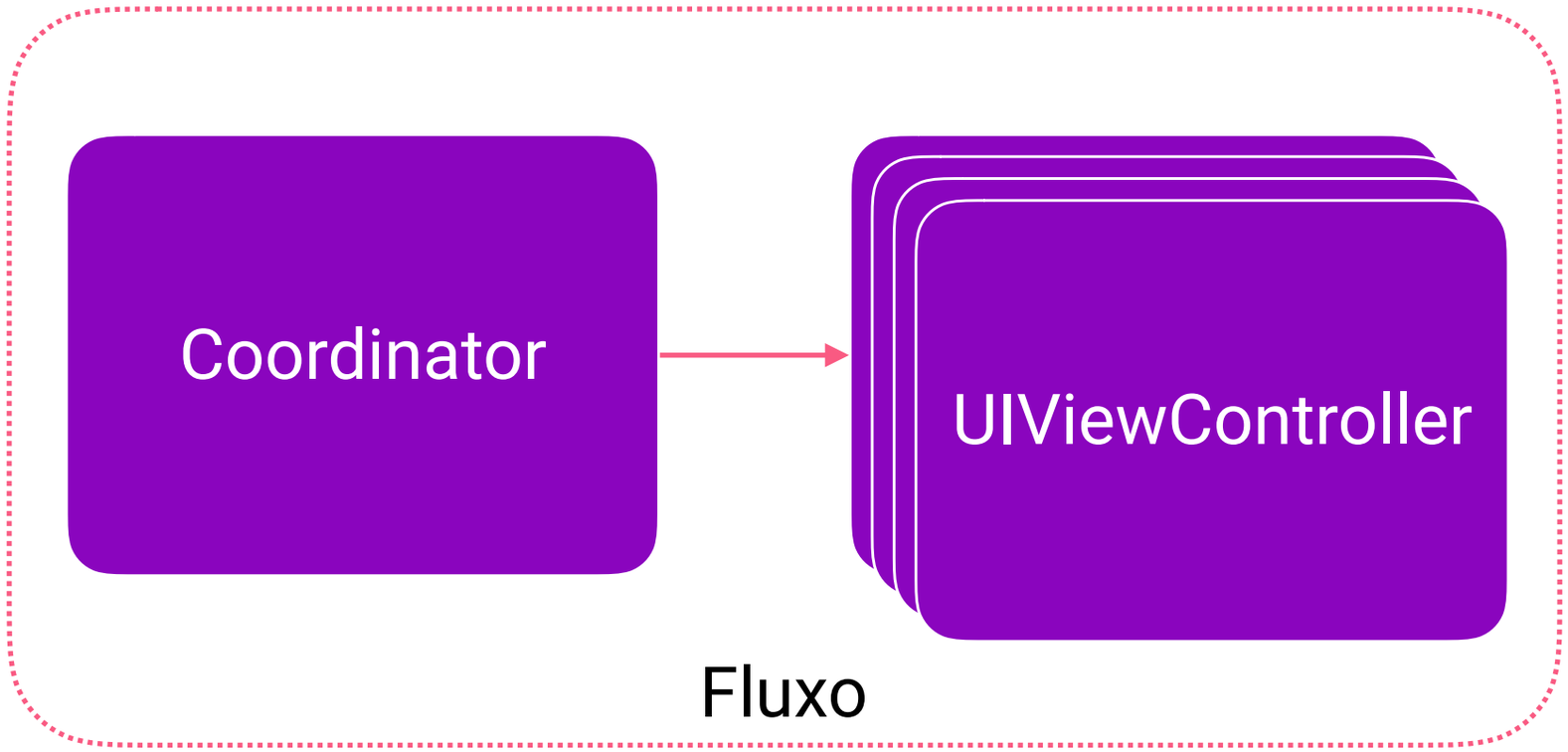
nuapp://<product>/<feature>&<parameter>=value

Coordinator



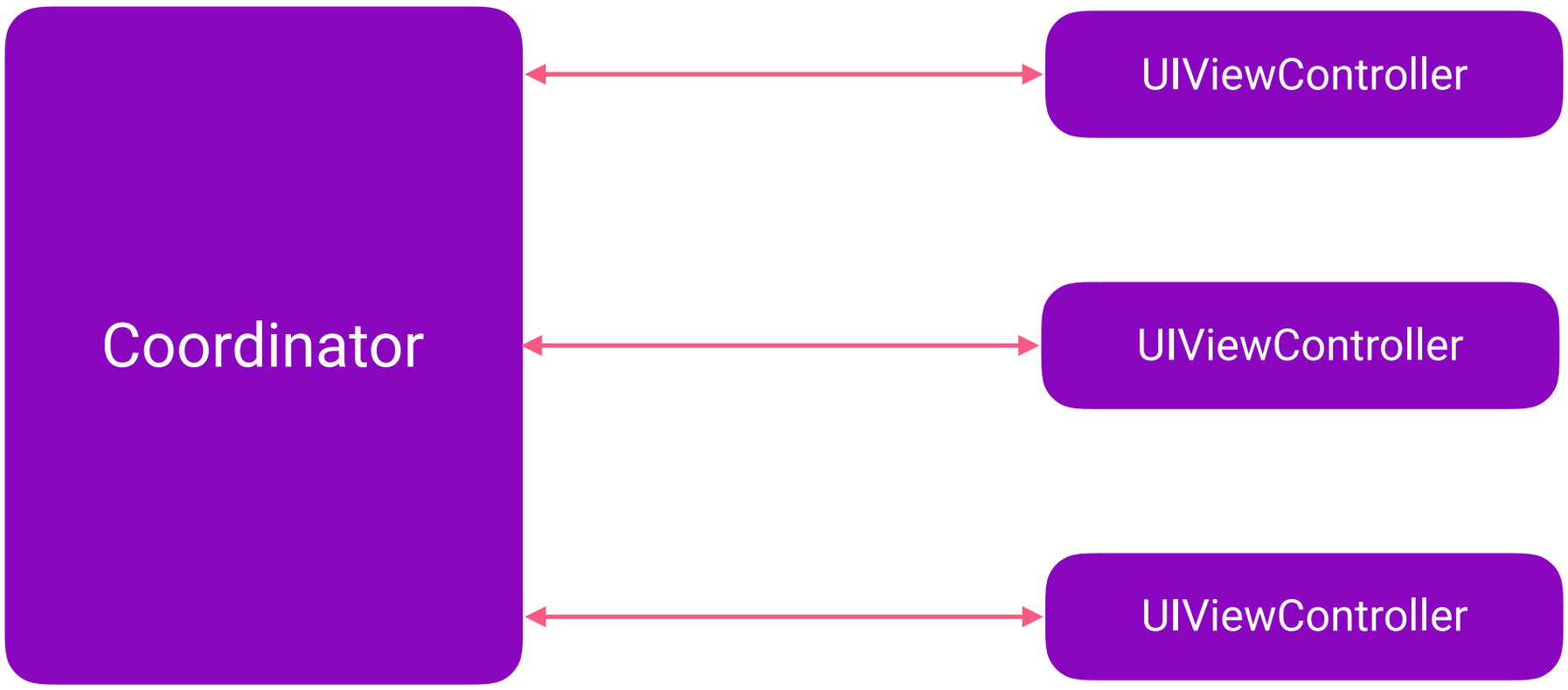
Coordinator

Unidade de abstração de um fluxo



Coordinator

Unidade de abstração de um fluxo



Perguntas?

Eduardo Pinto

@edulpn | eduardo.pinto@nubank.com.br

Piera Marchesini

@pieramarchesini | piera.marchesini@nubank.com.br

Obrigado!

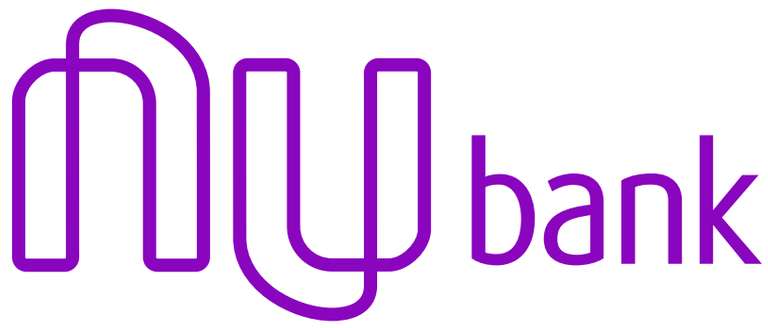
sou.nu/jobs-at-nubank

Eduardo Pinto

@edulpn | eduardo.pinto@nubank.com.br

Piera Marchesini

@pieramarchesini | piera.marchesini@nubank.com.br


bank