FLOW COORDINATOR

Alexandre Tavares, iOS Developer at STRV

STRV

iOS APPS

Views

ViewModel Bindings

Animations

Data Manipulation

User Interaction

Networking

Navigation

Storage

iOS APPS

Views

ViewModel Bindings

Animations

Data Manipulation

User Interaction

Networking

Navigation

Storage

Model

View

Controller

iOS APPS

Model

Data Manipulation

View

Views

Animations

Controller

ViewModel Bindings

User Interaction

Networking

Navigation

Storage

iOS APPS

Model
Data Manipulation

View
Views

Animations

Controller
ViewModel Bindings

User Interaction

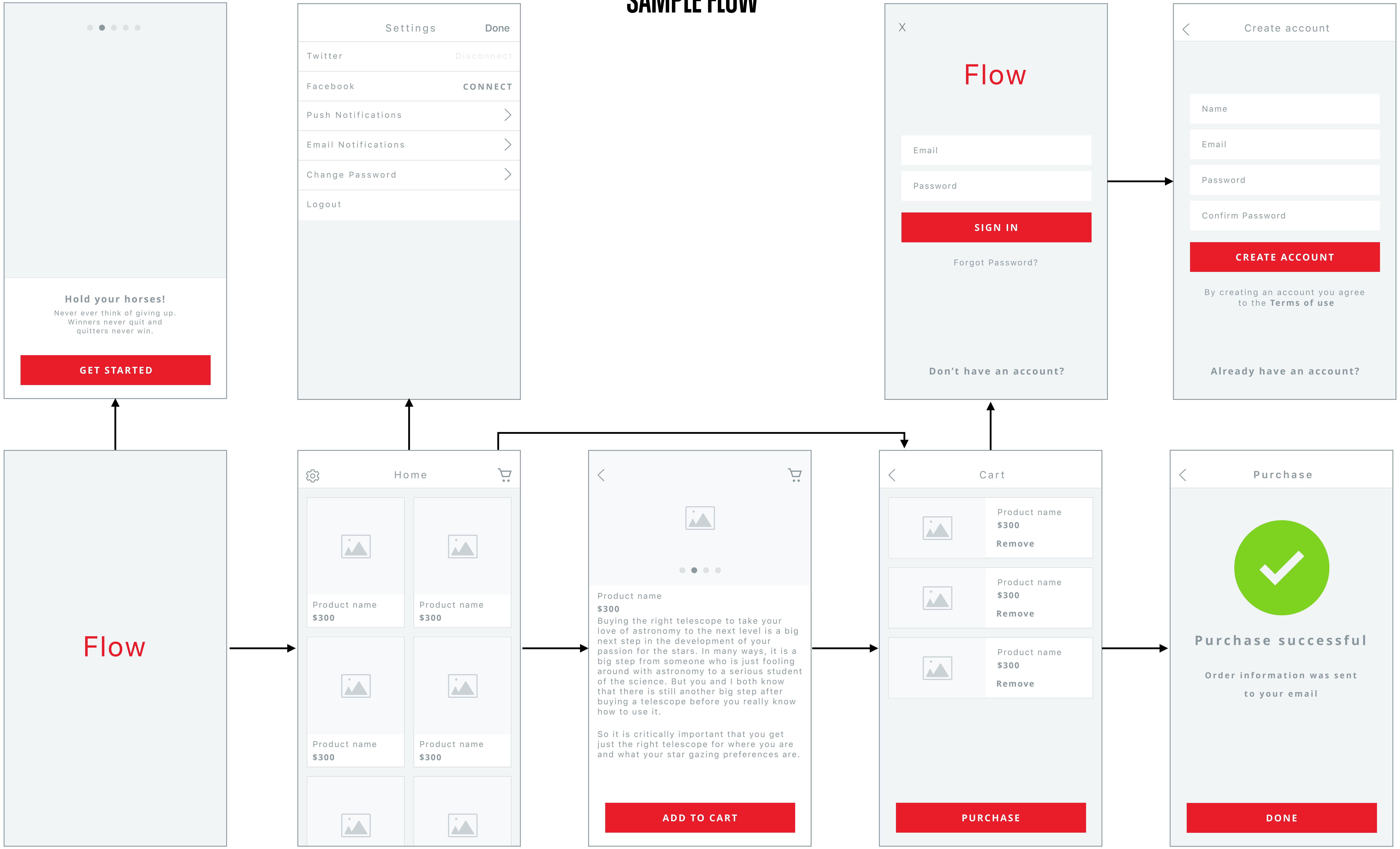
Navigation

Services
Networking

Storage

Navigation

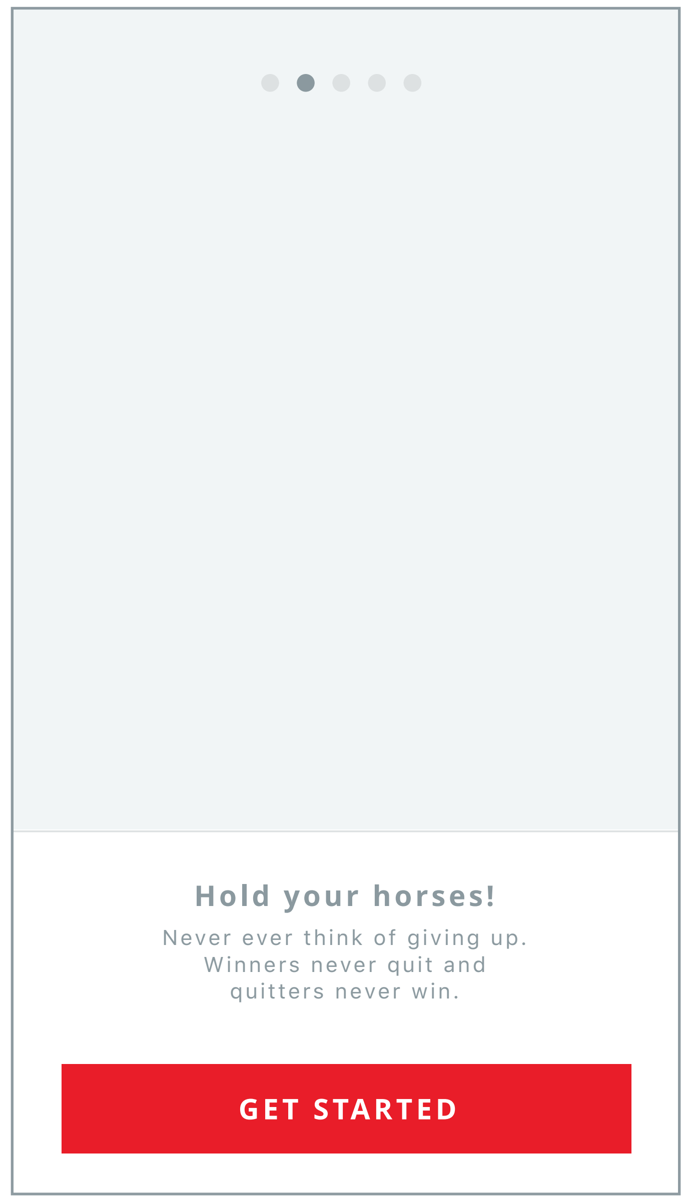
SAMPLE FLOW



Buying the right telescope to take your love of astronomy to the next level is a big next step in the development of your passion for the stars. In many ways, it is a big step from someone who is just fooling around with astronomy to a serious student of the science. But you and I both know that there is still another big step after buying a telescope before you really know how to use it.

So it is critically important that you get just the right telescope for where you are and what your star gazing preferences are.

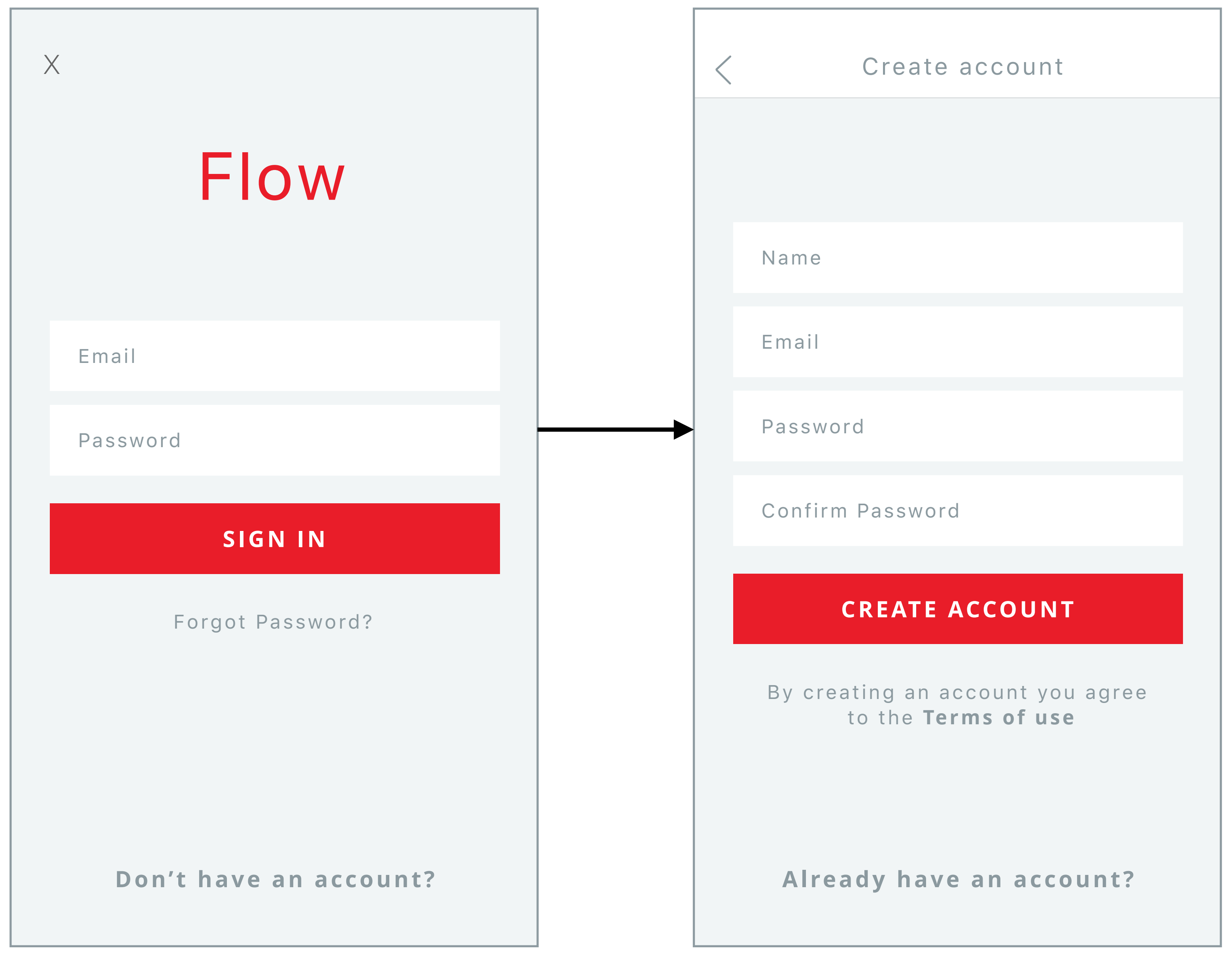
ONBOARDING FLOW



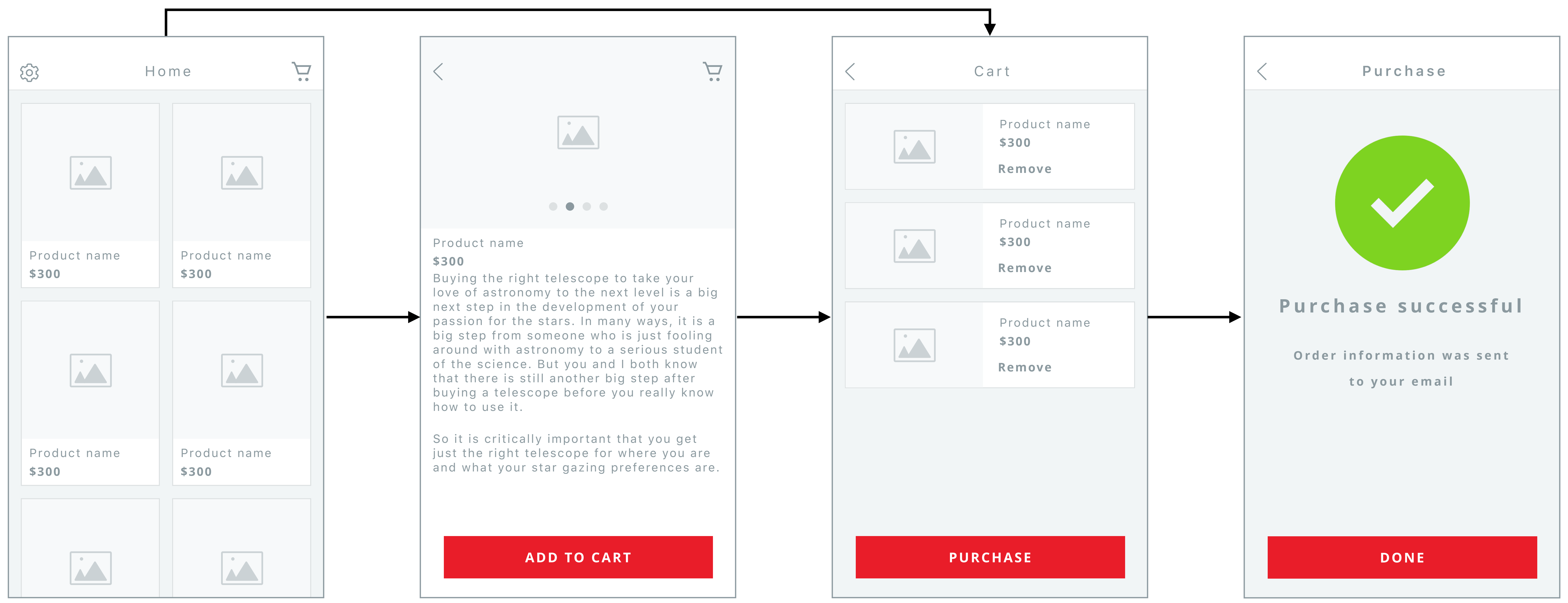
Hold your horses!

Never ever think of giving up.
Winners never quit and
quitters never win.

LOGIN FLOW


X
Flow
Email
Password
SIGN IN
Forgot Password?
Don't have an account?
Create account
Name
Email
Password
Confirm Password
CREATE ACCOUNT
By creating an account you agree to the Terms of use
Already have an account?

SHOPPING FLOW

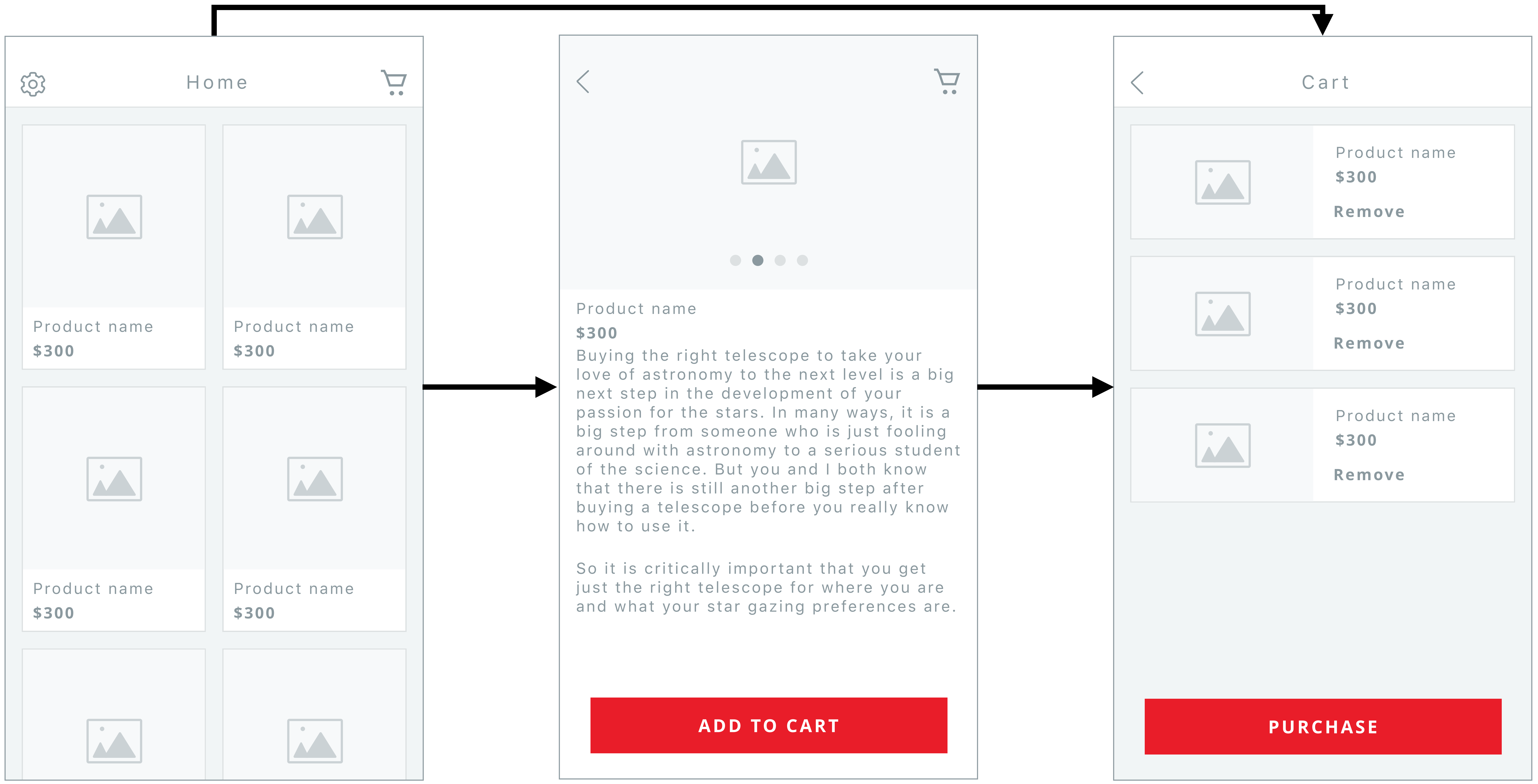


Buying the right telescope to take your love of astronomy to the next level is a big next step in the development of your passion for the stars. In many ways, it is a big step from someone who is just fooling around with astronomy to a serious student of the science. But you and I both know that there is still another big step after buying a telescope before you really know how to use it.

So it is critically important that you get just the right telescope for where you are and what your star gazing preferences are.

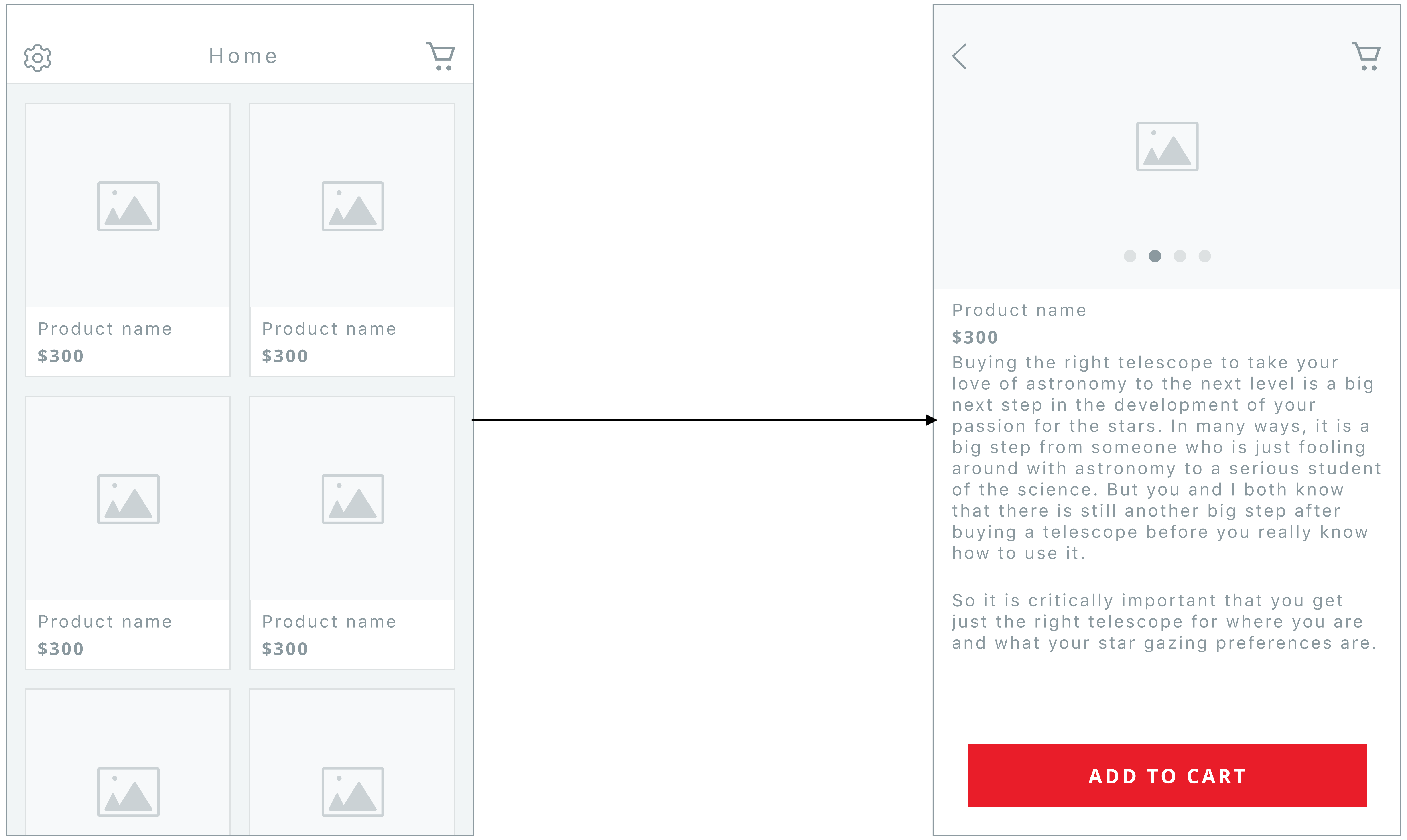
STRV

SHOPPING FLOW


Home
Product name
$300
Product name
$300
Product name
$300
Product name
$300
Product name
$300
Buying the right telescope to take your love of astronomy to the next level is a big next step in the development of your passion for the stars. In many ways, it is a big step from someone who is just fooling around with astronomy to a serious student of the science. But you and I both know that there is still another big step after buying a telescope before you really know how to use it.
So it is critically important that you get just the right telescope for where you are and what your star gazing preferences are.
ADD TO CART
Cart
Product name
$300
Remove
Product name
$300
Remove
Product name
$300
Remove
PURCHASE

STRV

SHOPPING FLOW



Buying the right telescope to take your love of astronomy to the next level is a big next step in the development of your passion for the stars. In many ways, it is a big step from someone who is just fooling around with astronomy to a serious student of the science. But you and I both know that there is still another big step after buying a telescope before you really know how to use it.

So it is critically important that you get just the right telescope for where you are and what your star gazing preferences are.

STRV

SHOPPING FLOW

```swift
func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {

    let item = dataSource.item(at: indexPath)

    let itemDetailViewController = ItemDetailViewController()
    itemDetailViewController.item = item

    self.navigationController?.pushViewController(itemDetailViewController,
                                                  animated: true)
}
```

SHOPPING FLOW

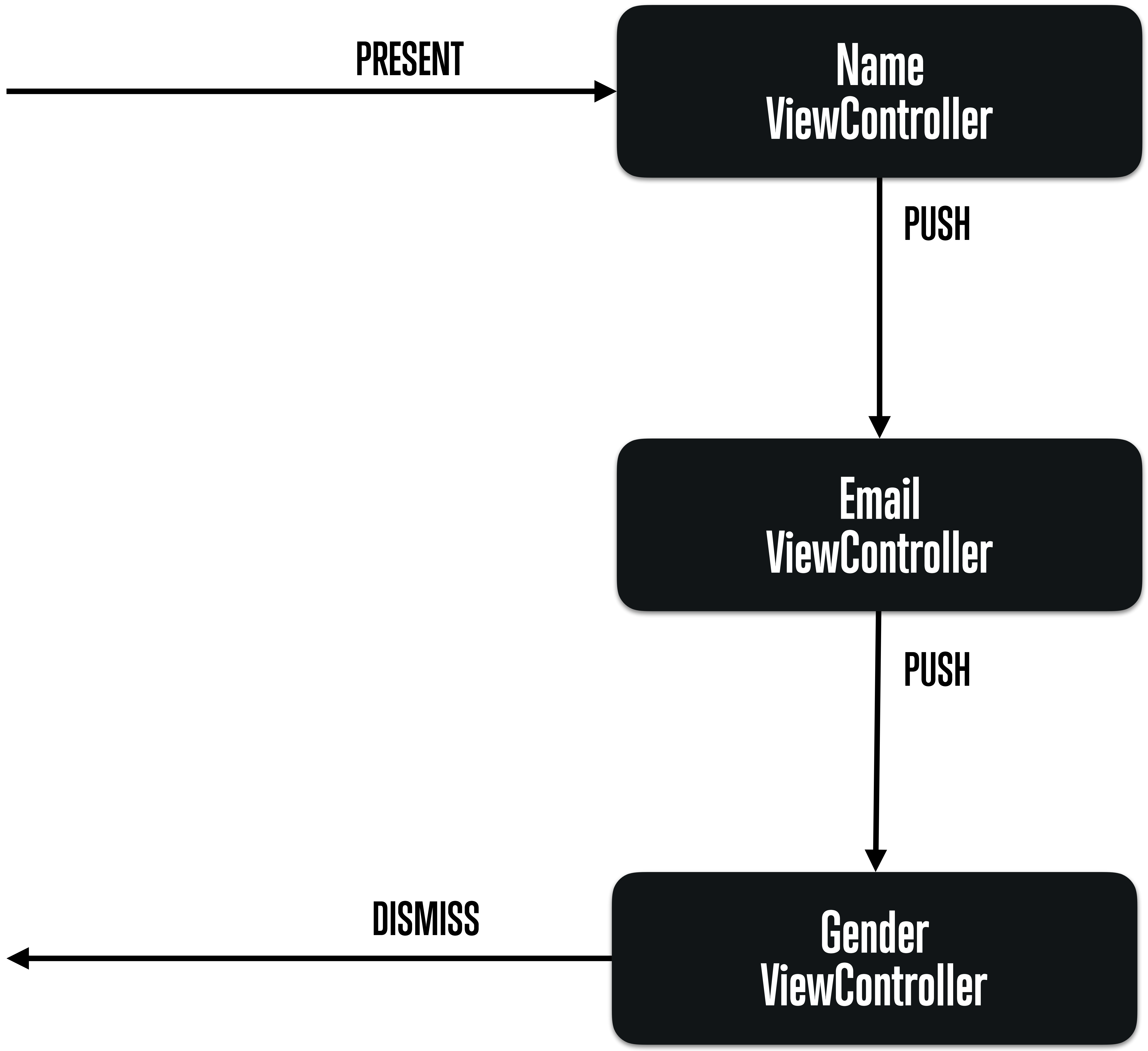
```swift
func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {

    let item = dataSource.item(at: indexPath)

    let itemDetailViewController = ItemDetailViewController()
    itemDetailViewController.item = item

    self.navigationController?.pushViewController(itemDetailViewController,
                                                   animated: true)
}
```

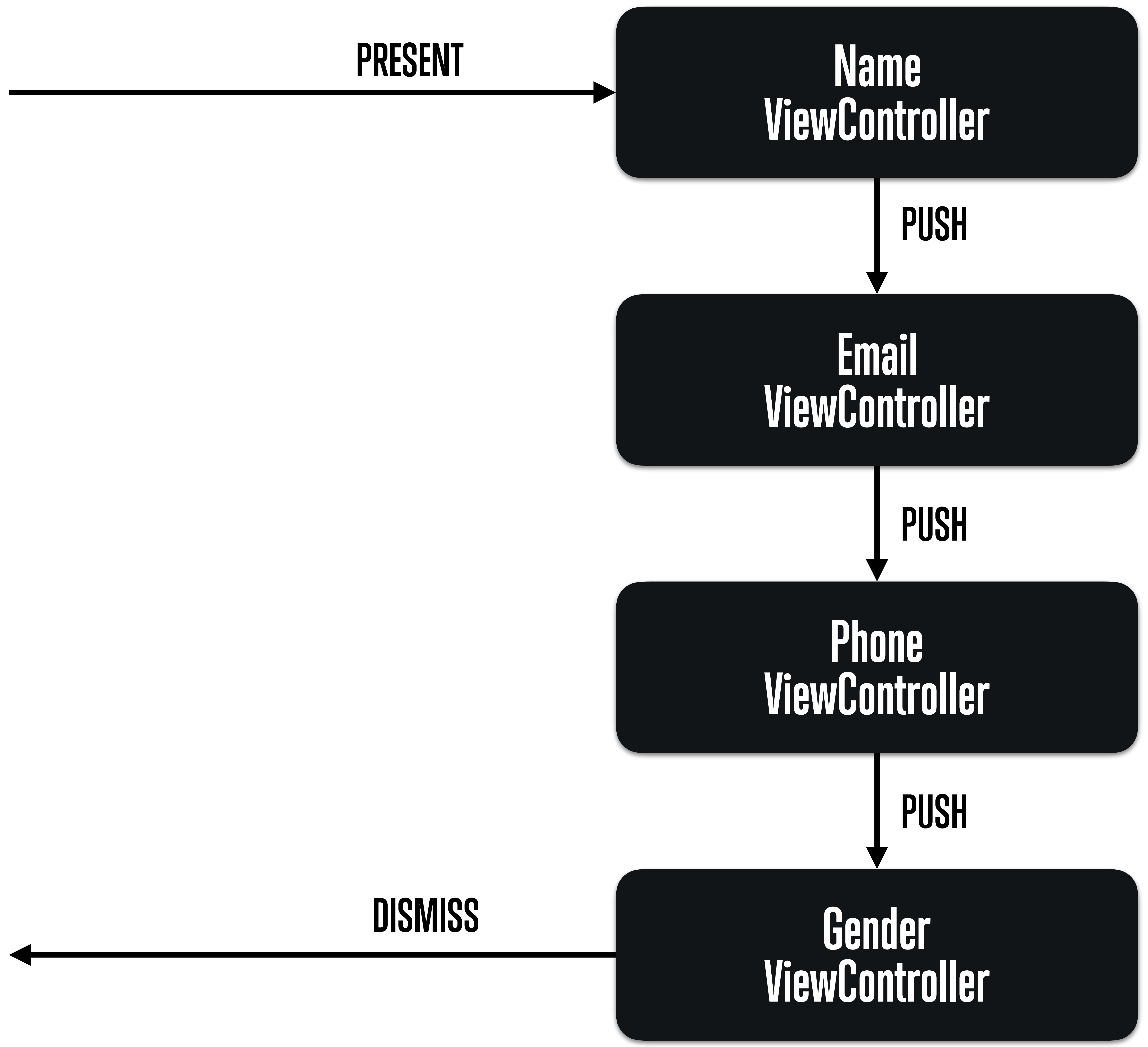
SHOPPING FLOW

```swift
func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {

    let item = dataSource.item(at: indexPath)

    let itemDetailViewController = ItemDetailViewController()
    itemDetailViewController.item = item

    self.navigationController?.pushViewController(itemDetailViewController,
                                                   animated: true)
}
```

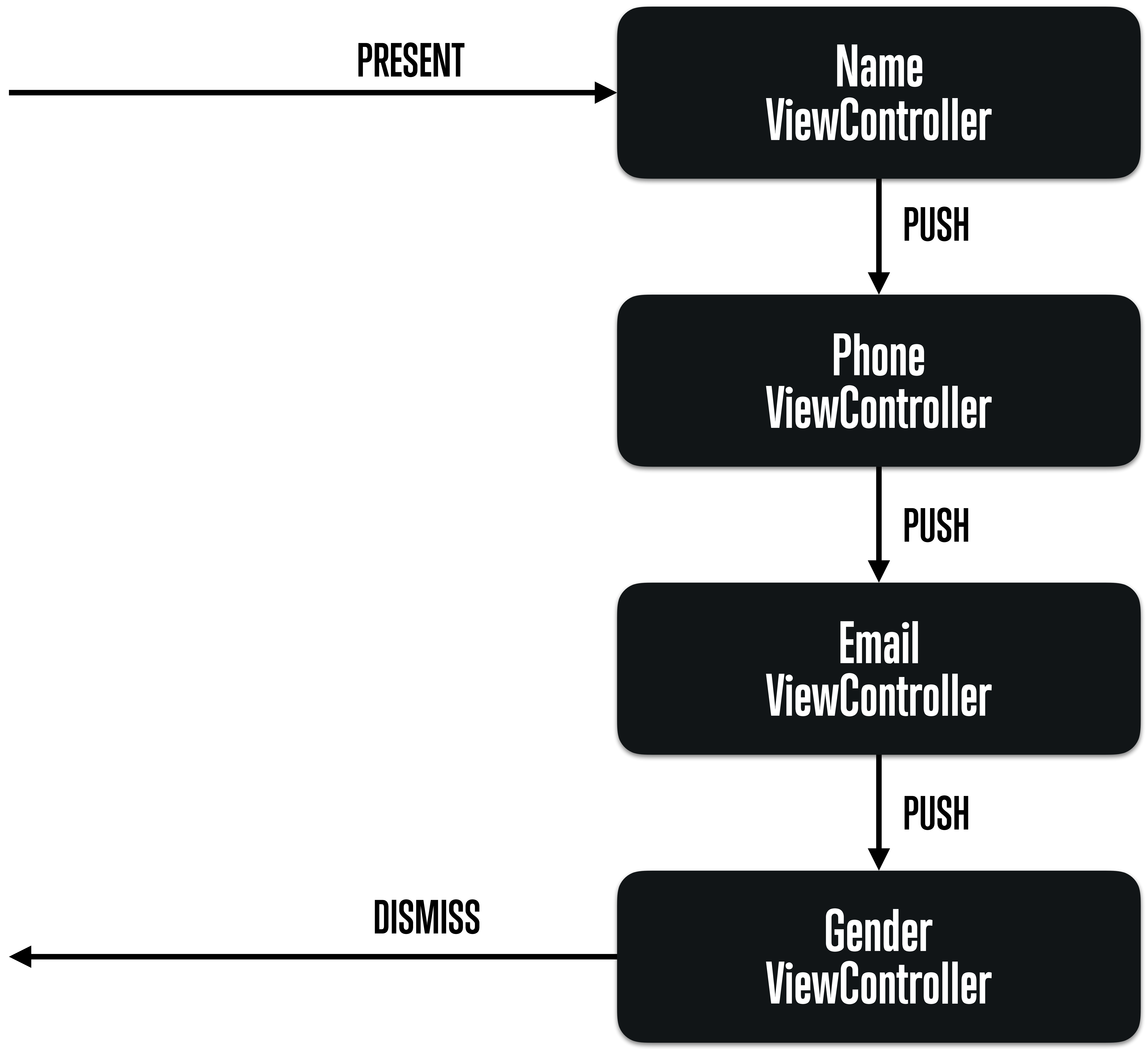
Children should not tell their parents what to do.

Most of the times, children shouldn't even know who their parents are.


PRESENT
Name ViewController
PUSH
Email ViewController
PUSH
Gender ViewController
DISMISS


PRESENT
Name ViewController
PUSH
Email ViewController
PUSH
Phone ViewController
PUSH
Gender ViewController
DISMISS

STRV


PRESENT
Name ViewController
PUSH
Phone ViewController
PUSH
Email ViewController
PUSH
Gender ViewController
DISMISS

WHAT IS A COORDINATOR?

COORDINATORS

- Responsible for the Application Flow
- Create, show and dismiss ViewControllers and Child Coordinators
- Removes responsibility from the ViewController
- Architecture agnostic

```swift
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?

    func application(_ application: UIApplication, didFinishLaunchingWithOptions
launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        let window = UIWindow(frame: UIScreen.main.bounds)
        self.window = window

        if Application.shouldShowOnboarding {
            window.rootViewController = OnboardingViewController()
        } else {
            let shoppingListViewController = ShoppingListViewController()
            shoppingListViewController.viewModel =  ShoppingListViewModel()
            window.rootViewController = shoppingListViewController
        }

        window.makeKeyAndVisible()
        return true
    }
}
```

```
class AppDelegate: UIResponder, UIApplicationDelegate {
    var window: UIWindow?
    var applicationCoordinator: FlowCoordinator?

    func application(_ application: UIApplication,
                     didFinishLaunchingWithOptions
      launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {

        let window = UIWindow(frame: UIScreen.main.bounds)
        self.window = window

        let applicationCoordinator = ApplicationCoordinator(window: window)
        self.window = window
        self.applicationCoordinator = applicationCoordinator
        applicationCoordinator.start()

        return true
    }
}
```

```swift
final class ApplicationCoordinator: FlowCoordinator {

    let window: UIWindow
    init(window: UIWindow) {
        self.window = window
        super.init()
    }

    var mainFlow: FlowCoordinator = ShoppingFlow()

    override func start() {
        self.root = mainFlow.root
        window.rootViewController = root

        showOnboardingFlowIfNeeded()

        window.makeKeyAndVisible()
    let window: UIWindow
    init(window: UIWindow) {
        self.window = window
        super.init()
    }

    var mainFlow: FlowCoordinator = ShoppingFlow()

    override func start() {
        self.root = mainFlow.root
        window.rootViewController = root

        showOnboardingFlowIfNeeded()

        window.makeKeyAndVisible()
    }

}
```

```
    let window: UIWindow
    init(window: UIWindow) {
        self.window = window
        super.init()
    }

    var mainFlow: FlowCoordinator = ShoppingFlow()

    override func start() {
        self.root = mainFlow.root
        window.rootViewController = root

        showOnboardingFlowIfNeeded()

        window.makeKeyAndVisible()
    }

}
```

STRV

```swift
protocol Coordinator {
    var children: [Coordinator] { get set }
    var root: UIViewController { get set }
    func start()
    func finish()
}
```

```swift
protocol Coordinator {
    var children: [Coordinator] { get set }
    var root: UIViewController { get set }
    func start()
    func finish()
}
```

```swift
protocol Coordinator {
    var children: [Coordinator] { get set }
    var root: UIViewController { get set }
    func start()
    func finish()
}
```

STRV

```swift
protocol Coordinator {
    var children: [Coordinator] { get set }
    var root: UIViewController { get set }
    func start()
    func finish()
}
```

COORDINATOR

- Using events

```
open class FlowCoordinator: Hashable {
    var children = Set<FlowCoordinator>()
    var didFinish = Signal<FlowCoordinator>()
    var root = UIViewController()
    func start() {}
}
```

COORDINATOR

- Using events

```
open class FlowCoordinator: Hashable {
    var children = Set<FlowCoordinator>()
    var didFinish = Signal<FlowCoordinator>()
    var root = UIViewController()
    func start() {}
}
```

Events

```
class Signal<EventType> {
    typealias EventHandler = ((EventType) -> Void)

    func emit(_ event: EventType) { ... }

    func on(_ handler: @escaping EventHandler) { ... }

}
```

Events

```
class Signal<EventType> {
    typealias EventHandler = ((EventType) -> Void)

    func emit(_ event: EventType) { ... }

    func on(_ handler: @escaping EventHandler) { ... }

}
```

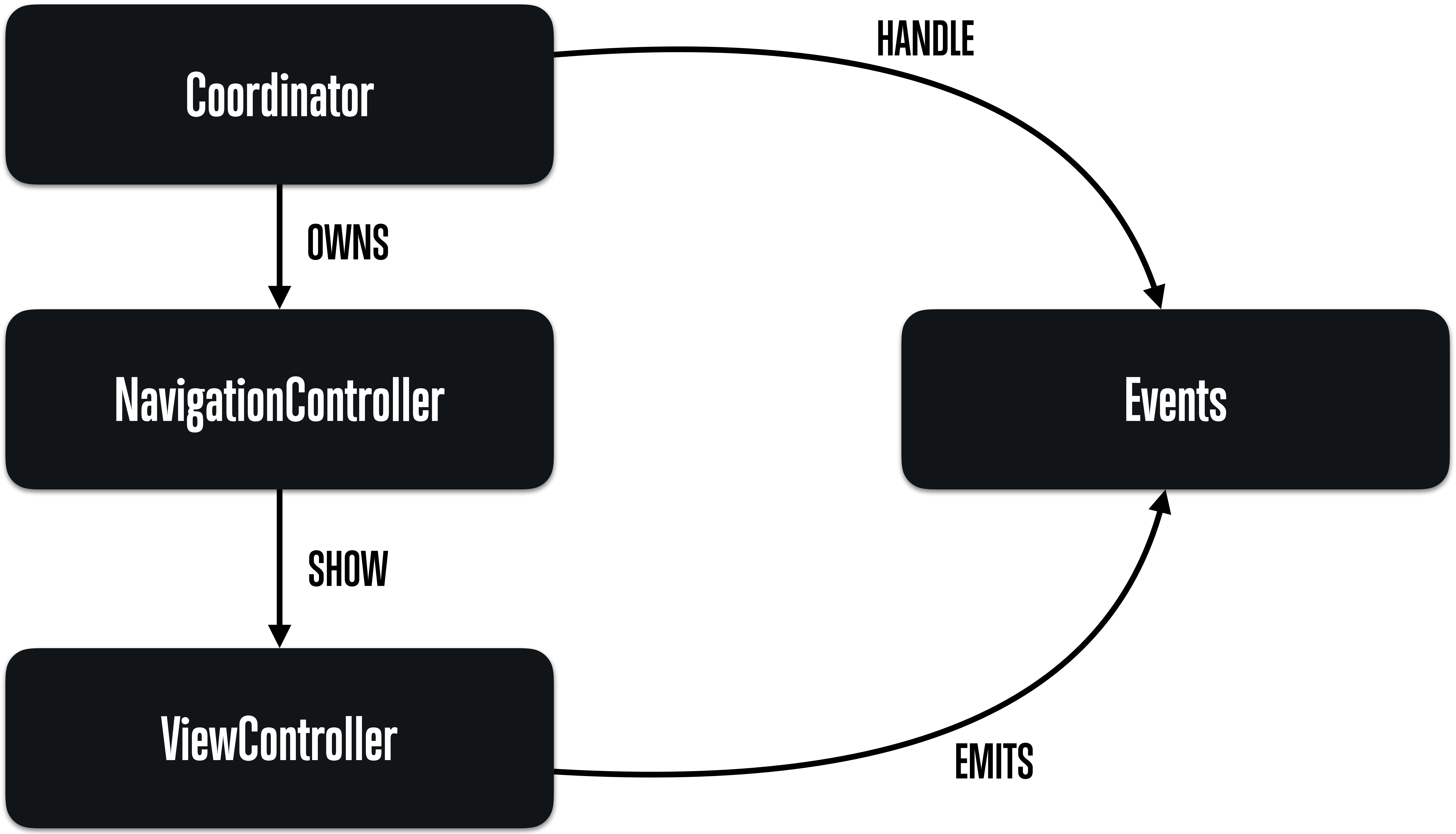
VIEW CONTROLLERS

VIEW CONTROLLERS

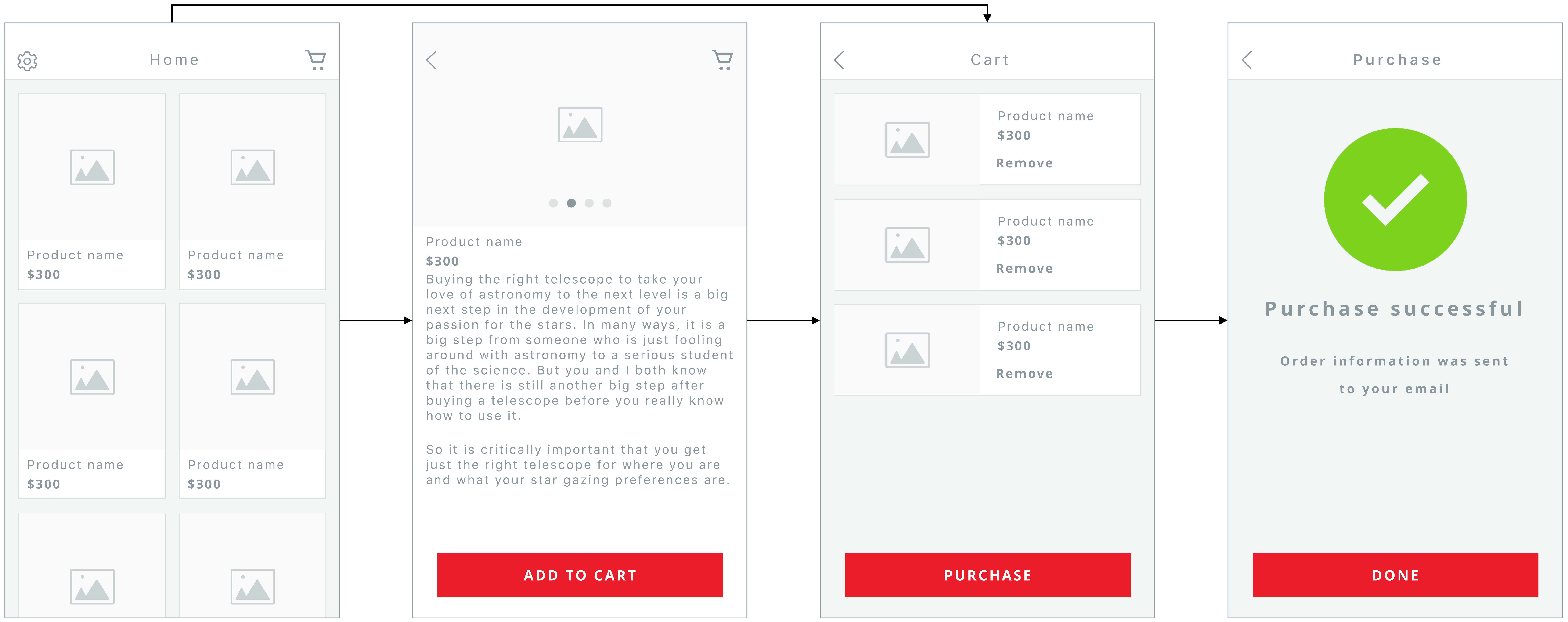
- Always created and shown by the Coordinator
- Emits interaction events

Doesn't know about:

- Other ViewControllers
- The Flow
- It's parent
  - NavigationController
  - TabBarController
  - Coordinator
  - etc


Coordinator
OWNS
NavigationController
SHOW
ViewController
HANDLE
Events
EMITS

SHOPPING FLOW



Buying the right telescope to take your love of astronomy to the next level is a big next step in the development of your passion for the stars. In many ways, it is a big step from someone who is just fooling around with astronomy to a serious student of the science. But you and I both know that there is still another big step after buying a telescope before you really know how to use it.

So it is critically important that you get just the right telescope for where you are and what your star gazing preferences are.

```swift
override func start() {
    showShoppingList()
}

func start(with deepLinkOption: DeepLinkOption) {
    self.start()
    switch deepLinkOption {
    case .itemDetail(let itemID):
        showItemDetail(itemID: itemID)
    case .cart:
        showCart()
    }
}
```

```
override func start() {
    showShoppingList()
}

func start(with deepLinkOption: DeepLinkOption) {
    self.start()
    switch deepLinkOption {
    case .itemDetail(let itemID):
        showItemDetail(itemID: itemID)
    case .cart:
        showCart()
    }
}
```

```swift
func showShoppingList() {
    let shoppingListViewController = ShoppingListViewController()

    shoppingListViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didSelectItem(let itemID):
            self.showItemDetail(itemID: itemID)
        case .didTapCart:
            self.showCart()
        }
    }

    navigationController.pushViewController(shoppingListViewController,
                                            animated: true)
}
```

```swift
func showShoppingList() {

    let shoppingListViewController = ShoppingListViewController()

    shoppingListViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didSelectItem(let itemID):
            self.showItemDetail(itemID: itemID)
        case .didTapCart:
            self.showCart()
        }
    }

    navigationController.pushViewController(shoppingListViewController,
                                            animated: true)
}
```

STRV

ShoppingListViewController

```
func collectionView(_ collectionView: UICollectionView,
                    didSelectItemAt indexPath: IndexPath) {

    let item = dataSource.item(at: indexPath)

    let itemDetailViewController = ItemDetailViewController()
        itemDetailViewController.item = item
    events.emit(.didSelectItem(itemID: item.id))
        self.navigationController?.pushViewController(itemDetailViewController,
                                                      animated: true)

}
```

```swift
func showShoppingList() {

    let shoppingListViewController = ShoppingListViewController()

    shoppingListViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didSelectItem(let itemID):
            self.showItemDetail(itemID: itemID)
        case .didTapCart:
            self.showCart()
        }
    }

    navigationController.pushViewController(shoppingListViewController,
                                            animated: true)

}
```

```
func showShoppingList() {

    let shoppingListViewController = ShoppingListViewController()

    shoppingListViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didSelectItem(let itemID):
            self.showItemDetail(itemID: itemID)
        case .didTapCart:
            self.showCart()
        }
    }

    navigationController.pushViewController(shoppingListViewController,
                                            animated: true)

}
```

STRV

```swift
func showItemDetail(itemID: String) {
    let itemDetailViewController = ItemDetailViewController()
    itemDetailViewController.item = ShoppingItem(id: itemID)

    itemDetailViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didTapCart:
            self.showCart()
        case .didSelectAddToCart(let itemID):
            self.cart.addItem(withID: itemID)
            self.showCart()
        }
    }

    navigationController.pushViewController(itemDetailViewController,
                                            animated: true)
}
```

```swift
func showItemDetail(itemID: String) {
    let itemDetailViewController = ItemDetailViewController()
    itemDetailViewController.item = ShoppingItem(id: itemID)

    itemDetailViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didTapCart:
            self.showCart()
        case .didSelectAddToCart(let itemID):
            self.cart.addItem(withID: itemID)
            self.showCart()
        }
    }

    navigationController.pushViewController(itemDetailViewController,
                                            animated: true)
}
```

```swift
func showCart() {
    let cartViewController = CartViewController()
    cartViewController.cart = cart

    cartViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didTapPurchase(let viewController, let cart):
            guard self.loginService.isUserLogged else {
                return self.showAuthenticationFlow()
            }
            self.shoppingService.purchase(cart: cart) { error in
                guard error == nil else {
                    return viewController.show(error: error!)
                }
                self.showSuccess()
            }
        }
    }

    navigationController.pushViewController(cartViewController,
                                            animated: true)
}
```

```
func showCart() {
    let cartViewController = CartViewController()
    cartViewController.cart = cart

    cartViewController.events.on { [weak self] event in
        guard let self = self else { return }
        switch event {
        case .didTapPurchase(let viewController, let cart):
            guard self.loginService.isUserLogged else {
                return self.showAuthenticationFlow()
            }
            self.shoppingService.purchase(cart: cart) { error in
                guard error == nil else {
                    return viewController.show(error: error!)
                }
                self.showSuccess()
            }
        }
    }

    navigationController.pushViewController(cartViewController,
                                             animated: true)
}
```

```swift
func showAuthenticationFlow() {
    let authFlow = AuthenticationFlow()
    children.insert(authFlow)
    authFlow.start()

    authFlow.didFinish.on { (authFlow) in
        children.remove(authFlow)
        authFlow.root.dismiss(animated: true, completion: nil)
    }

    navigationController.present(authFlow.root,
                                 animated: true,
                                 completion: nil)
}
```

```swift
func showAuthenticationFlow() {
    let authFlow = AuthenticationFlow()
    children.insert(authFlow)
    authFlow.start()

    authFlow.didFinish.on { (authFlow) in
        children.remove(authFlow)
        authFlow.root.dismiss(animated: true, completion: nil)
    }

    navigationController.present(authFlow.root,
                                 animated: true,
                                 completion: nil)
}
```

```swift
func showAuthenticationFlow() {
    let authFlow = AuthenticationFlow()
    children.insert(authFlow)
    authFlow.start()

    authFlow.didFinish.on { (authFlow) in
        children.remove(authFlow)
        authFlow.root.dismiss(animated: true, completion: nil)
    }

    navigationController.present(authFlow.root,
                                 animated: true,
                                 completion: nil)
}
```

STRV

THAT'S IT

Alexandre Tavares

alexandre.tavares@strv.com

compiled.social/AlTavares

STRV

STRV