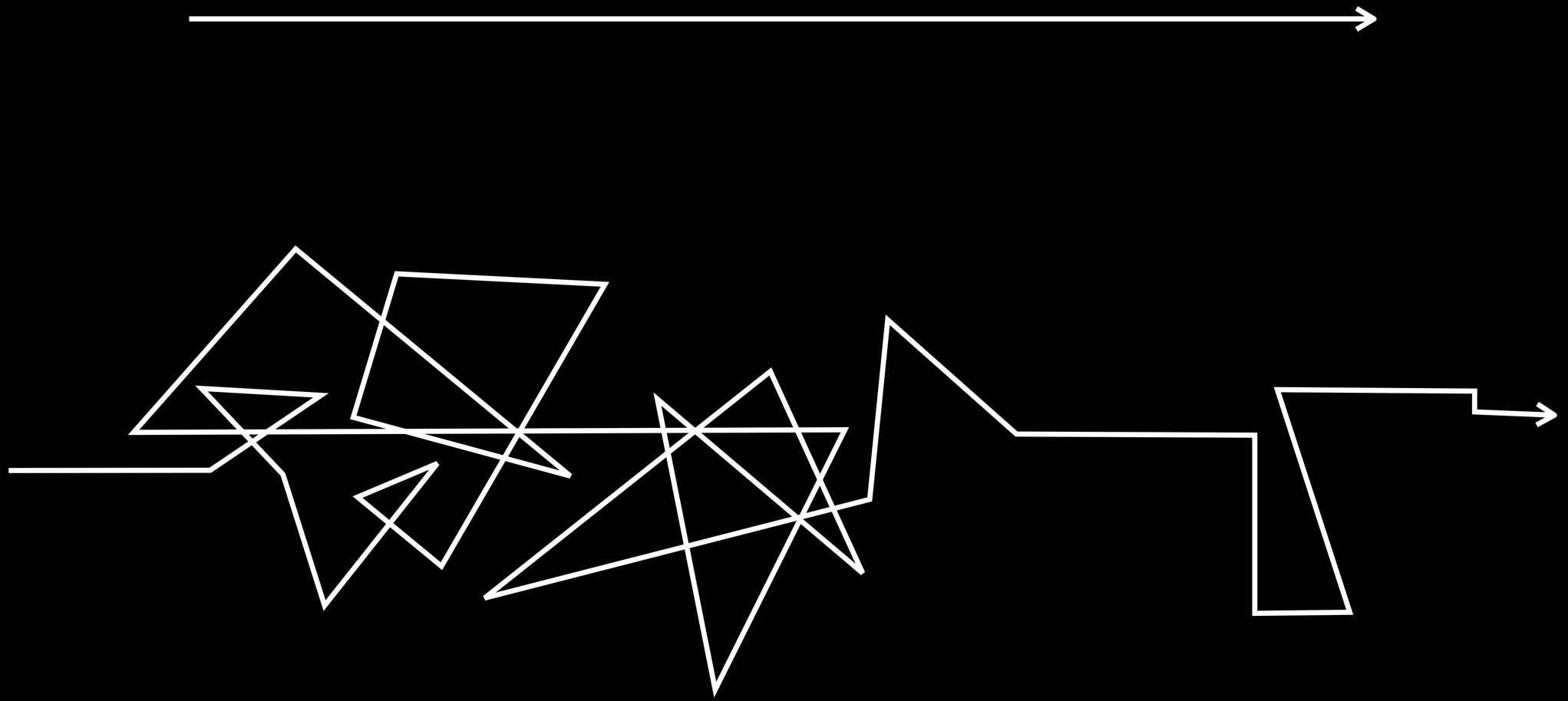
Navigation et architecture

Organiser son app

Navigation should feel natural and familiar, and shouldn’t dominate the interface or draw focus away from content. In iOS, there are three main styles of navigation.

Navigation - iOS Human Interface Guidelines - Apple

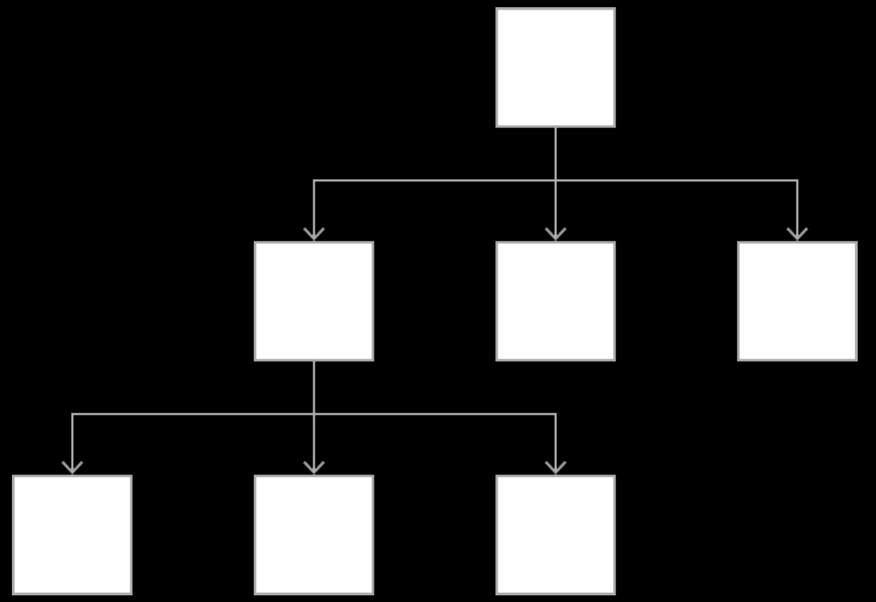


3 types de navigation

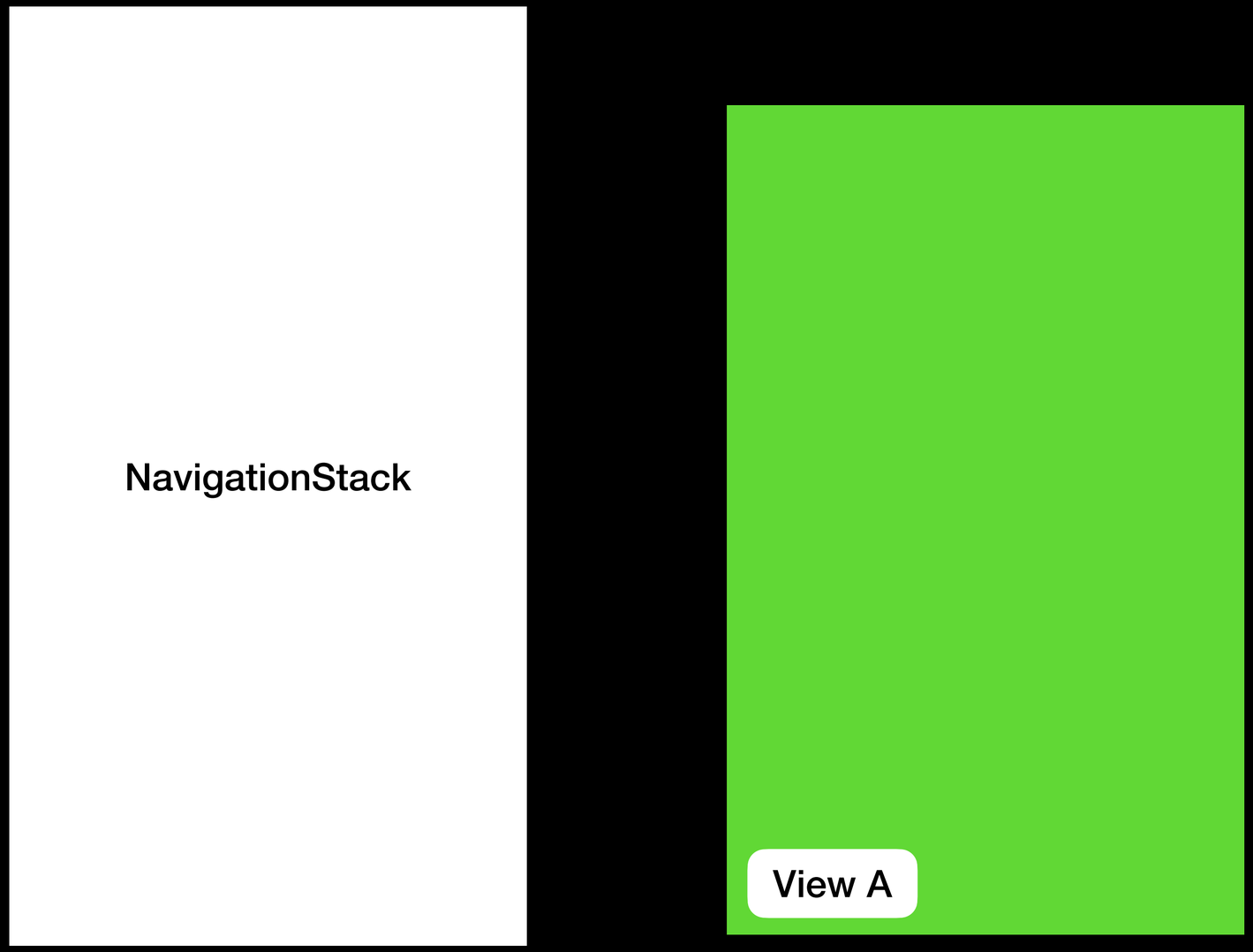
- Navigation hiérarchique
- Navigation à plat
- Navigation guidée par le contenu

Navigation hiérarchique

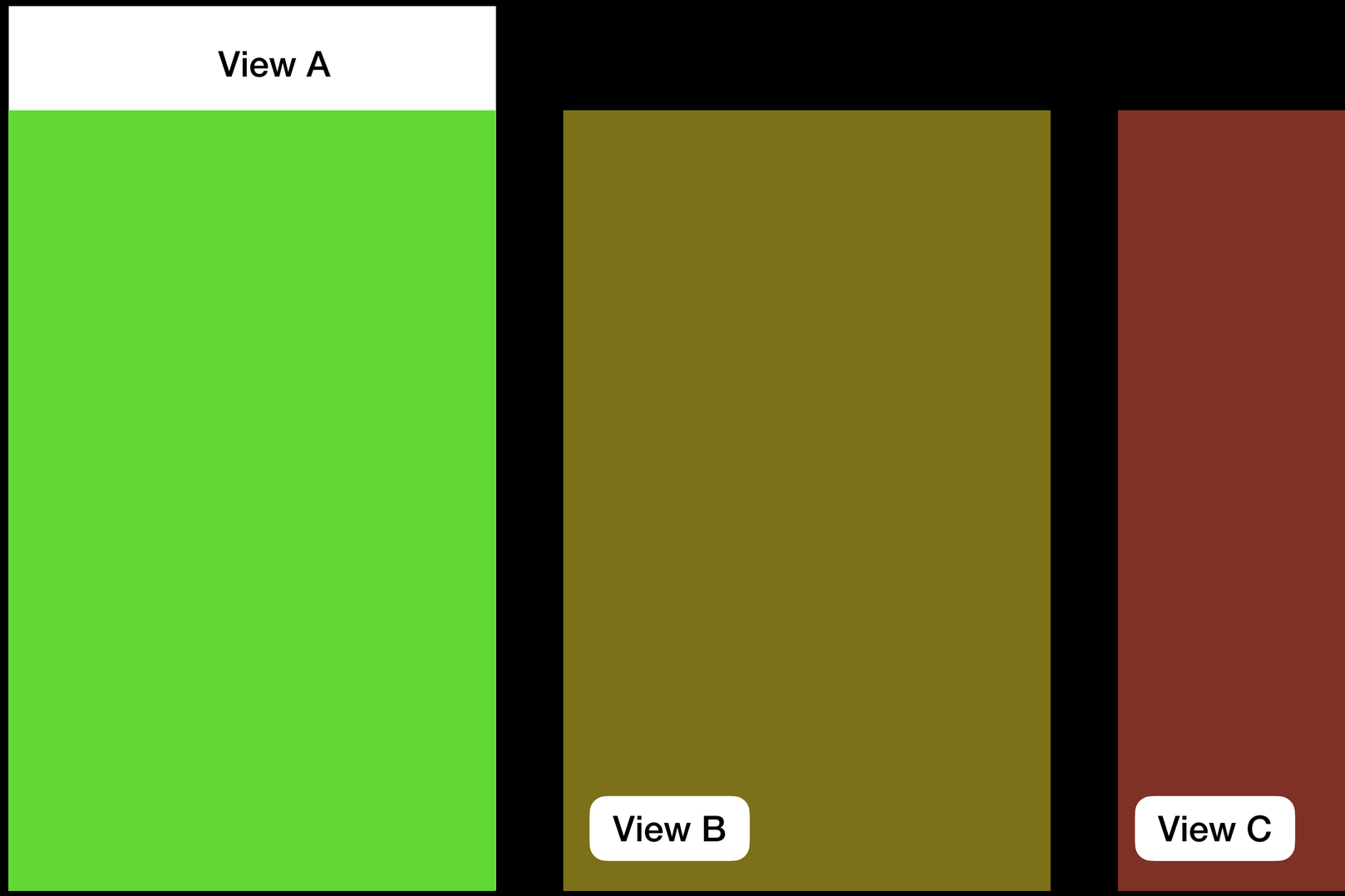
- Un choix par écran, jusqu'à destination
- Pour aller à une autre destination il faut rebrousser chemin, et faire d'autres choix
- Idéal pour les données hiérarchiques



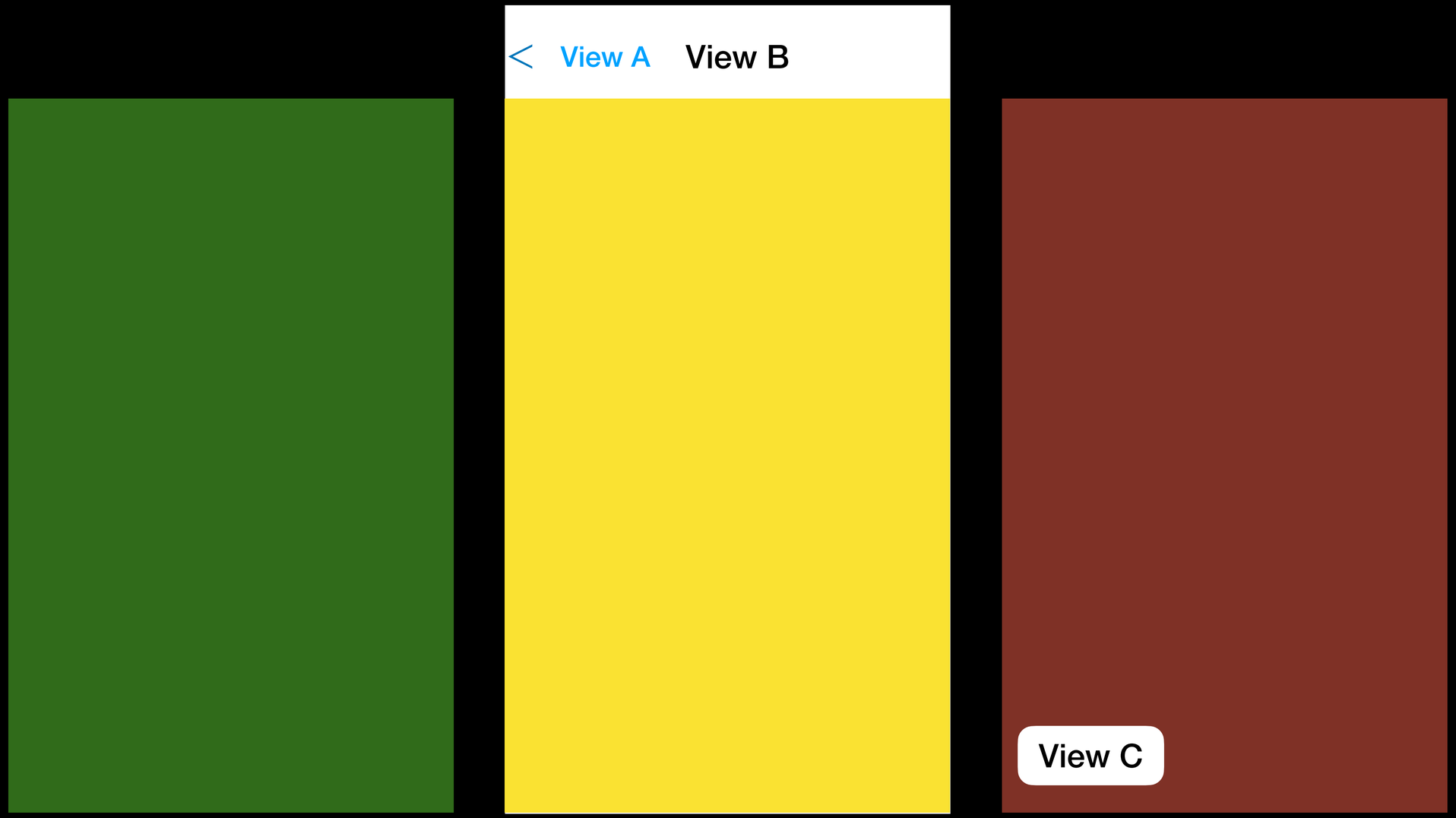
Navigation hiérarchique



Navigation hiérarchique

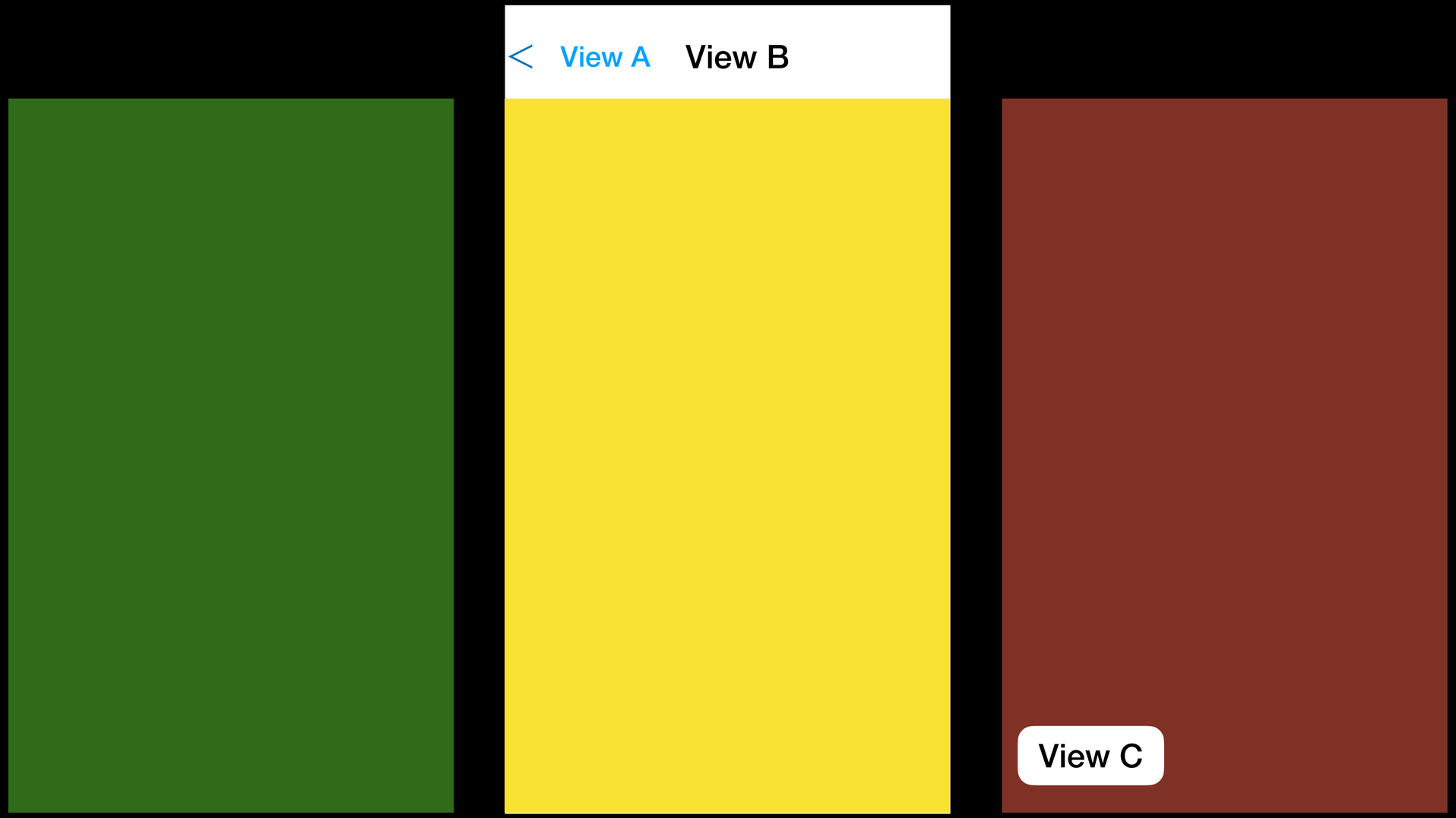


Navigation hiérarchique

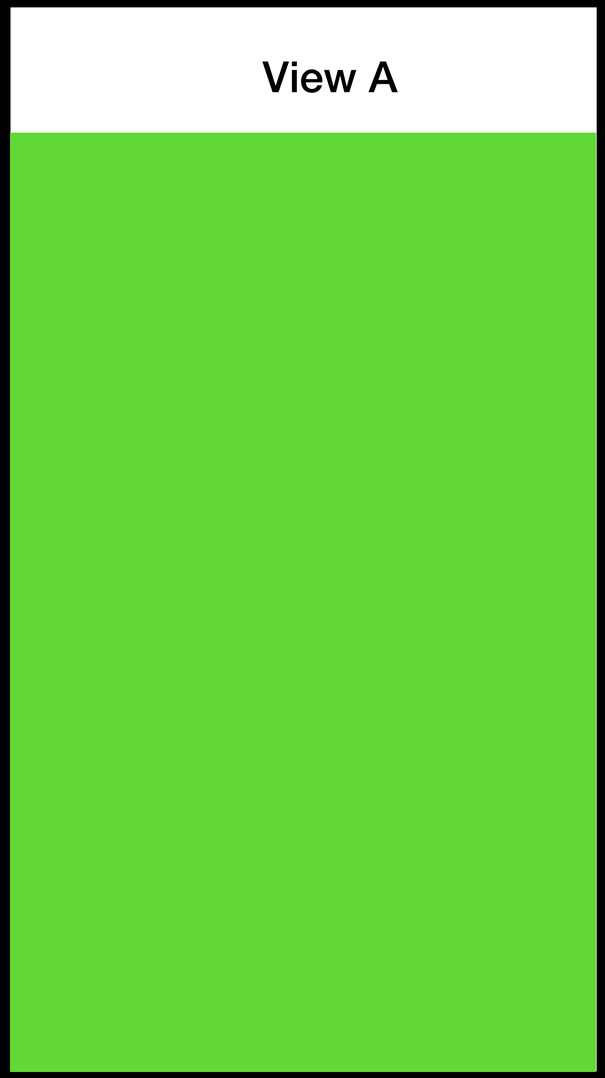


Navigation hiérarchique

Navigation hiérarchique



Navigation hiérarchique

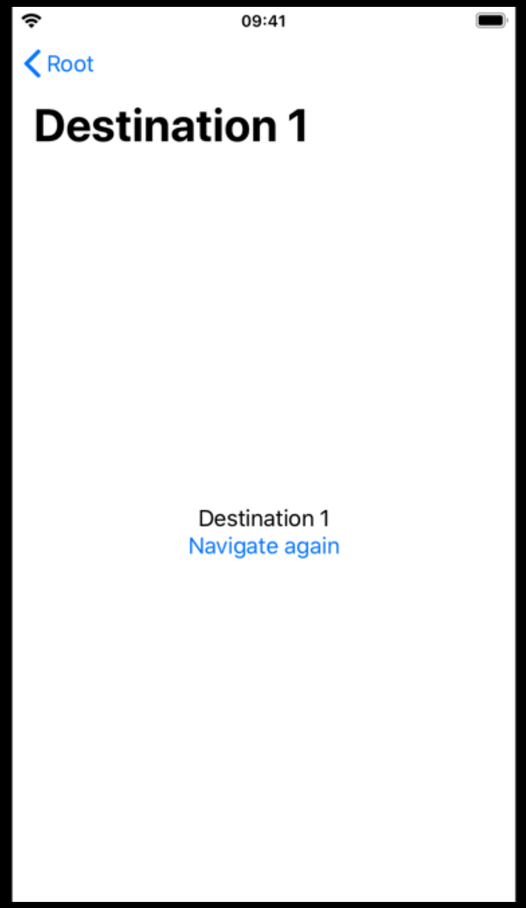


Navigation hiérarchique

- Une NavigationStack est nécéssaire pour ce type de navigation
  - Elle doit contenir la vue "point de départ" de la navigation
  - Les vues peuvent avoir un navigationBarTitle qui sera utilisé comme titre si affichés dans une NavigationStack
- Un NavigationLink permet de faire afficher un autre écran dans la NavigationView
  - La NavigationStack ajoute un bouton "Back" pour revenir à l'écran précédent

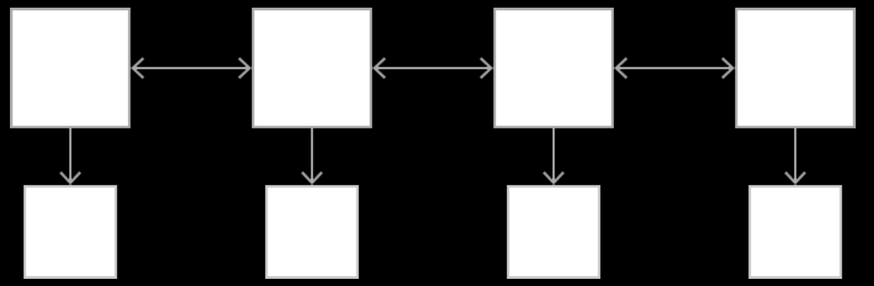
```
struct NavigationDemo: View {
    var body: some View {
        NavigationStack {
            NavigationLink(destination: DestinationView(textToDisplay: "Destination 1") ) {
                Text("Navigate")
            }.navigationTitle("Root")
        }
    }
}
```

```
struct NavigationDemo: View {
    var body: some View {
        NavigationStack {
            NavigationLink(destination: DestinationView(textToDisplay: "Destination 1") ) {
                Text("Navigate")
            }.navigationTitle("Root")
        }
    }
}
```



Navigation à plat

- Pour sélectionner des destinations différentes
- Idéal pour des catégories de données différentes

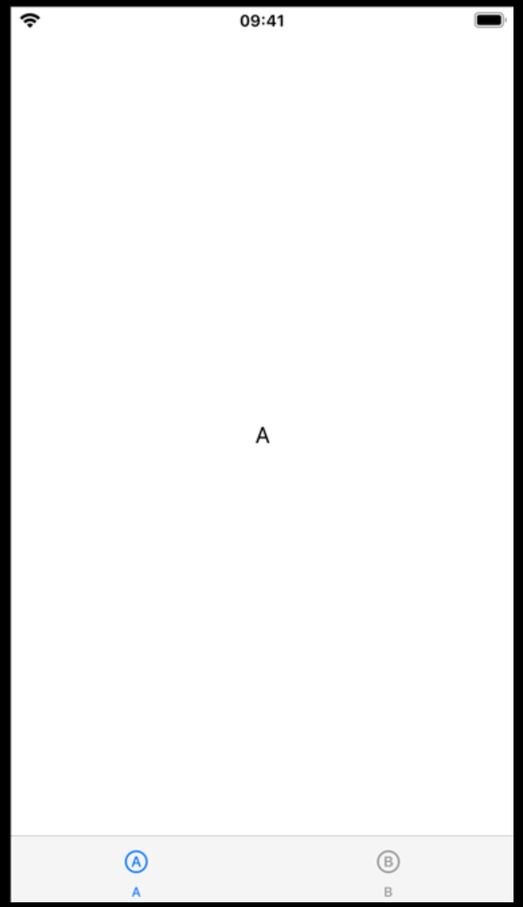


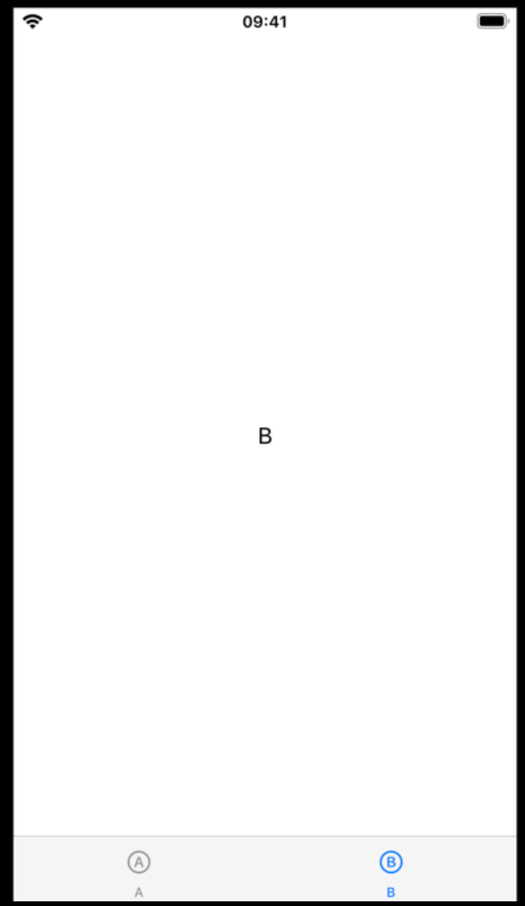
Navigation à plat

- Une TabView est nécéssaire pour ce type de navigation
  - Elle doit contenir les différentes vues à afficher
  - Les vues peuvent avoir un tabItem contenant text et/ou image qui sera utilisé comme titre dans l'onglet

```swift
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                 }
            Text("B")
                .tabItem {
                    Text("B")
                    Image(systemName: "b.circle")
                }
        }
    }
}
```

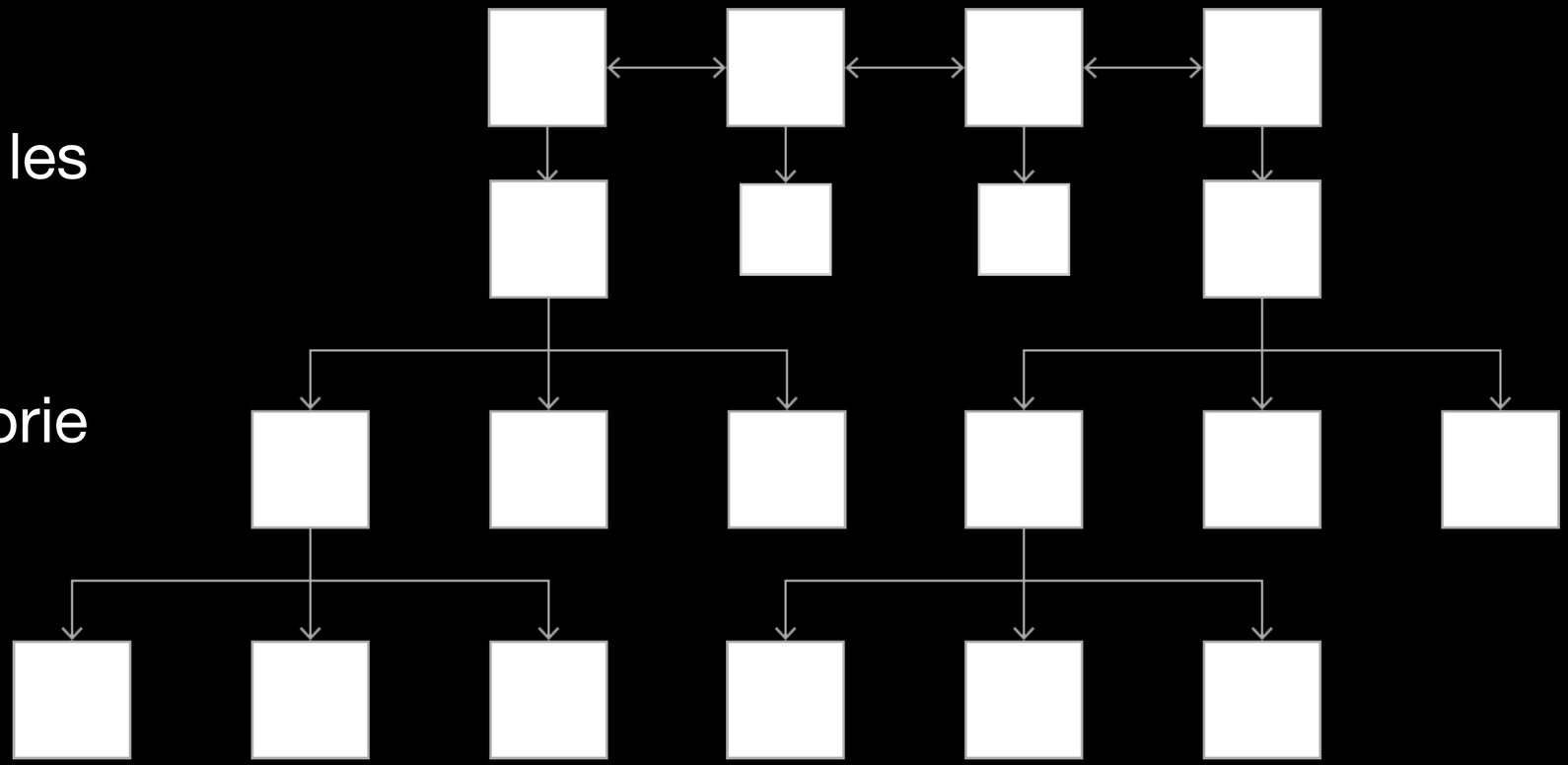
```swift
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")
                    Image(systemName: "b.circle")
                }
        }
    }
}
```





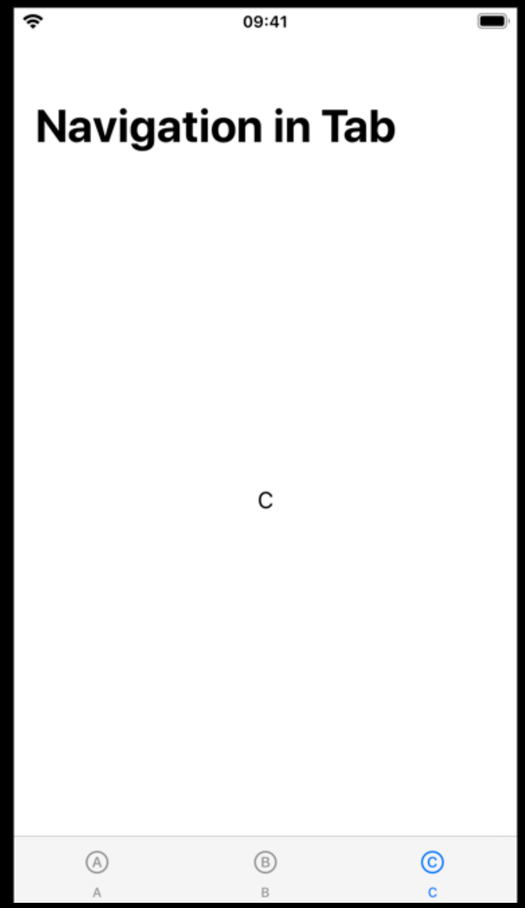
Navigation à plat + hiérarchie

- Il est possible de combiner les types de navigation
- A plat, puis navigation à l'intérieur de chaque catégorie



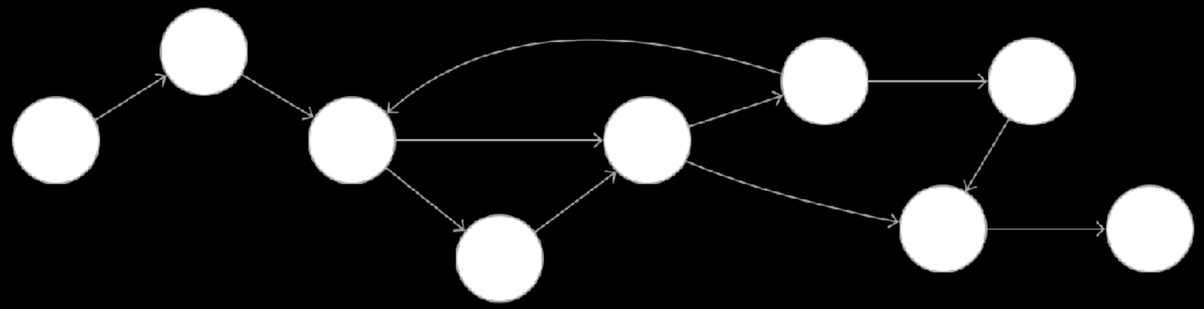
```
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
            }
            Text("B")
                .tabItem {
                    Text("B")
                    Image(systemName: "b.circle")
            }
            NavigationView {
                Text("C")
                    .navigationBarTitle("Navigation in Tab")
            }
            .tabItem {
                Text("C")
                Image(systemName: "c.circle")
            }
        }
    }
}
```

```
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")
                    Image(systemName: "b.circle")
                }
            NavigationView {
                Text("C")
                    .navigationBarTitle("Navigation in Tab")
            }
            .tabItem {
                Text("C")
                Image(systemName: "c.circle")
            }
        }
    }
}
```



Navigation guidée par le contenu

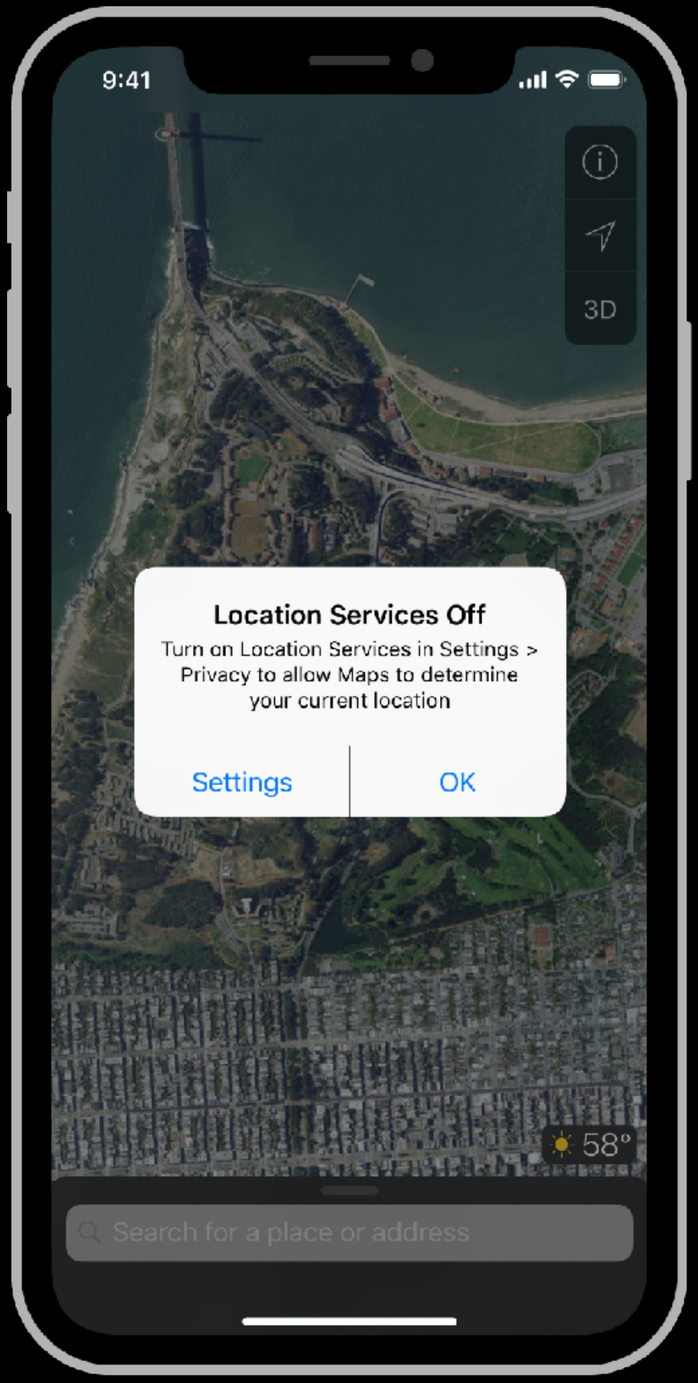
- Navigation libre dans le contenu, ou définie par le contenu
- Jeux, livres, ou autres experience immersive

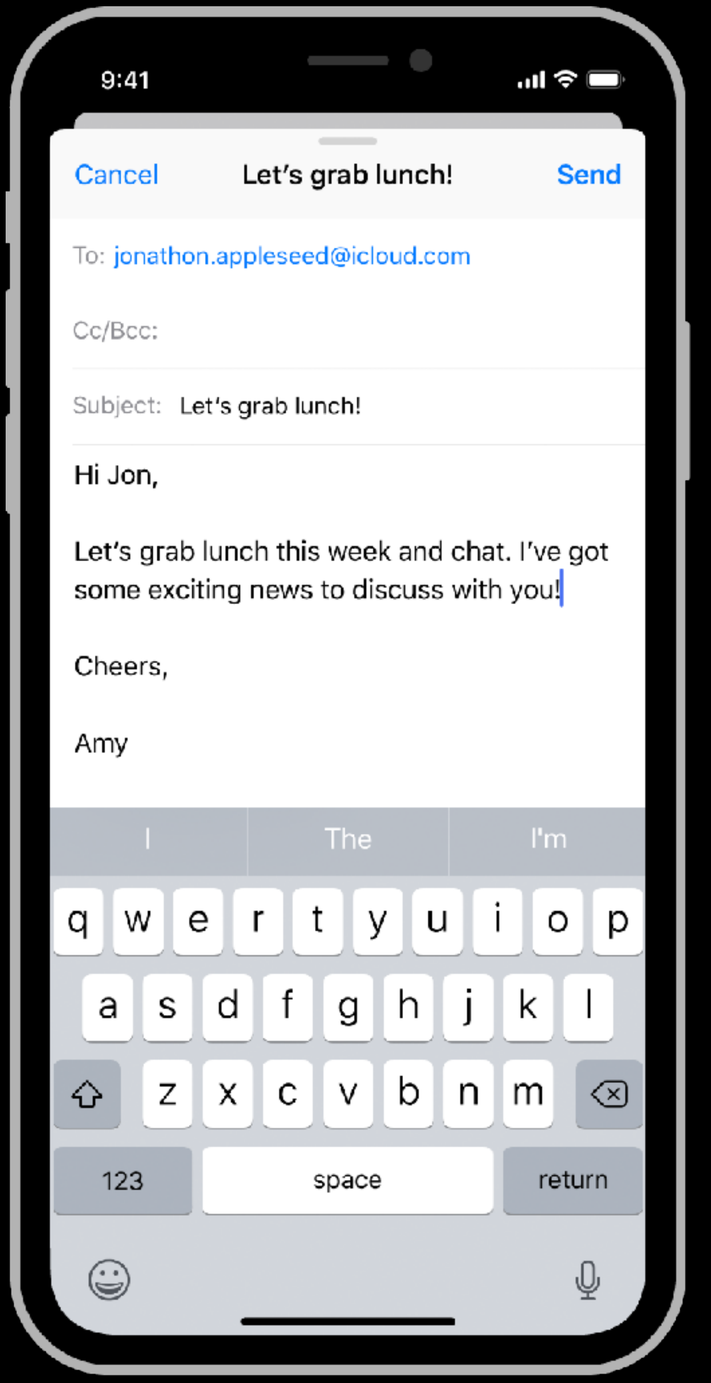


Modalité

Modality is a design technique that presents content in a temporary mode that’s separate from the user's previous current context and requires an explicit action to exit.

Modality - iOS Human Interface Guidelines - Apple


9:41
3D
Location Services Off
Turn on Location Services in Settings > Privacy to allow Maps to determine your current location
Settings
OK
58°
Search for a place or address

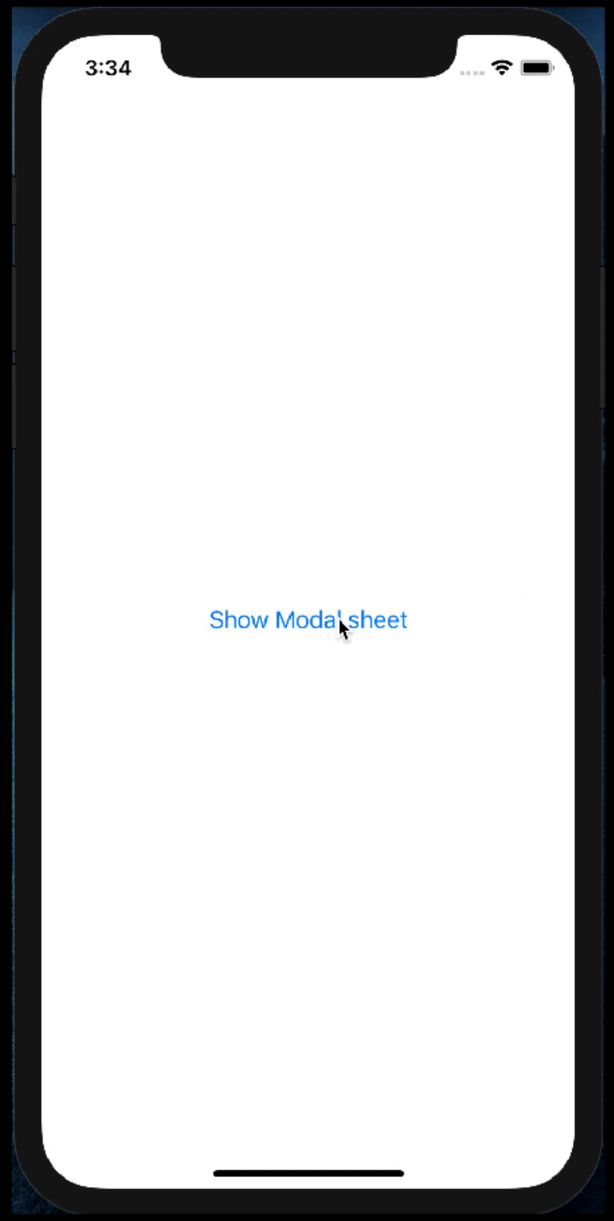

9:41
Cancel
Let's grab lunch!
Send
To: jonathon.appleseed@icloud.com
Cc/Bcc:
Subject: Let's grab lunch!
Hi Jon,
Let's grab lunch this week and chat. I've got some exciting news to discuss with you!
Cheers,
Amy
I
The
I'm
q w e r t y u i o p
a s d f g h j k l
z x c v b n m
123
space
return

Modalité

- La modalité permet
  - D'aider l'utilisateur à rester concentrer sur une action
  - De s'assurer que l'utilisateur reçoive et agisse sur une information

Modalité

```
struct Modality: View {

    @State private var isDisplayingSheet = false

    var body: some View {
        Button("Show Modal sheet") {
            self.isDisplayingSheet.toggle()
        }.sheet(isPresented: $isDisplayingSheet) {
            Text("This is a modal sheet")
        }
    }
}
```

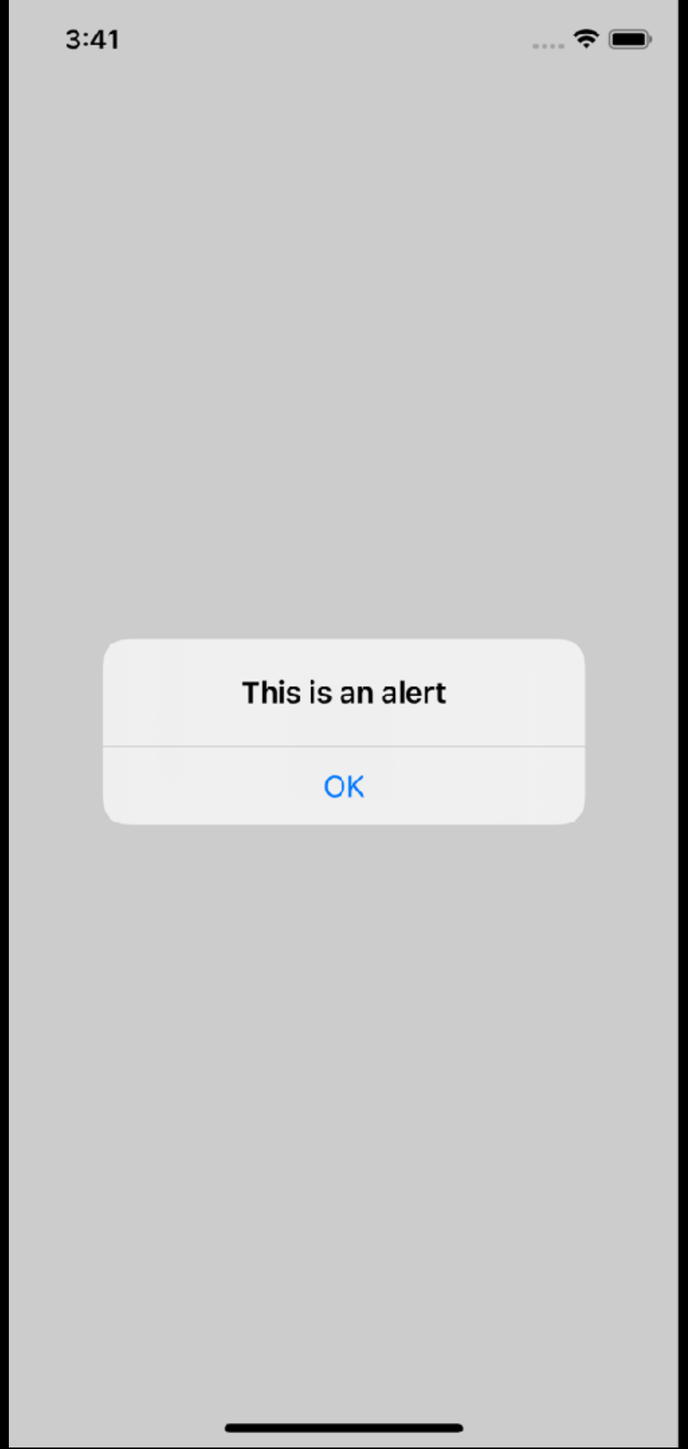


Modalité

```
struct Modality: View {

    @State private var isDisplayingSheet = false

    var body: some View {
        Button("Show Modal sheet") {
            self.isDisplayingSheet.toggle()
        }.sheet(isPresented: $isDisplayingSheet) {
            Text("This is a modal sheet")
        }
    }
}
```

Alerte

```
struct Modality: View {

   @State private var isDisplayingAlert = false

    var body: some View {
       Button("Show alert") {
            self.isDisplayingAlert.toggle()
       }.alert("Important alert", isPresented: $isDisplayingAlert) {
            Button("Delete", role: .destructive) {}
            Button("Cancel", role: .cancel) {}
       }
    }
}
```



Alerte

```
struct Modality: View {

   @State private var isDisplayingAlert = false

    var body: some View {
       Button("Show alert") {
            self.isDisplayingAlert.toggle()
       }.alert("Important alert", isPresented: $isDisplay
            Button("Delete", role: .destructive) {}
            Button("Cancel", role: .cancel) {}
       }
    }
}
```

Action (ConfirmationDialog)

```
struct Modality: View {

    @State private var isDisplayingActionSheet = false

    var body: some View {

        Button("Show Action sheet") {
            self.isDisplayingActionSheet.toggle()
        }.actionSheet(isPresented: $isDisplayingActionSheet) {
            ActionSheet(title: Text("Action title"), message: Text("Action message"),
buttons: [.destructive(Text("Destructive action"), action: {

            }), .default(Text("Default action"), action: {

            }), .cancel()])
        }
    }
}
```

Action (ConfirmationDialog)

```
iew {

 var isDisplayingActionSheet = false

 View {

ow Action sheet") {
sDisplayingActionSheet.toggle()
eet(isPresented: $isDisplayingActionSheet) {
Sheet(title: Text("Action title"), message: Text("Action message"),
tive(Text("Destructive action"), action: {

efault(Text("Default action"), action: {

ancel()])
```