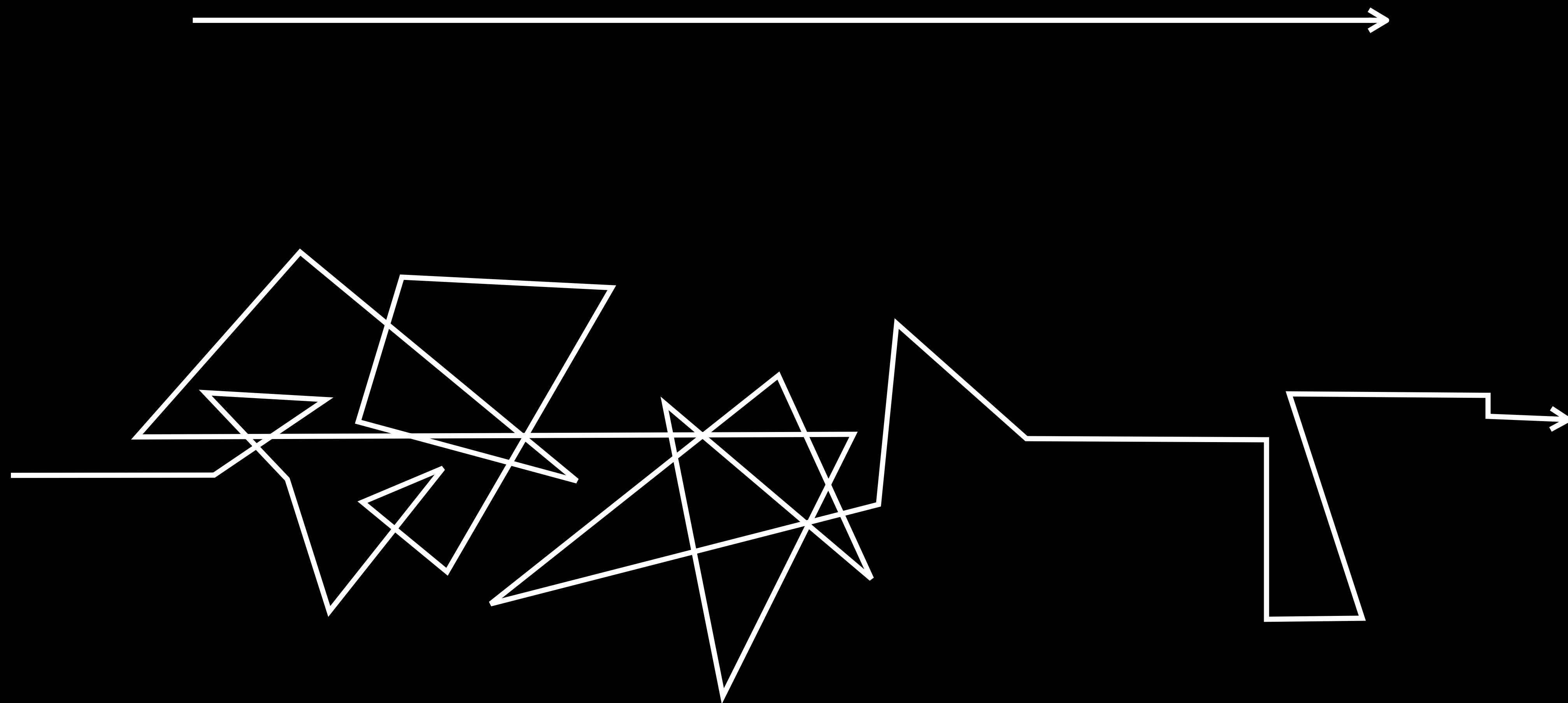


# Navigation et architecture

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**

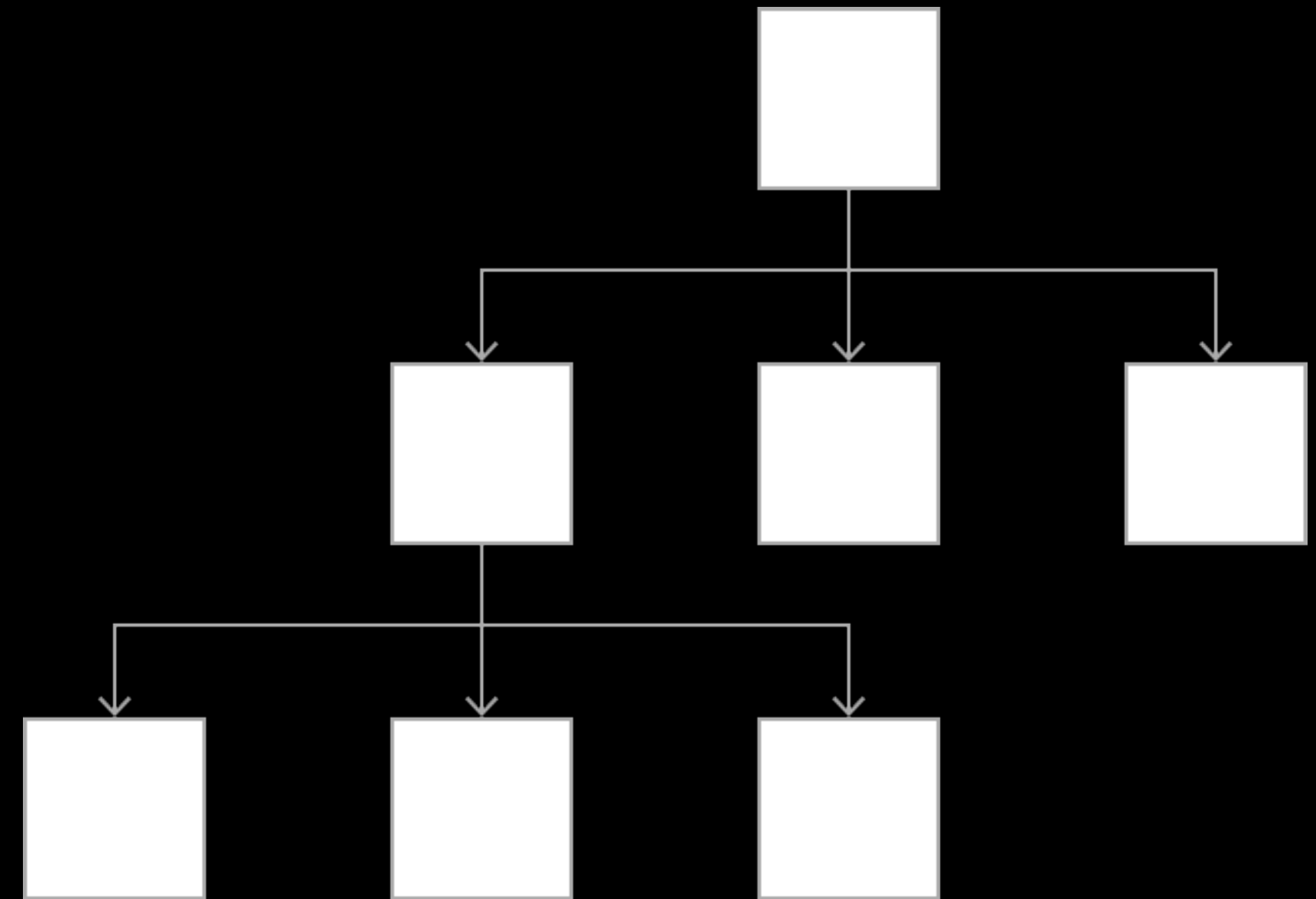


# 3 types de navigation

- 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

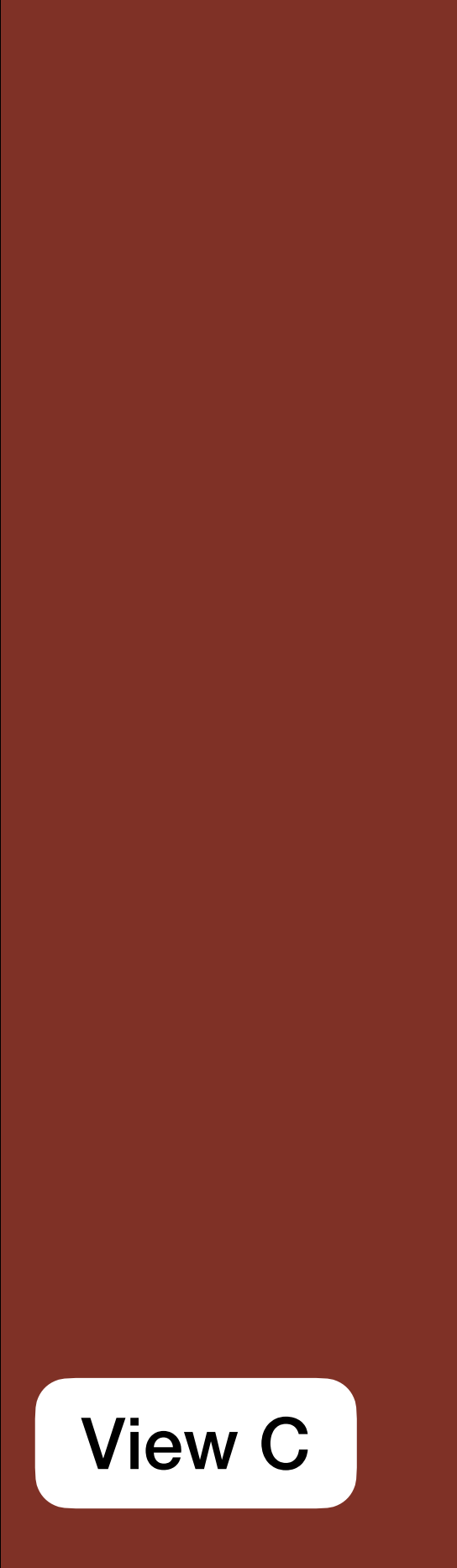
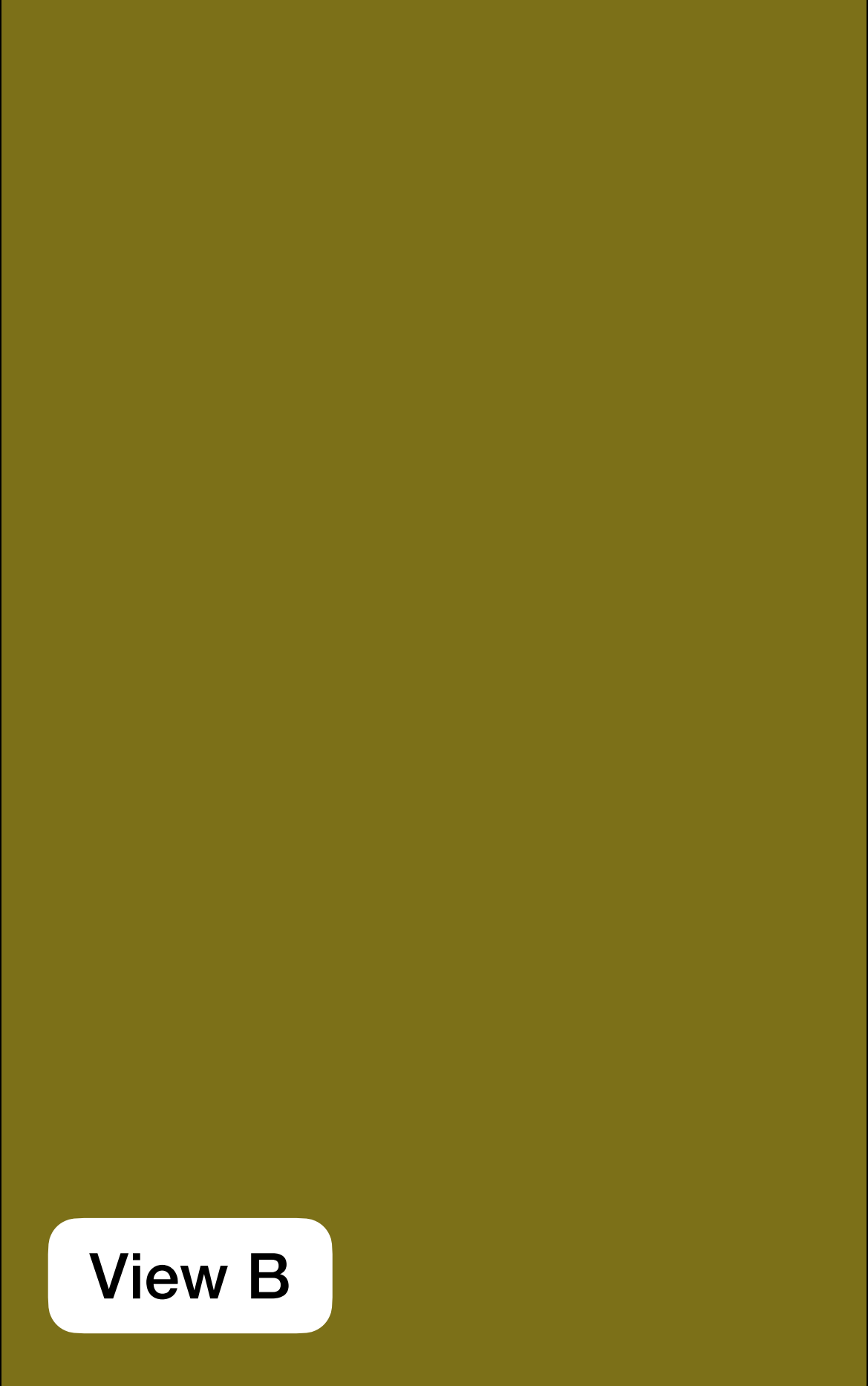
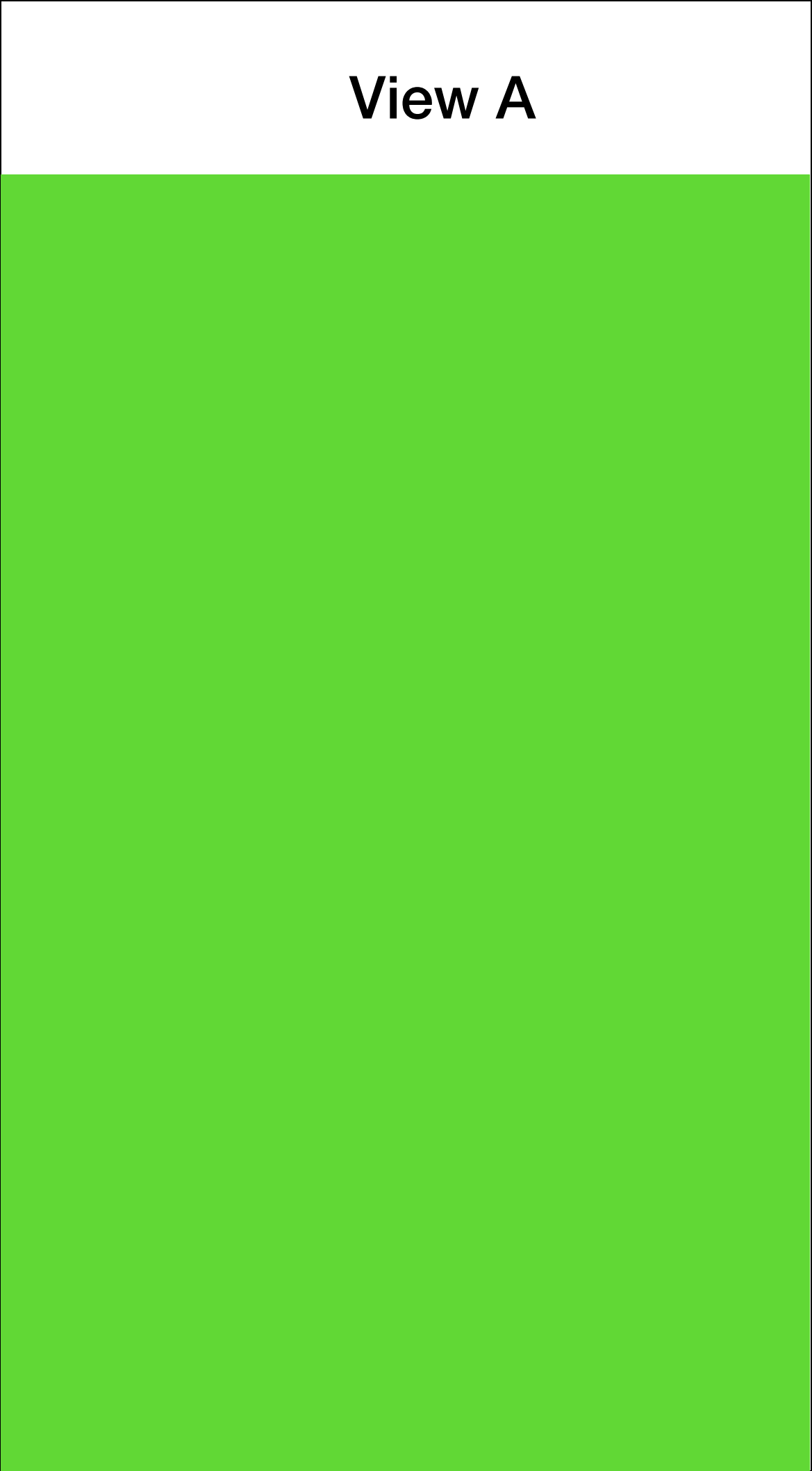


# Navigation hiérarchique

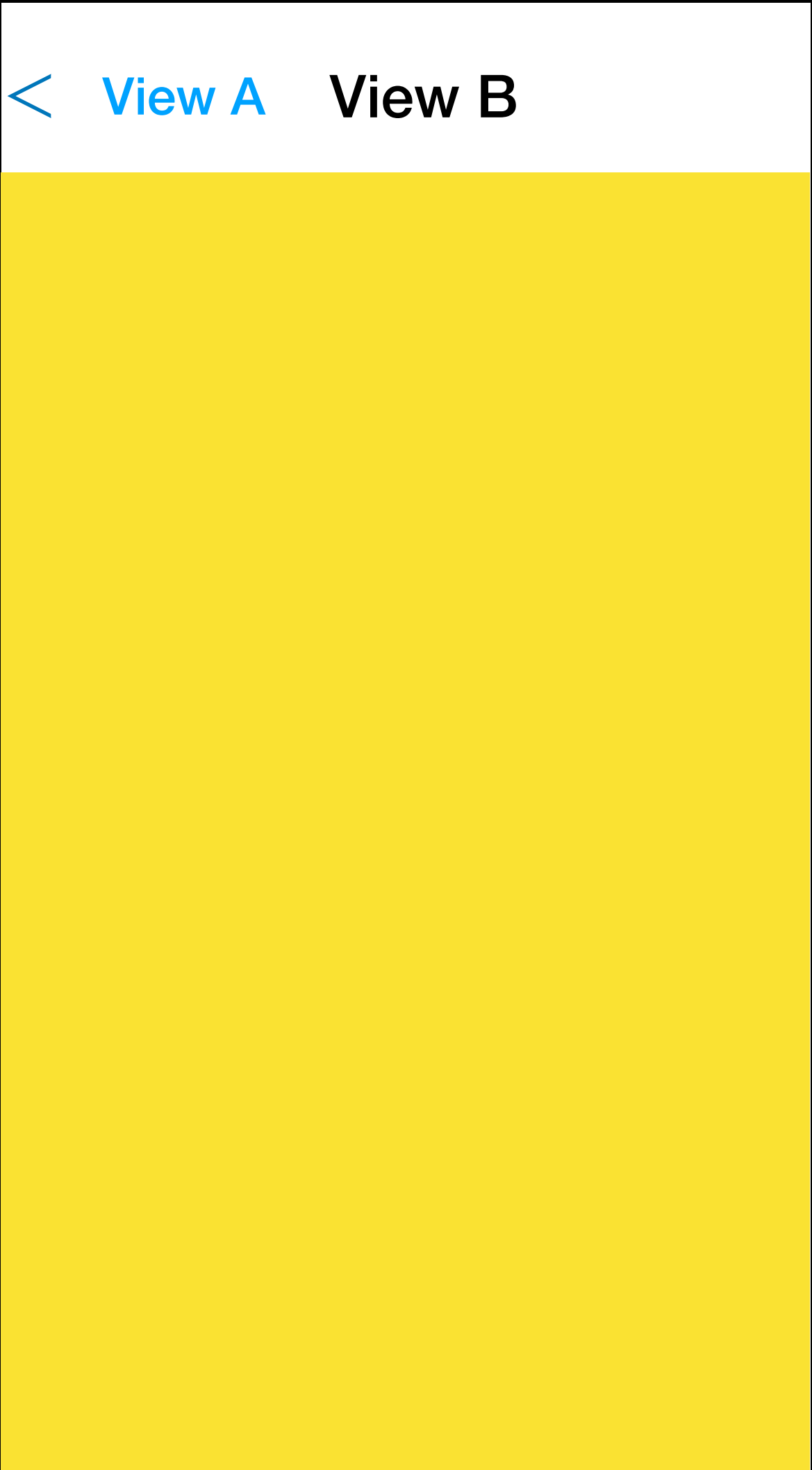
NavigationView

View A

# Navigation hiérarchique

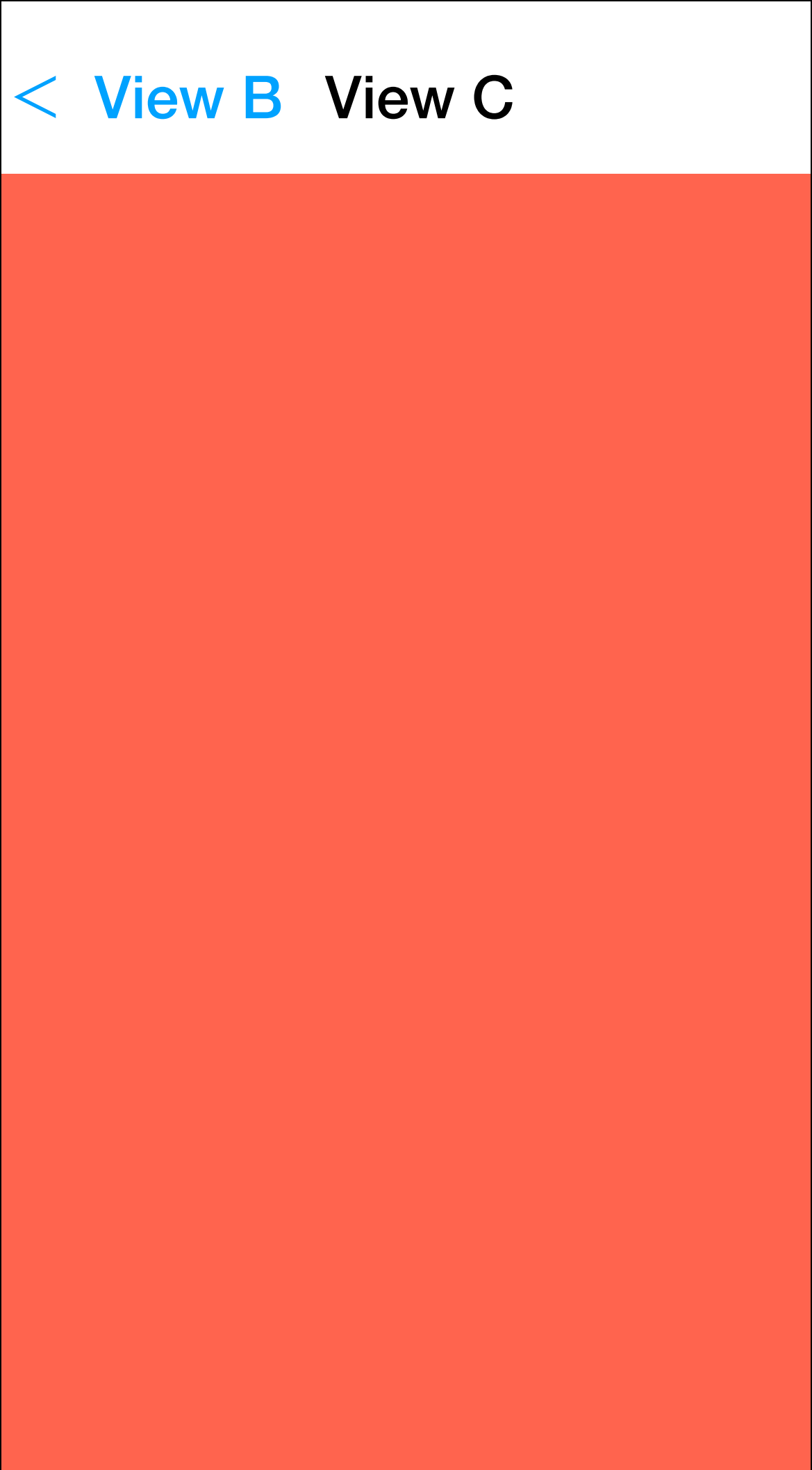


# Navigation hiérarchique

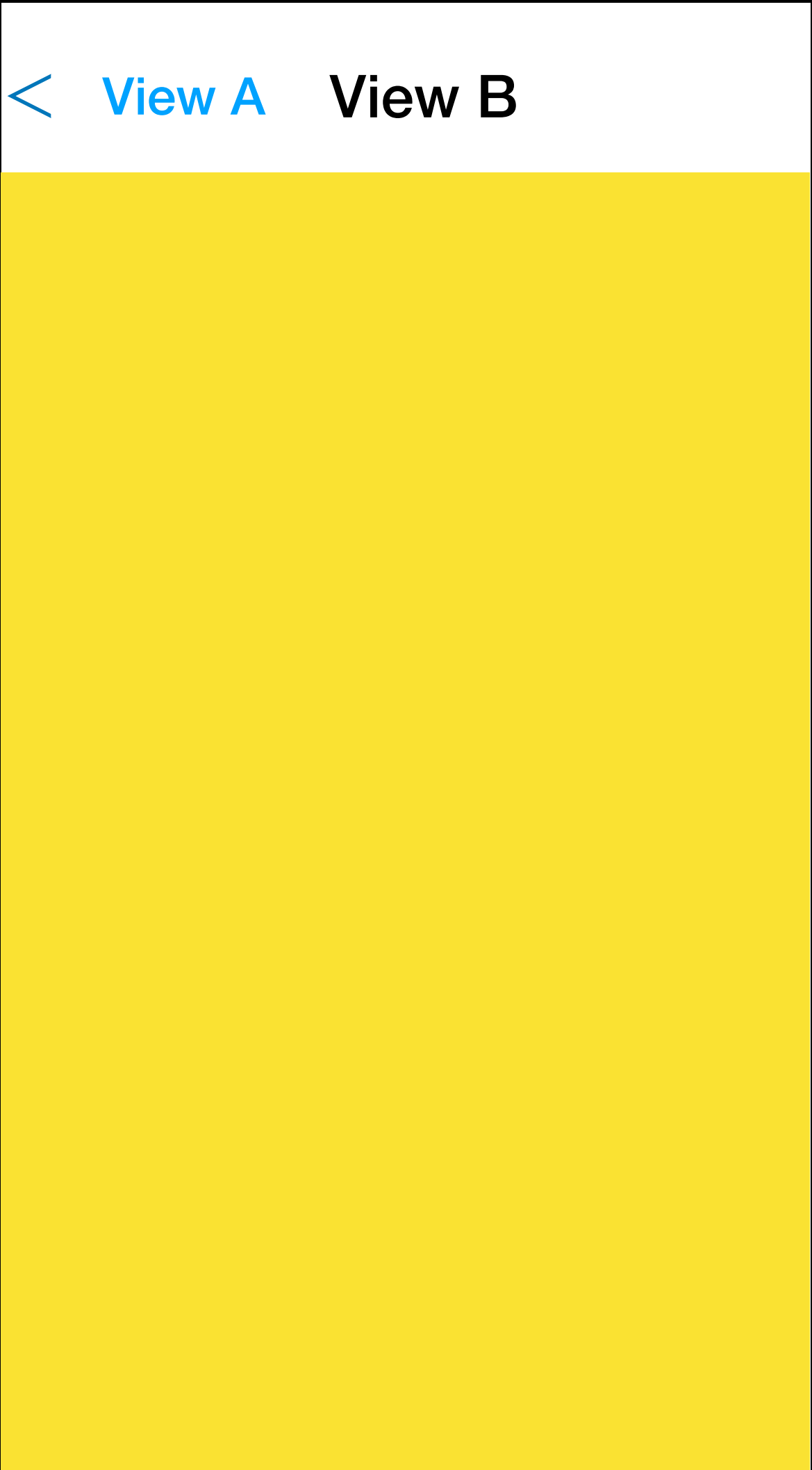




# Navigation hiérarchique



# Navigation hiérarchique



# Navigation hiérarchique

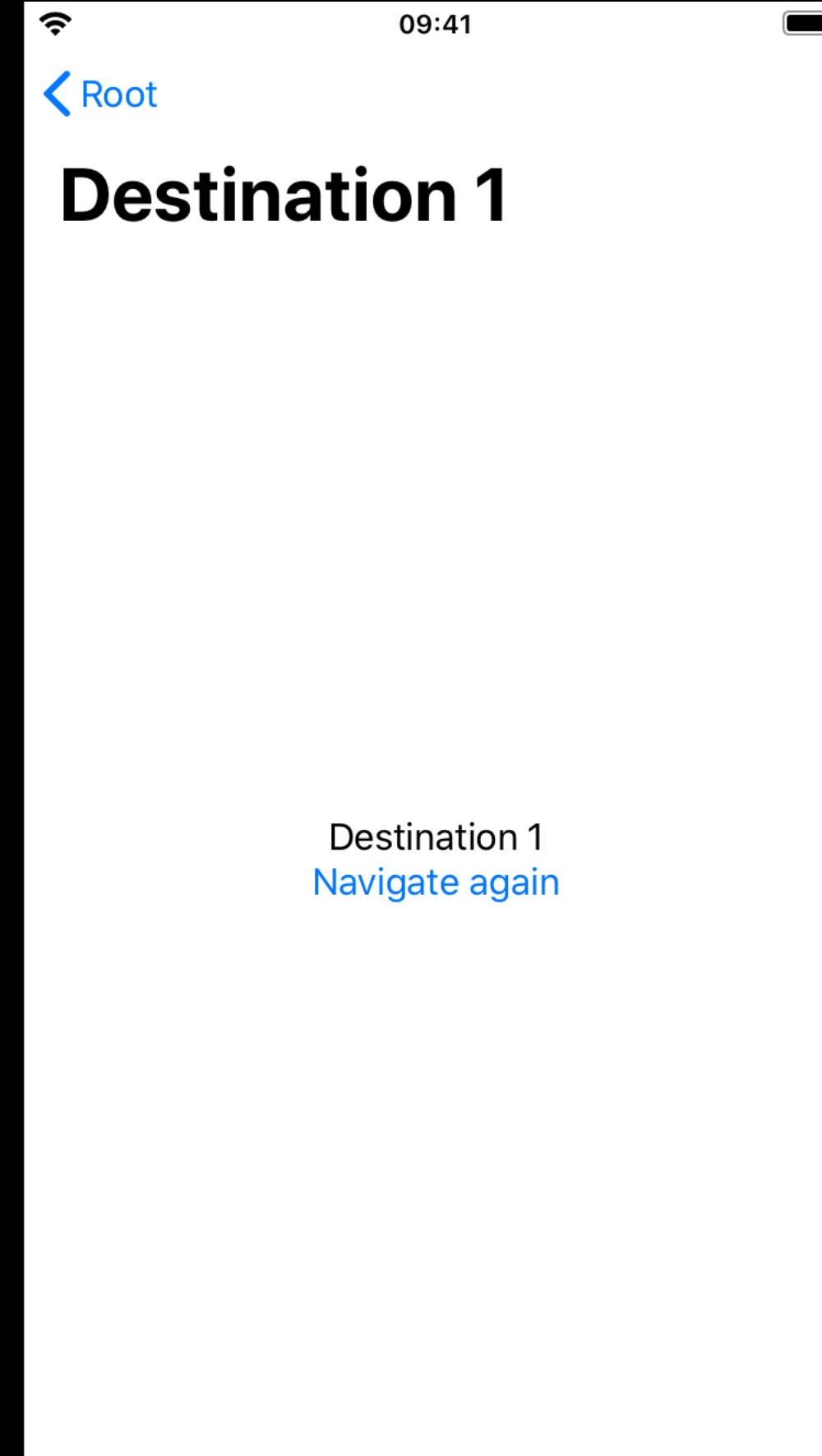
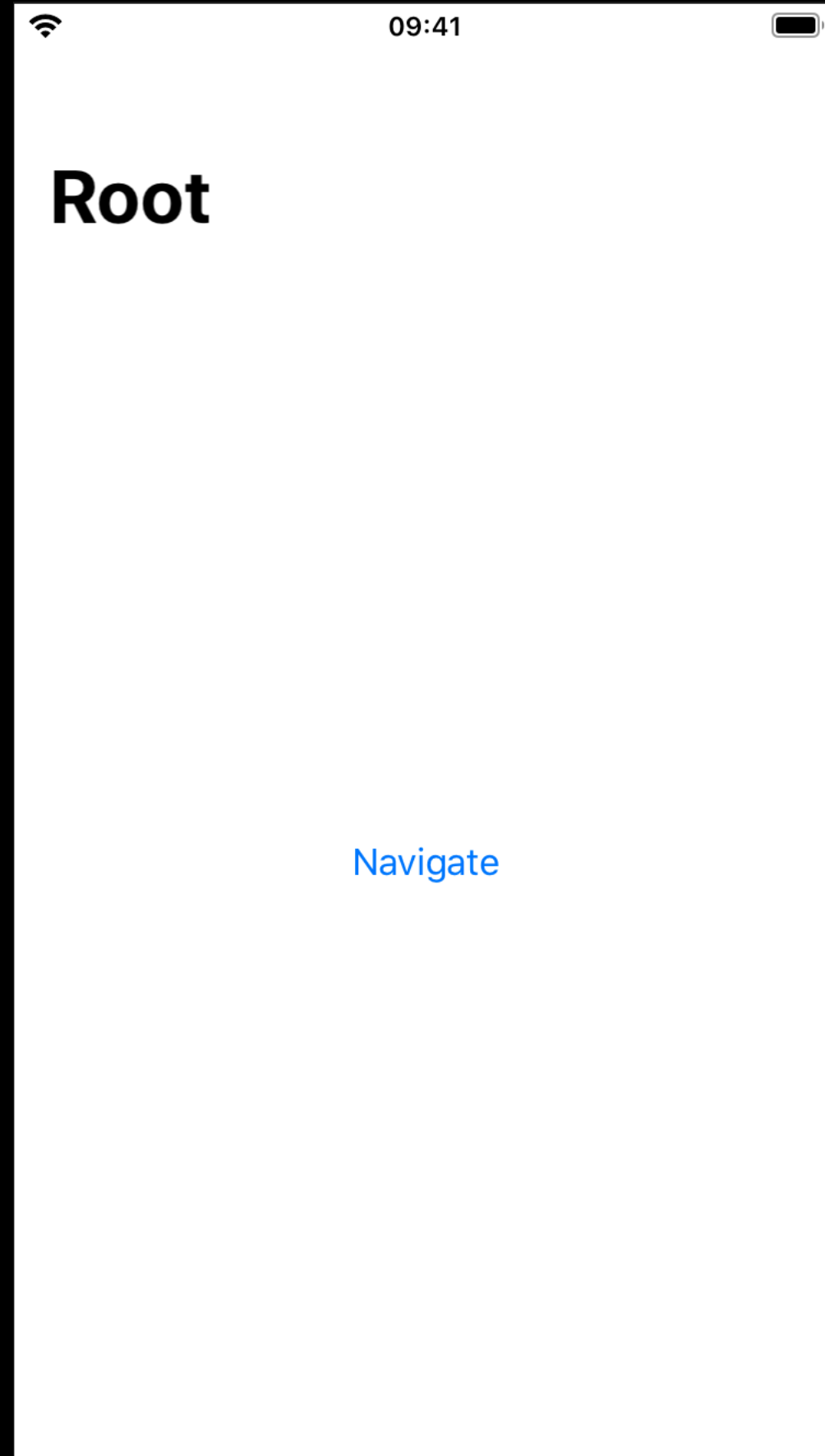


# Navigation hiérarchique

- Une `NavigationView` est nécessaire 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 `NavigationView`
- Un `NavigationLink` permet de faire afficher un autre écran dans la `NavigationView`
  - La `NavigationView` ajoute un bouton "Back" pour revenir à l'écran précédent

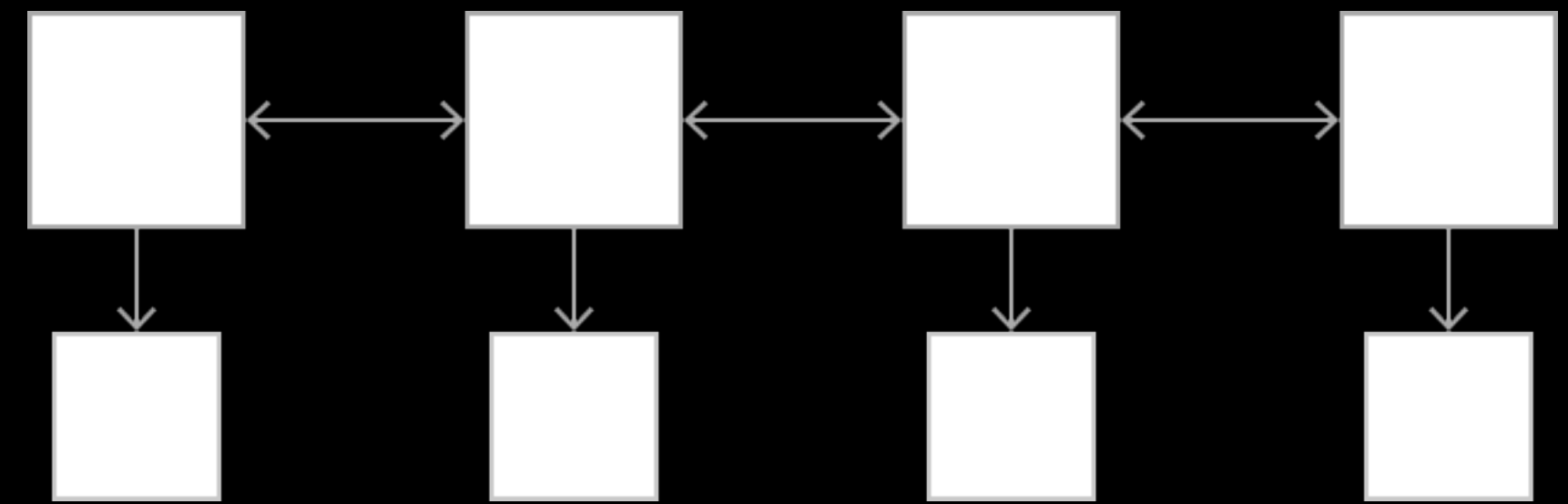
```
struct NavigationDemo: View {
    var body: some View {
        NavigationView {
            NavigationLink(destination: DestinationView(textToDisplay: "Destination 1") ) {
                Text("Navigate")
            }.navigationBarTitle("Root")
        }
    }
}
```

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



# Navigation à plat

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



# Navigation à plat

- Une TabView est nécessaire 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

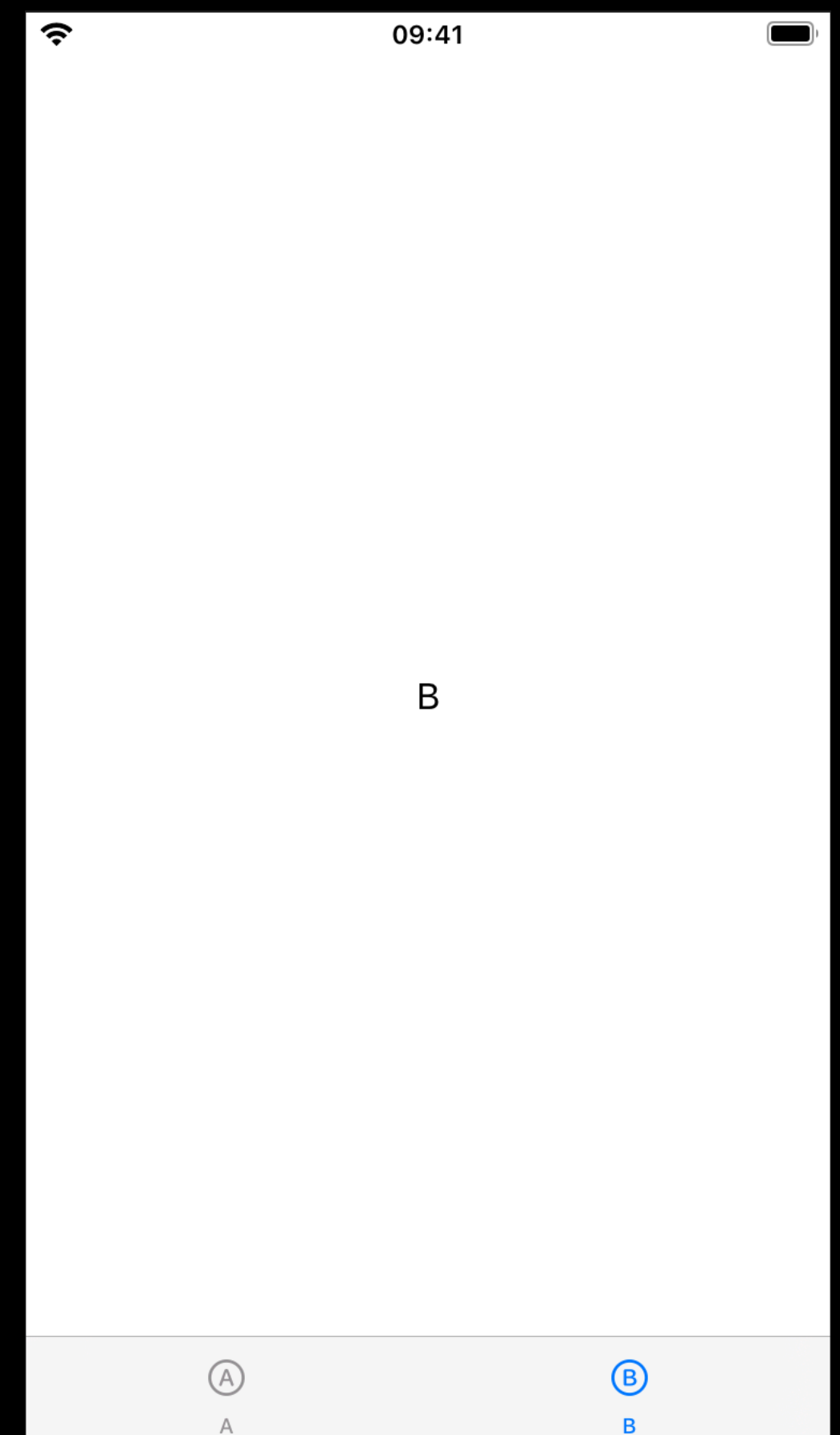
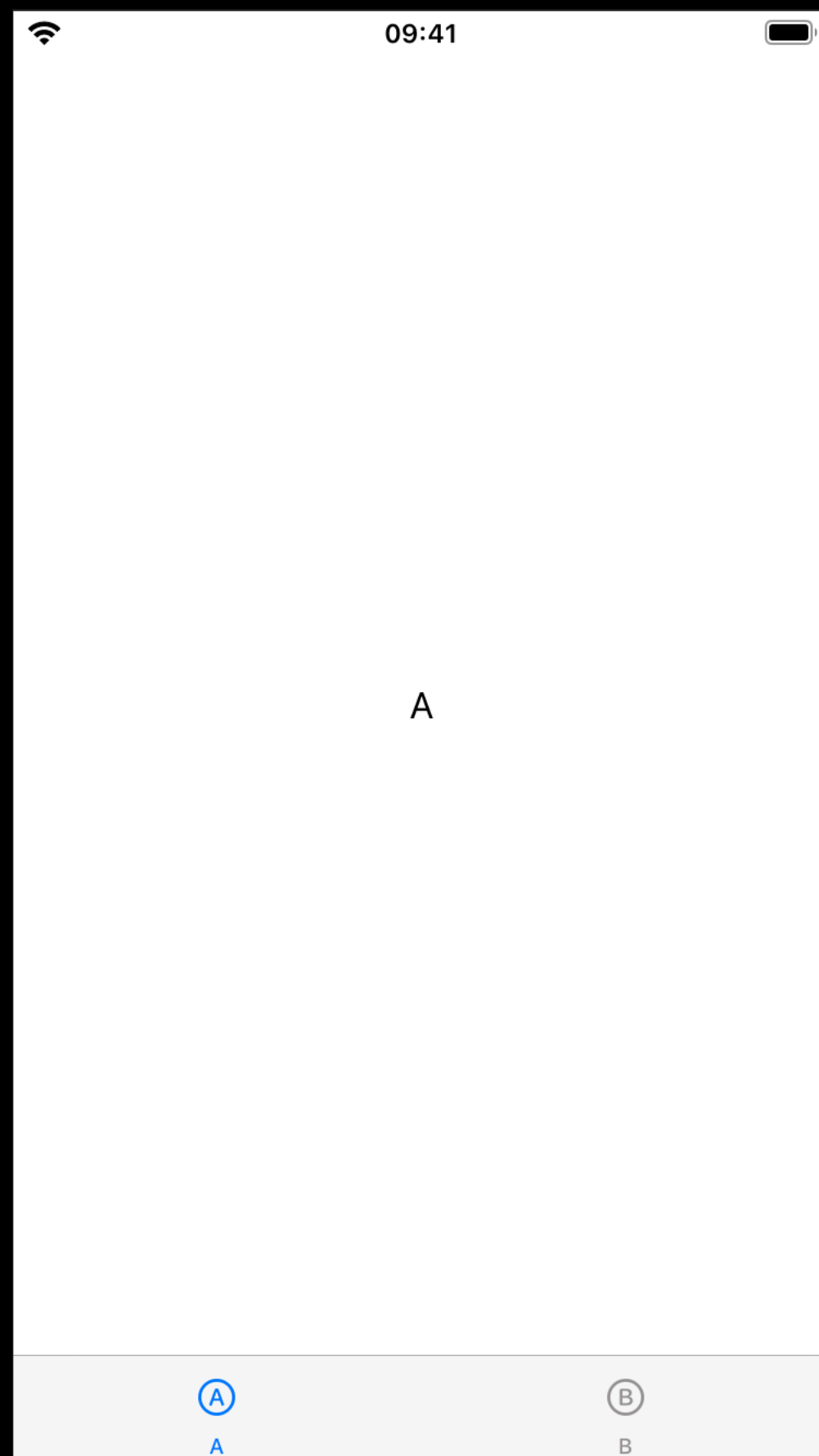


```
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")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

```

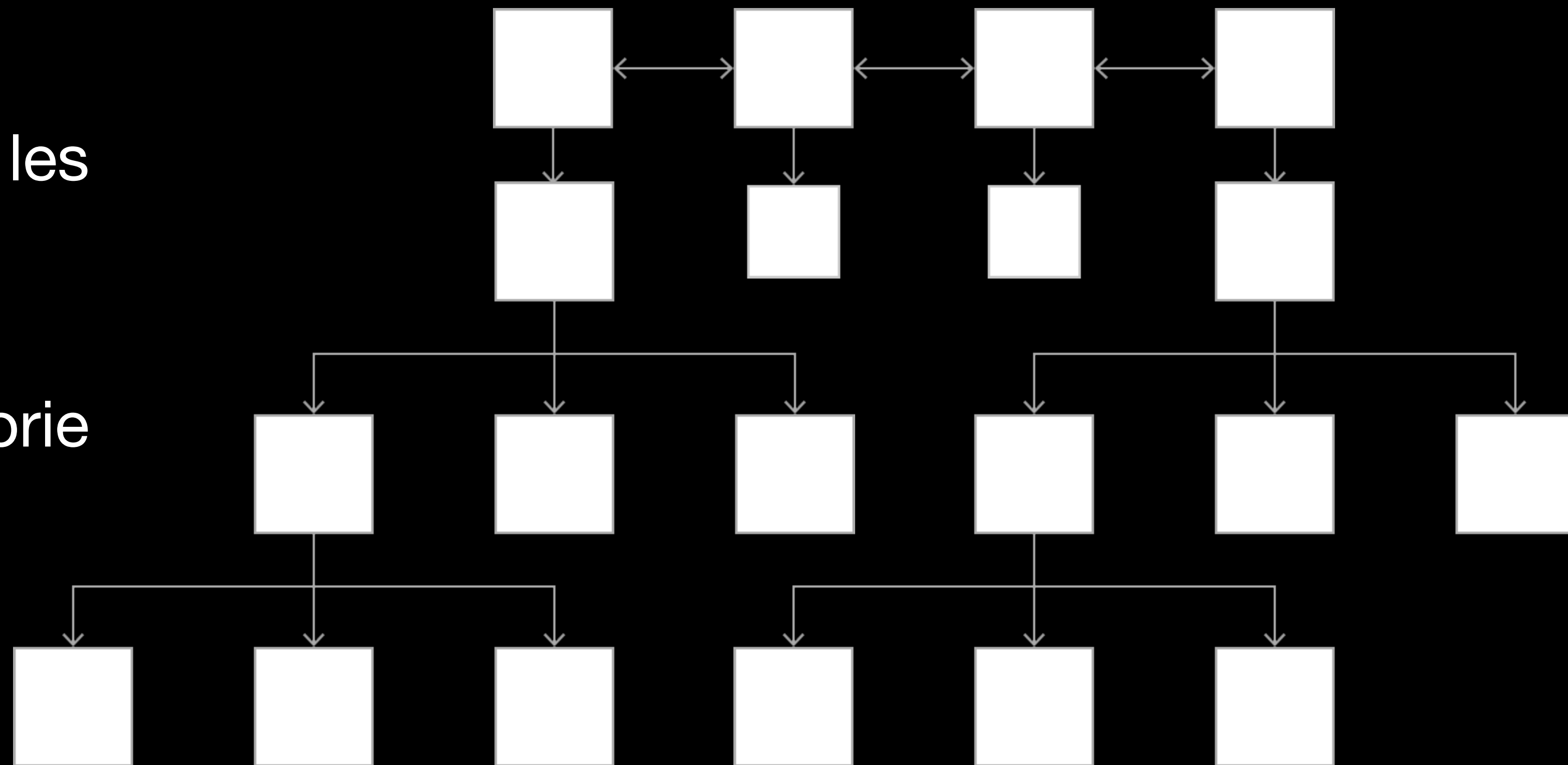
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")
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

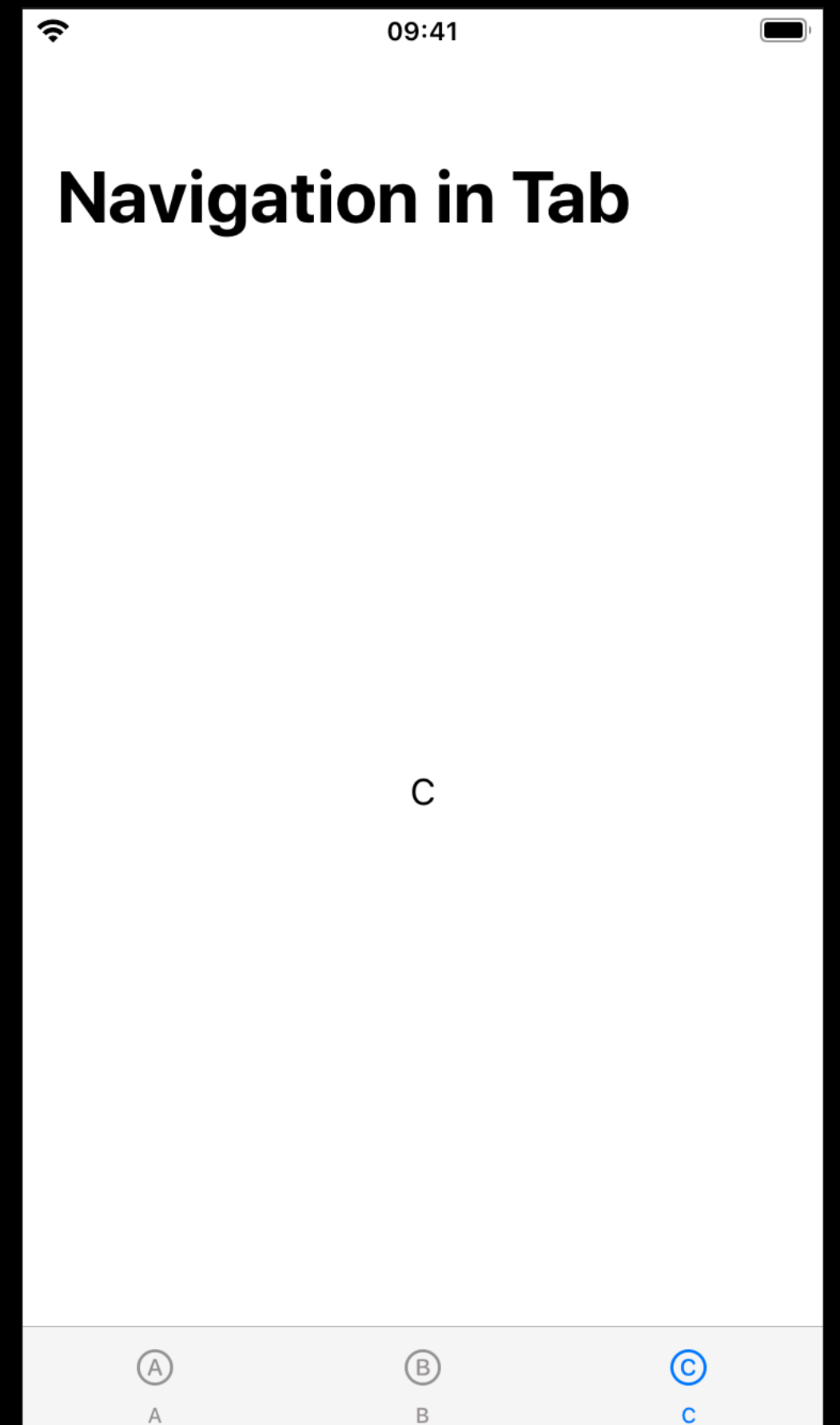


```
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")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

```

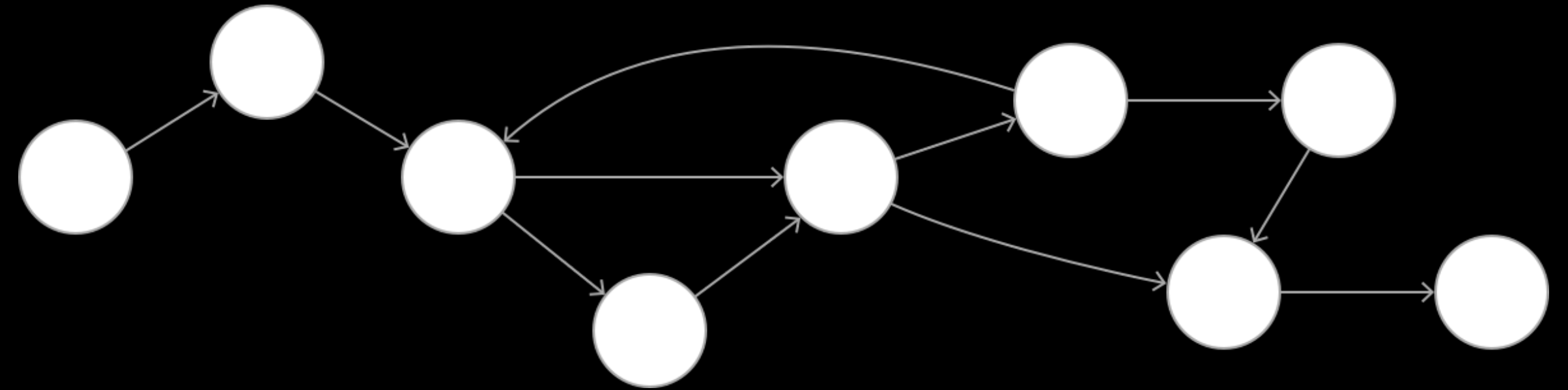
struct InitialTabView: View {
    var body: some View {
        TabView {
            Text("A")
                .tabItem {
                    Text("A")
                    Image(systemName: "a.circle")
                }
            Text("B")
                .tabItem {
                    Text("B")
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

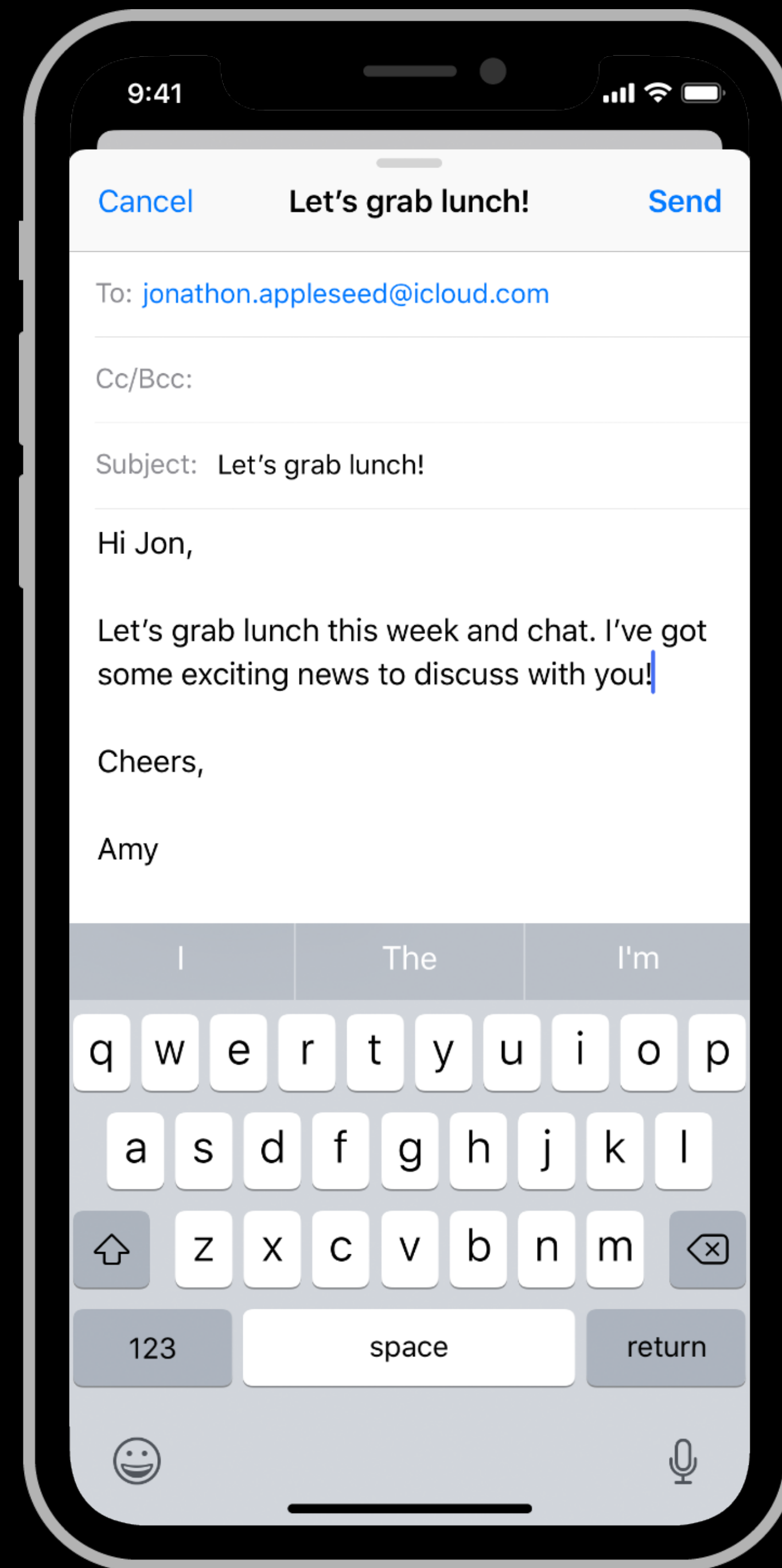
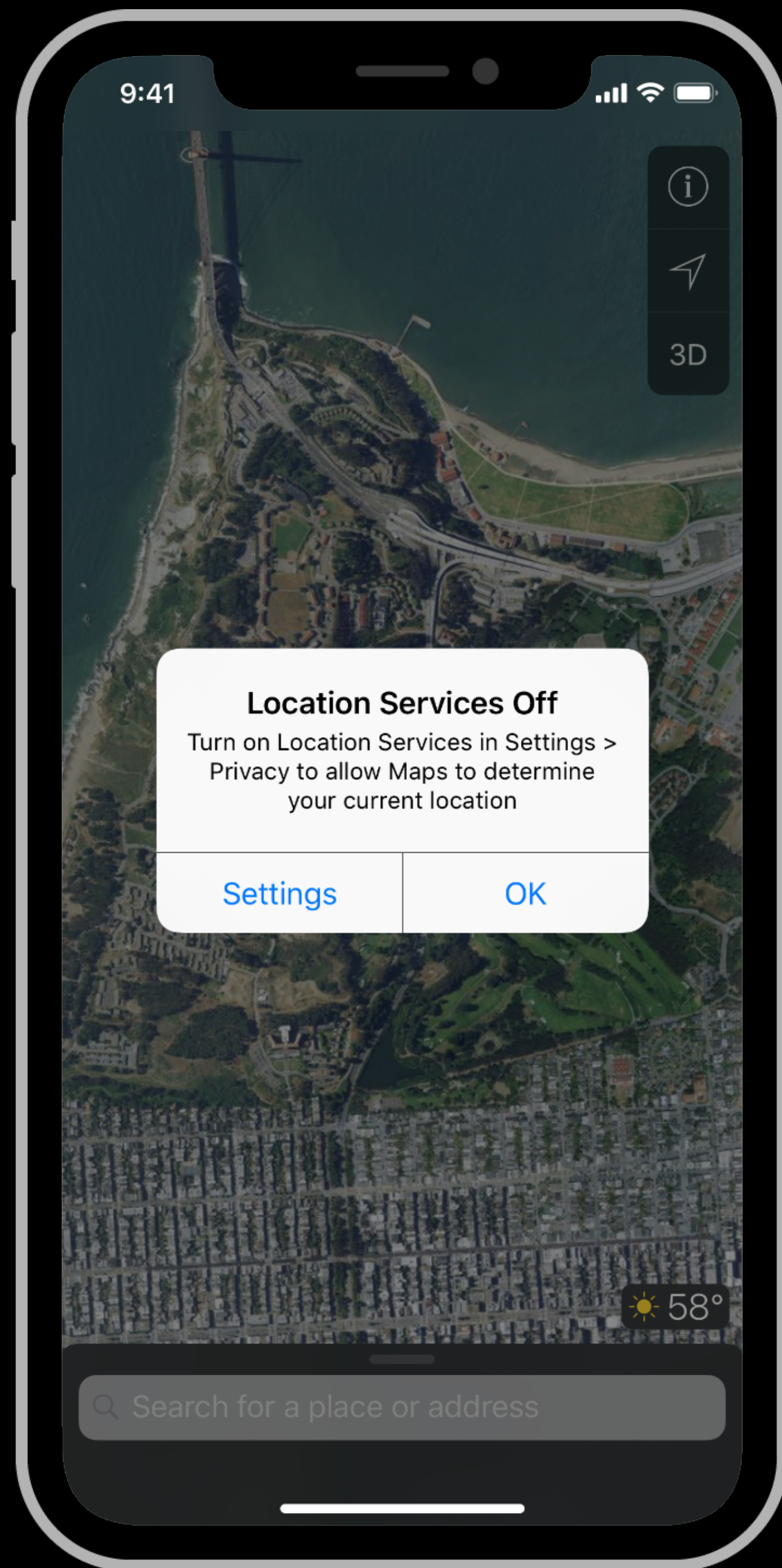


# Modalité

**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**



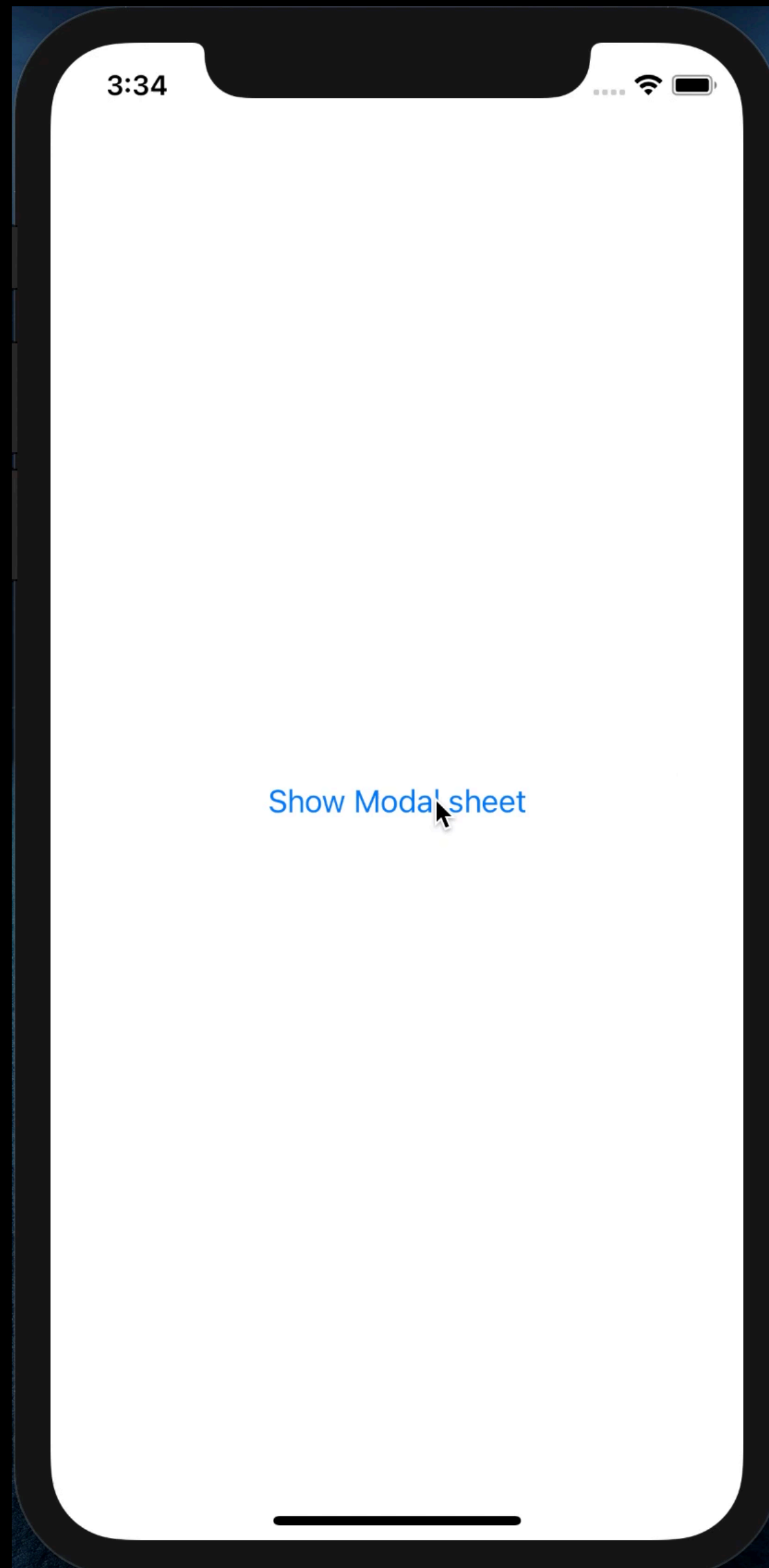


# Modalité

- 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(isPresented: $isDisplayingAlert) {  
            Alert(title: Text("This is an alert"))  
        }  
    }  
}
```



3:41



This is an alert

OK

# Alerte

```
struct Modality: View {  
    @State private var isDisplayingAlert = false  
  
    var body: some View {  
        Button("Show alert") {  
            self.isDisplayingAlert.toggle()  
        }.alert(isPresented: $isDisplayingAlert) {  
            Alert(title: Text("This is an alert"))  
        }  
    }  
}
```

# Action

```
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

```
view {  
  
    var isDisplayingActionSheet = false  
  
    View {  
  
        show Action sheet") {  
            isDisplayingActionSheet.toggle()  
            ActionSheet(isPresented: $isDisplayingActionSheet) {  
                ActionSheet(title: Text("Action title"), message: Text("Action message"),  
                    destructive(Text("Destructive action"), action: {  
                        // ...  
                    },  
                    default(Text("Default action"), action: {  
                        // ...  
                    },  
                    cancel()))  
        }  
    }  
}
```

3:42



Show Action sheet

Action title

Action message

Destructive action

Default action

Cancel