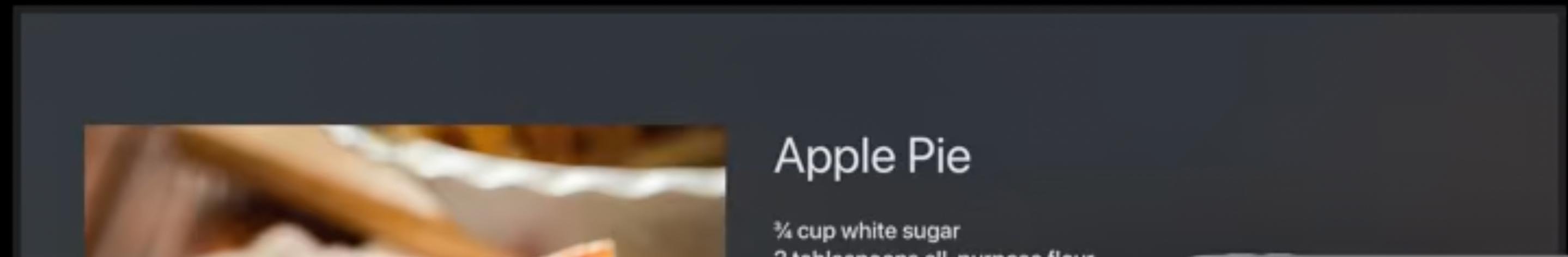


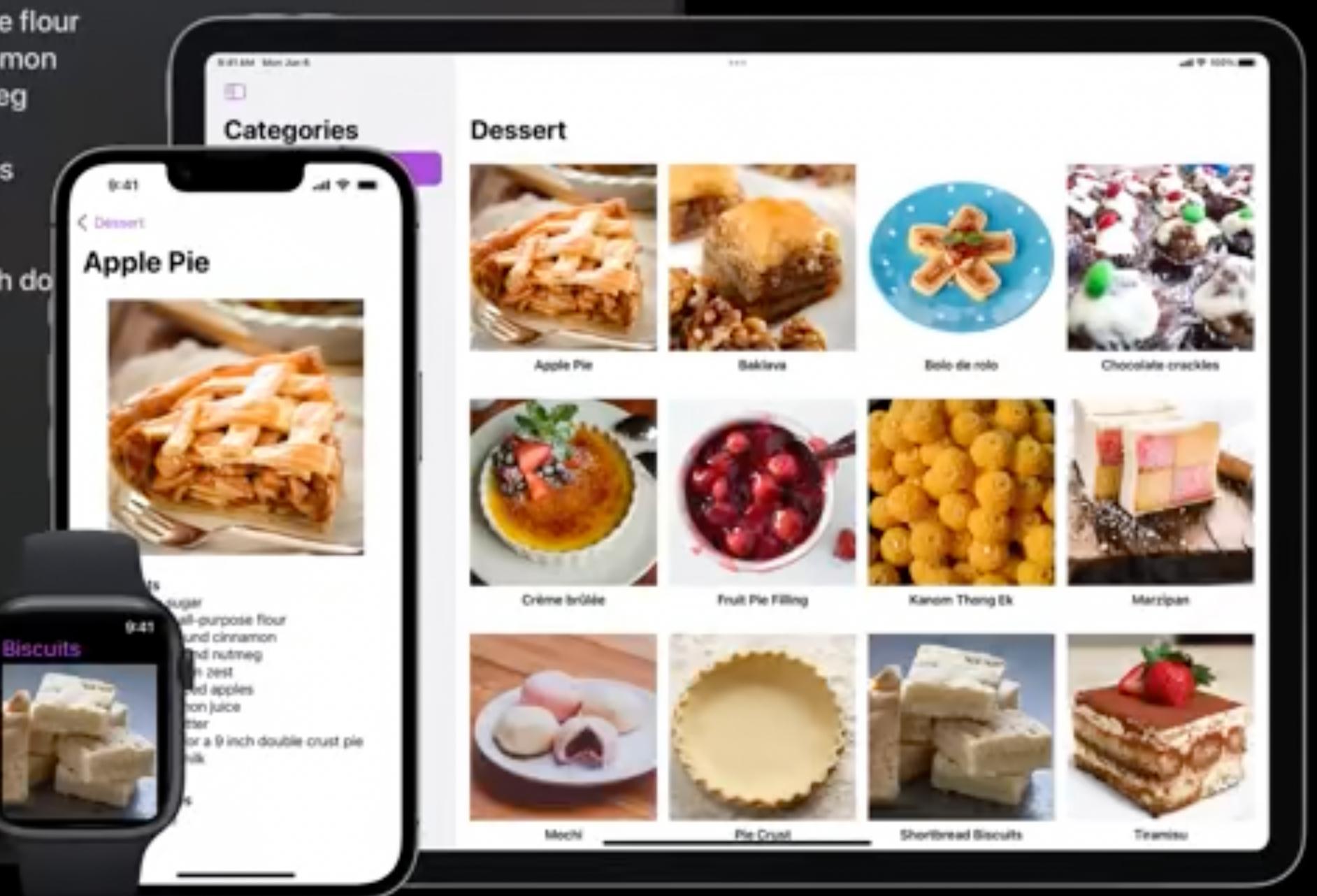
The Quimera: SwiftUI Navigation

FrenchKit 2022

@lascorbe



¾ cup white sugar
2 tablespoons all-purpose flour
½ teaspoon ground cinnamon
¼ teaspoon ground nutmeg
½ teaspoon lemon zest
7 cups thinly sliced apples
2 teaspoons lemon juice
1 tablespoon butter
1 recipe pastry for a 9 inch double crust pie
4 tablespoons milk



Luis Ascorbe

@lascorbe

Agenda



Photo by [Marissa Grootes](#) on [Unsplash](#)

Agenda

- Navigation with UIKit



Photo by [Marissa Grootes](#) on [Unsplash](#)

Agenda

- Navigation with UIKit
- Navigation with SwiftUI



Photo by [Marissa Grootes](#) on [Unsplash](#)

Agenda

- Navigation with UIKit
- Navigation with SwiftUI
- Which one to choose



Photo by [Marissa Grootes](#) on [Unsplash](#)

Navigation with UIKit

UIHostingController<Content>

```
class UIHostingController<Content> where Content: View

import SwiftUI

struct FrenchView: View {
    var body: some View {
        Text("Bonjour")
    }
}
```

```
class UIHostingController<Content> where Content: View

import SwiftUI

struct FrenchView: View {
    var body: some View {
        Text("Bonjour")
    }
}

UIHostingController(rootView: FrenchView())
```

```
class UIHostingController<Content> where Content: View

import SwiftUI

struct FrenchView: View {
    @ObservedObject var viewModel: ViewModel
    var body: some View {
        Text("Bonjour")
    }
}

UIHostingController(
    rootView: FrenchView(viewModel: ViewModel()))
)
```

```
class UIHostingController< UIViewController> {
    import SwiftUI
}

struct FrenchView: View {
    @ObservedObject var viewModel: ViewModel
    var body: some View {
        Text("Bonjour")
    }
}

UIHostingController(
    rootView: FrenchView(viewModel: ViewModel())
)
```

```
class ViewModel: ObservableObject {
}
```

```
class UIHostingController< UIViewController> {
    var view: View? {
        didSet {
            if let view = view {
                self.view = view.uiView
            }
        }
    }
}

class FrenchView: View {
    @ObservedObject var viewModel: ViewModel
    var body: some View {
        Text(viewModel.title)
    }
}

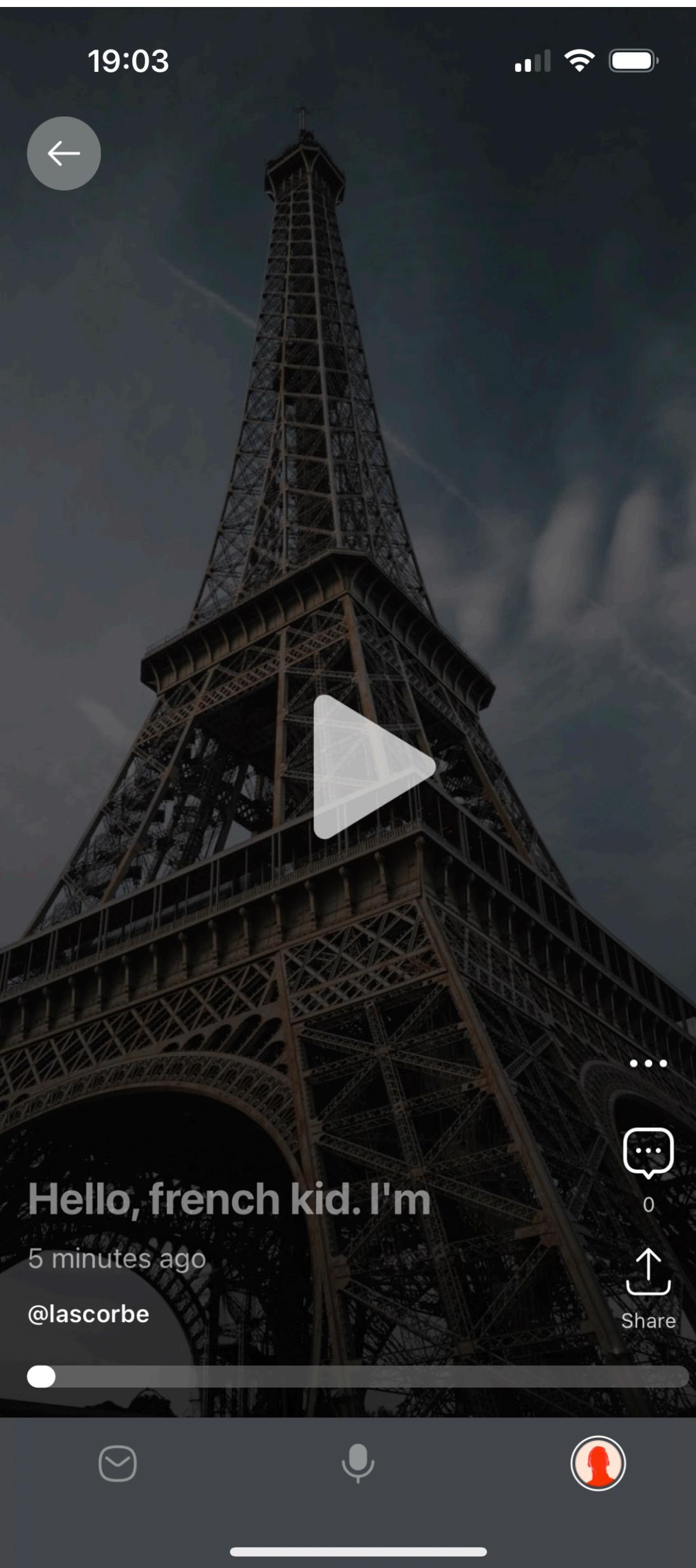
UIHostingController(
    rootView: FrenchView(viewModel: ViewModel()))

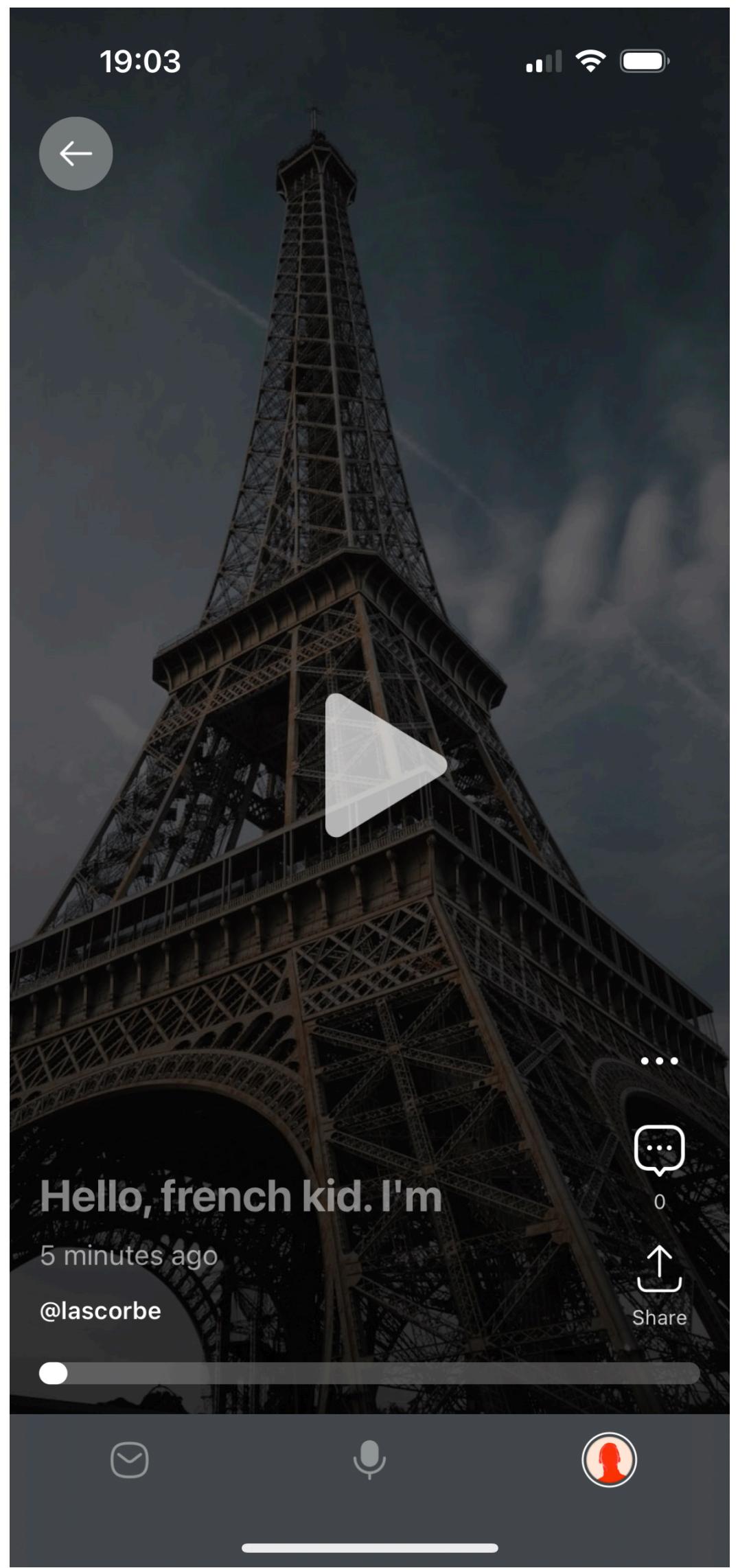
```

```
class ViewModel: ObservableObject {
    @Published var title: String
}
```

Real life use case

beams.fm





19:03



Share

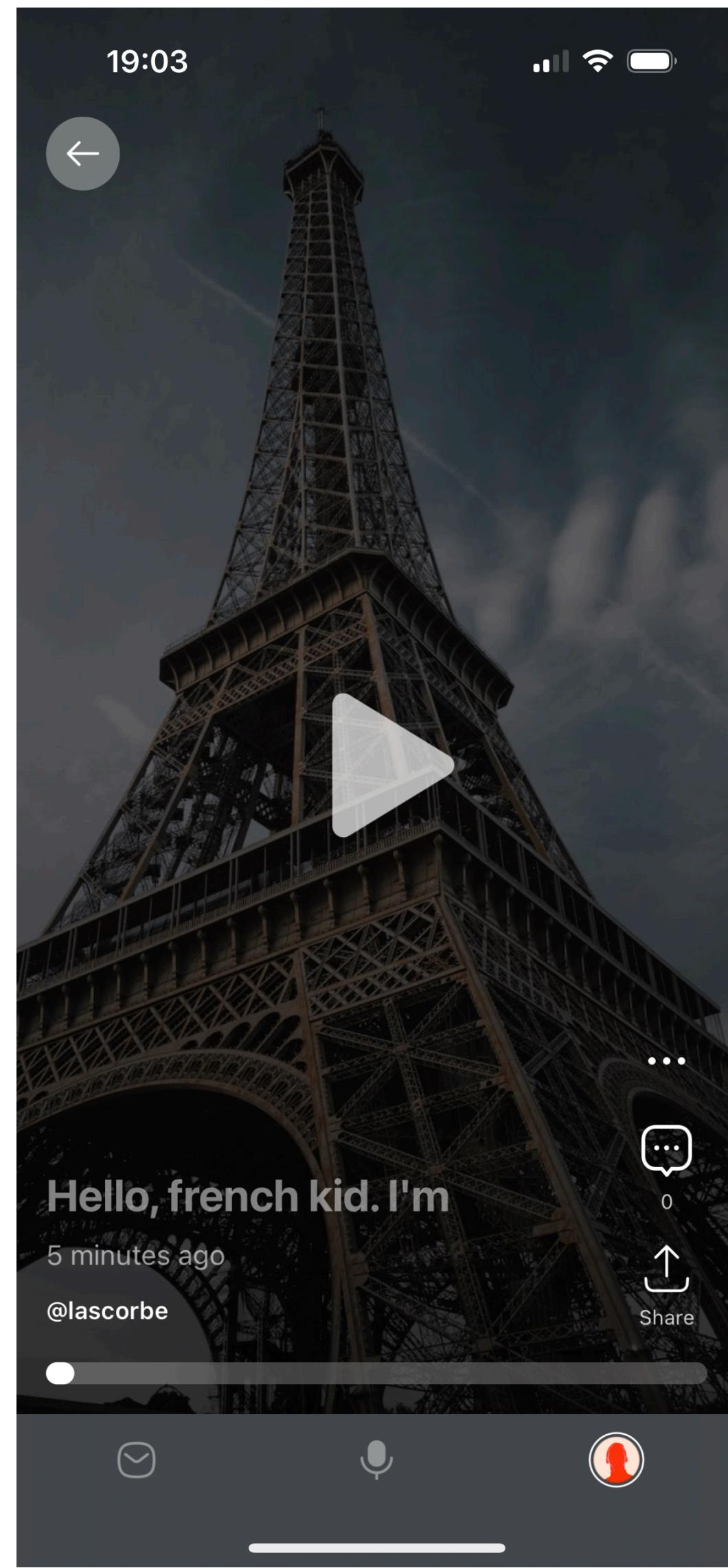
Hello, french kid. I'm

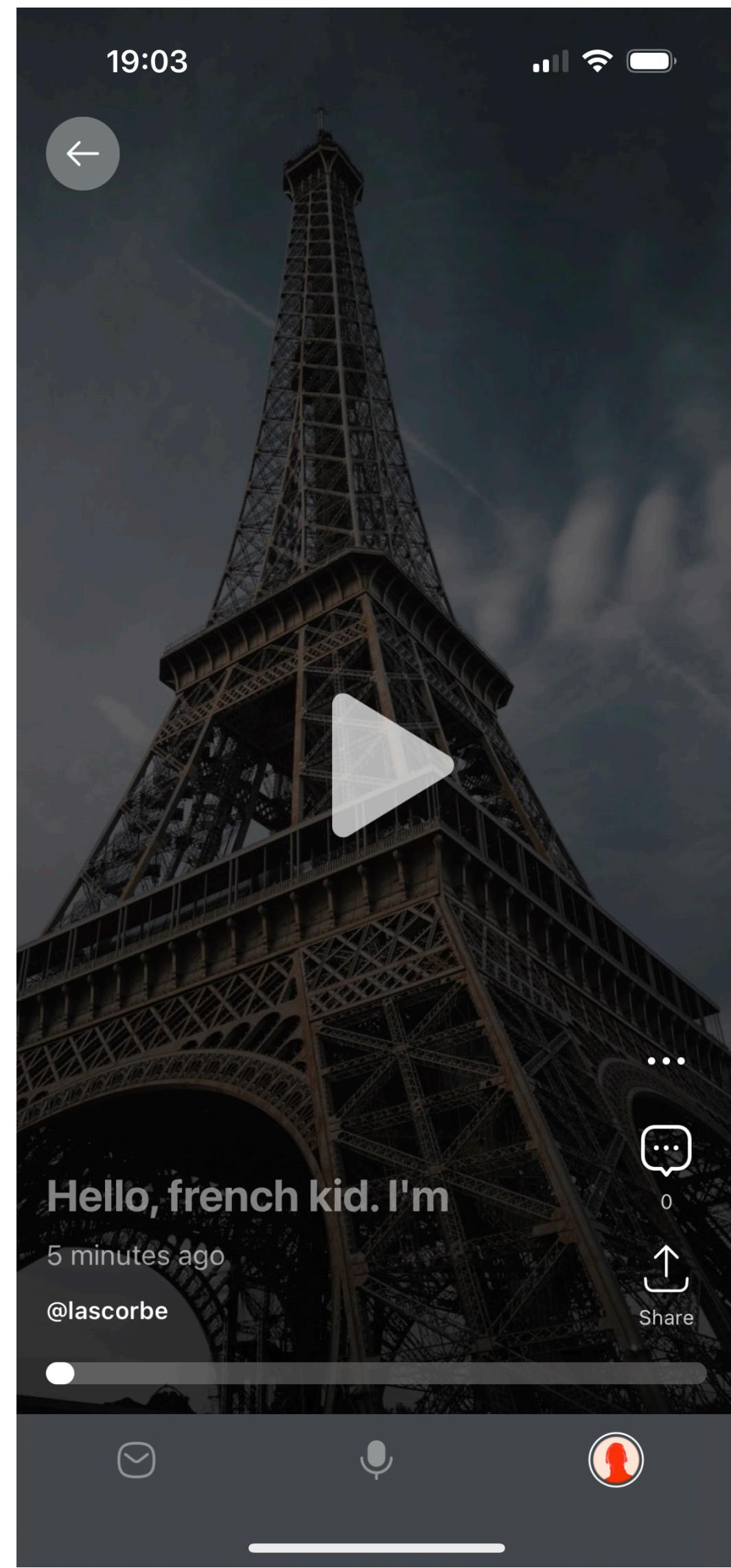
5 minutes ago

@lascorbe



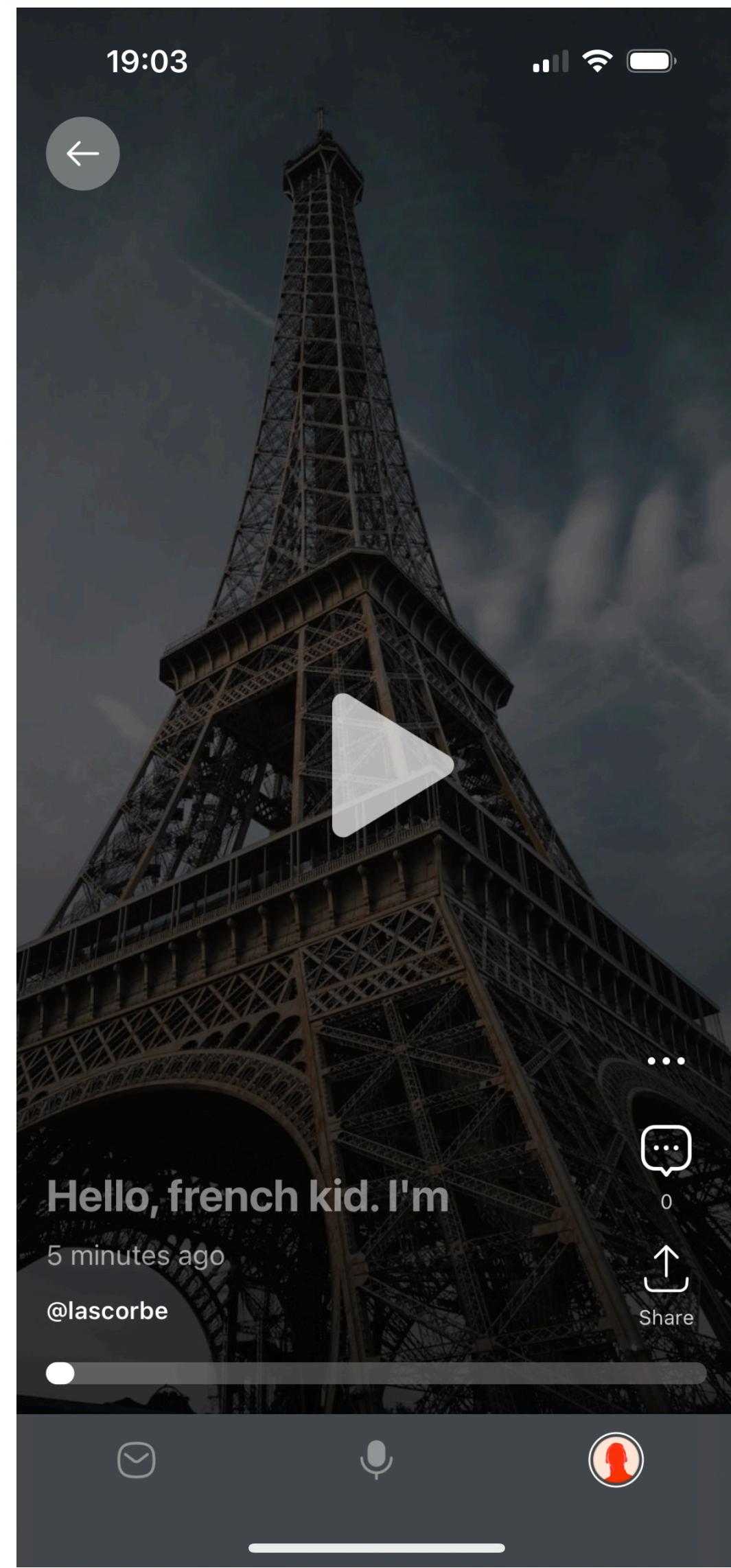
UITabBarController





UITabBarController

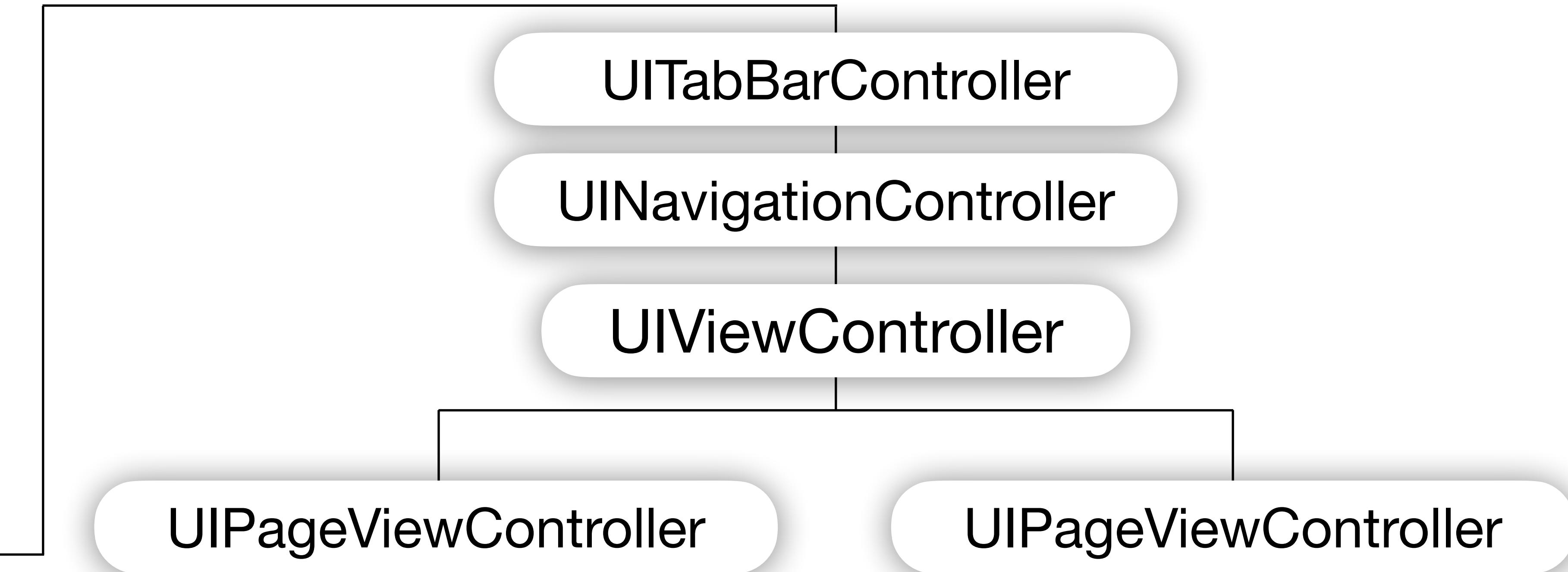
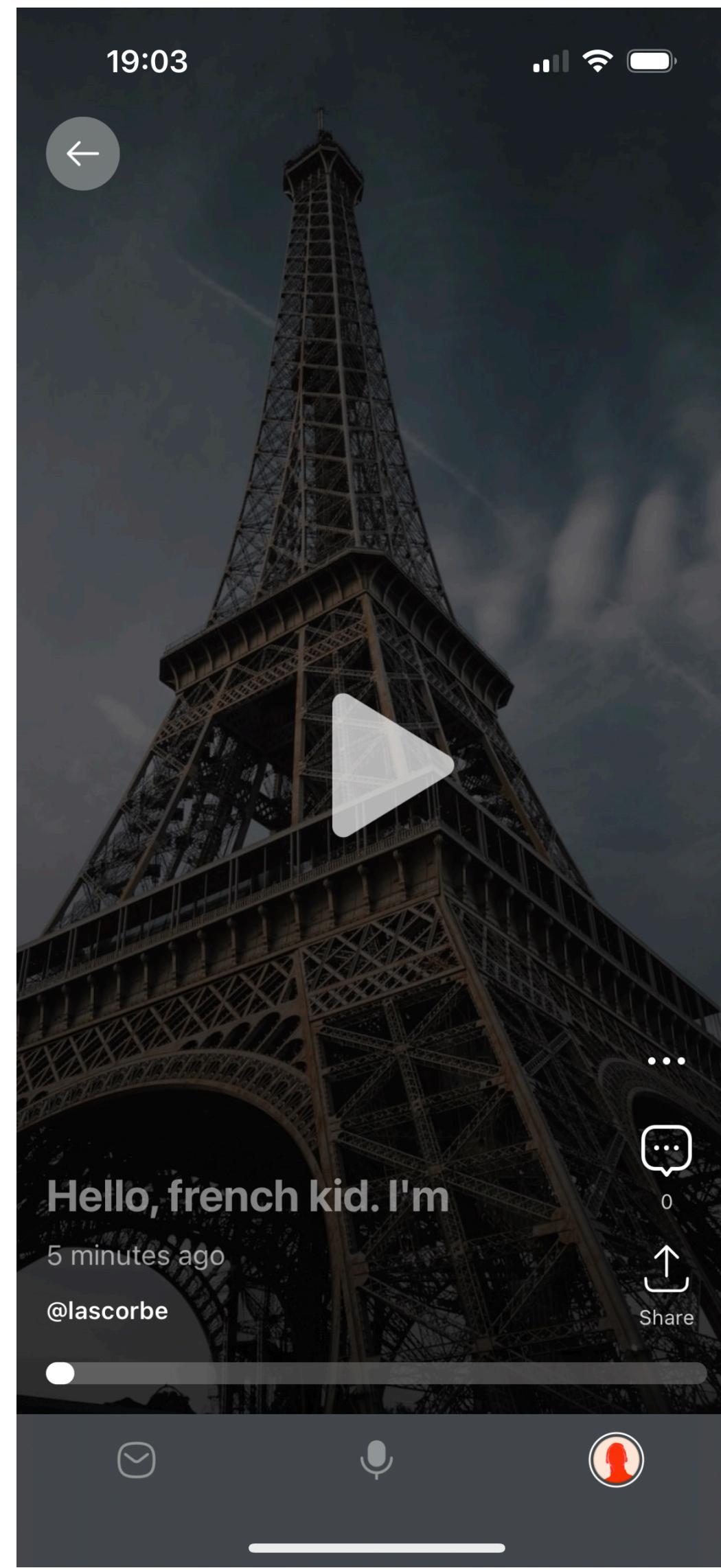
UINavigationController

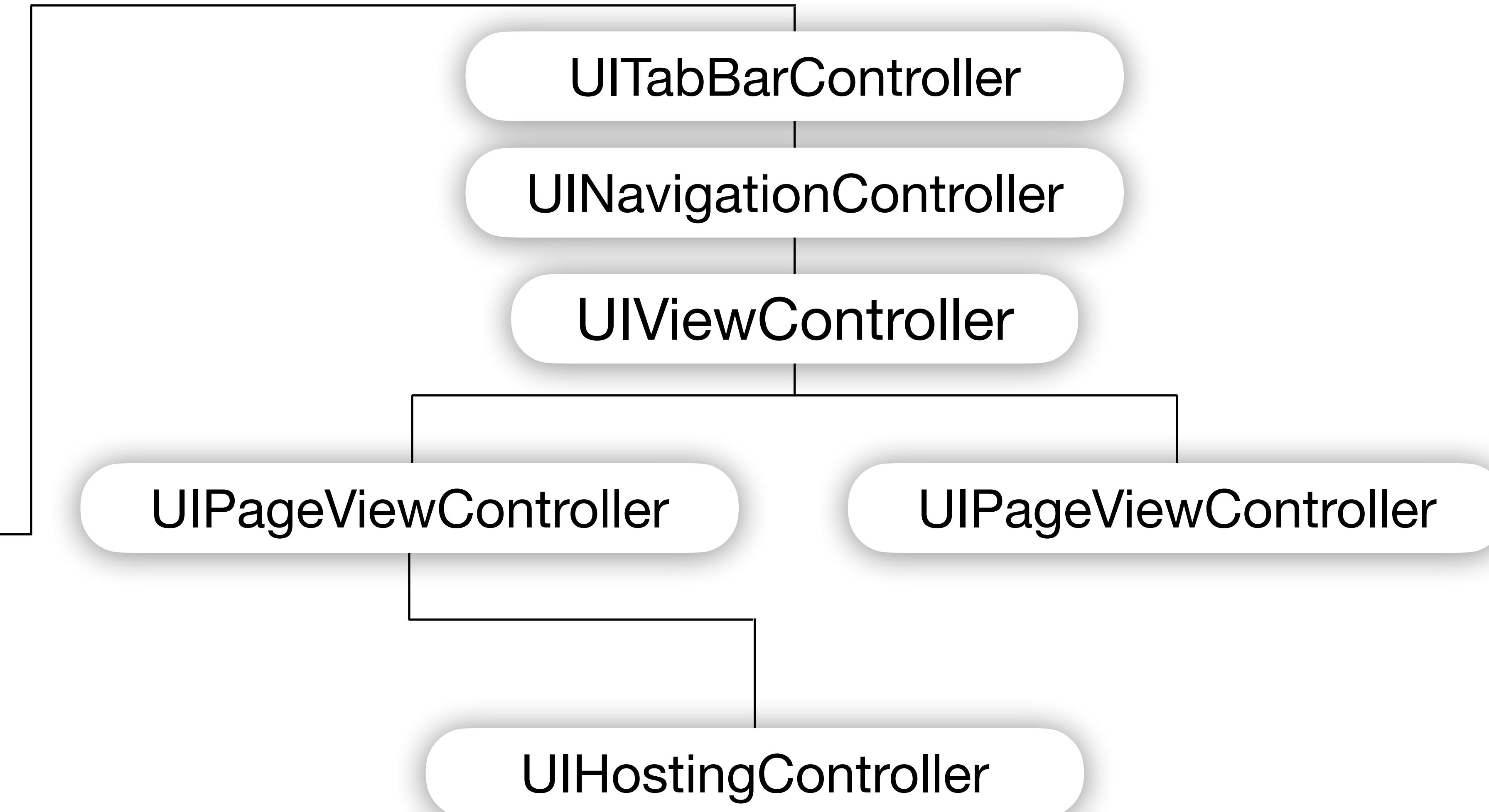
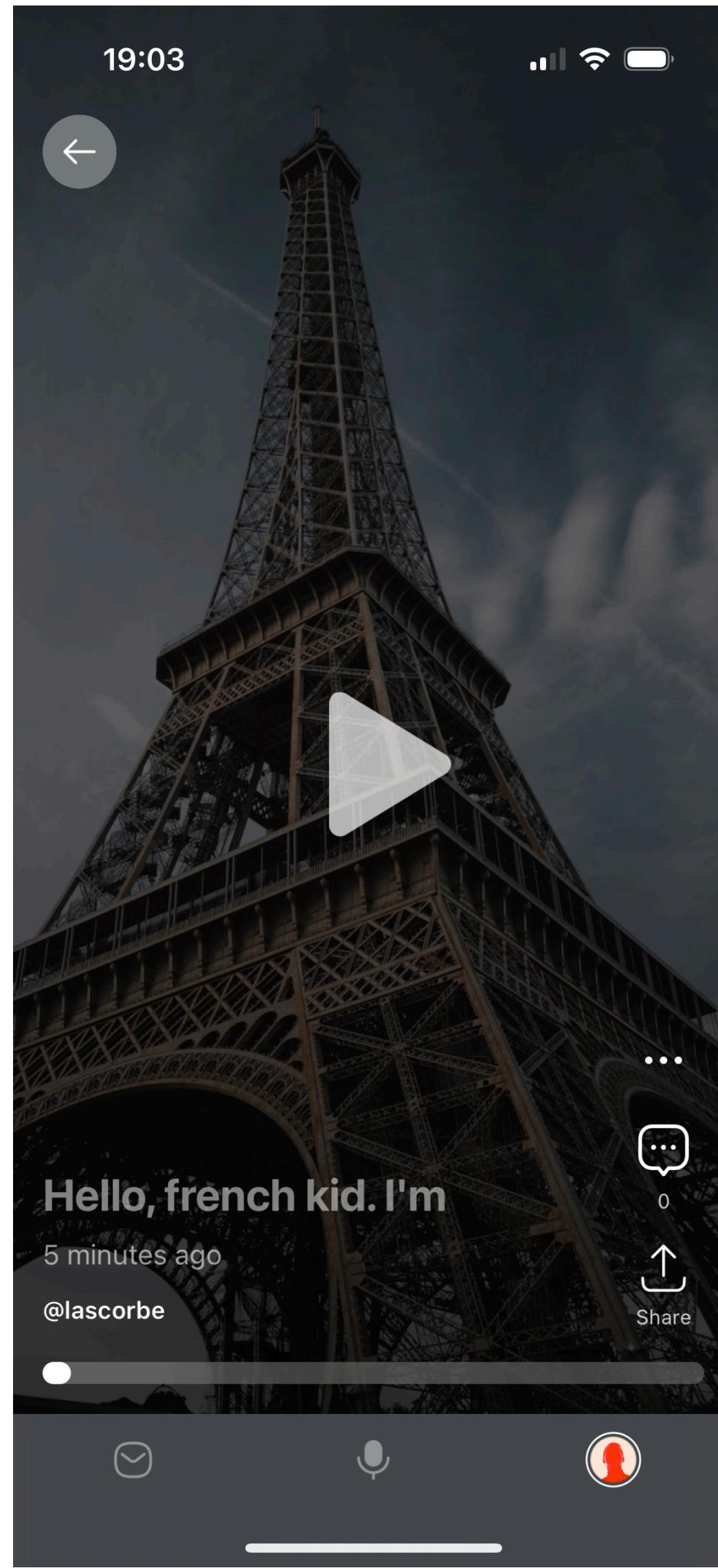


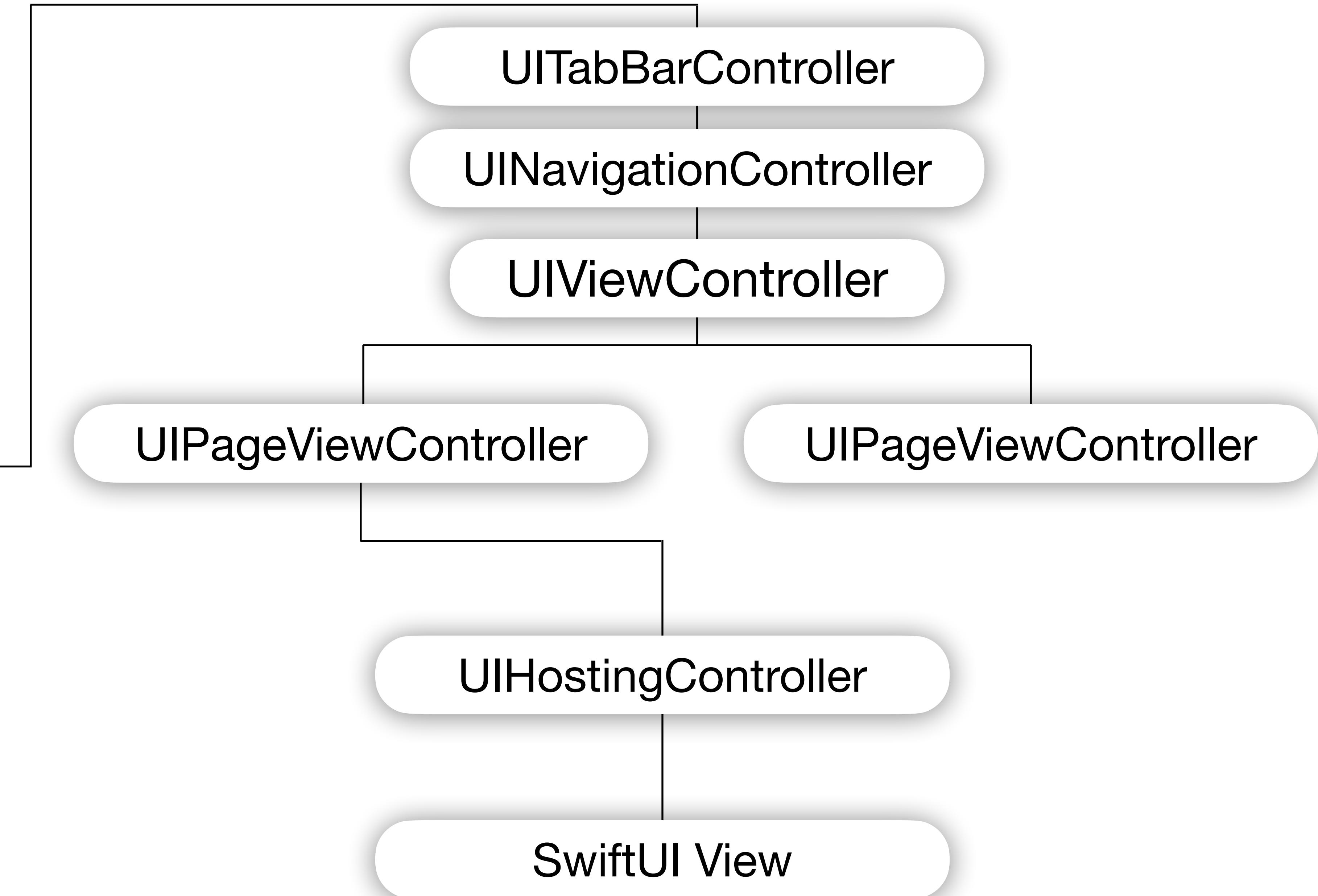
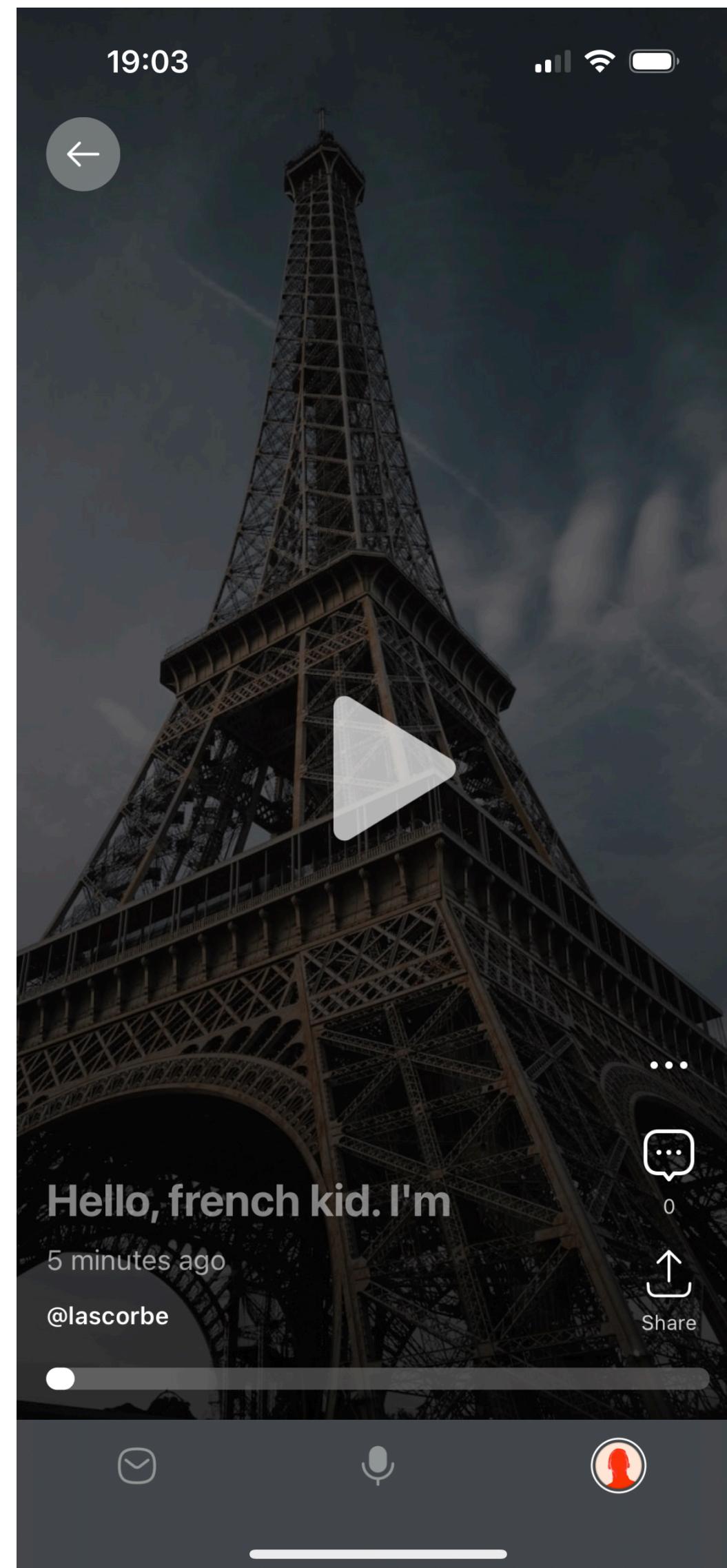
UITabBarController

UINavigationController

UIViewController







Why?

Why?

- All our navigation was working with UIKit

Why?

- All our navigation was working with UIKit
- No vertical page view on SwiftUI

Why?

- All our navigation was working with UIKit
- No vertical page view on SwiftUI
- Development on SwiftUI is a joy

IntrinsicSize-SwiftUI

Running IntrinsicSize-SwiftUI on iPhone 13

Main.storyboard (Base) RootViewController.swift SwiftUIView.swift SwiftUIViewController.swift View hierarchy for IntrinsicSize-SwiftUI

CPU 0% Memory 13 MB Disk 8 KB/s Network Zero KB/s

Object Class Name _UIHostingView<SwiftUIView> Address 0x7fee14f09a30

View Layer <CALayer: 0x60000090afc0> Layer Class CALayer Content Mode Scale To Fill Tag 0 Interaction User Interaction Enabled On Multiple Touch Off Alpha 1 Background R:0 G:0 B:1 A:1 blueColor Tint R:0 G:0,48 B:1 A:1 systemBlueColor

Drawing Opaque On Hidden Off Clears Graphics Context On Clip To Bounds Off AutoresizesSubviews On

Stretching x 0 y 0 width 1 height 1

Trait Collection Light User Interface Style Regular Vertical Size Class Compact Horizontal Size Class Left To Right Layout Direction

Accessibility Not Accessibility Element Value <null> Traits None Elements <null> Description <null> Hint <null> Identifier <null> Actions <null> Not Focused

Description <TtGC7SwiftUI14_UIHostingViewV21IntrinsicSize_SwiftUI11s swiftUIview_: 0x7fee14f09a30; frame = (0 691.667; 390 69.3333); gestureRecognizers = <NSArray: 0x600000757660>; layer = <CALayer: 0x60000090afc0>>

Hierarchy _UIHostingView<SwiftUIView> UIView UIResponder NSObject

Backtrace Malloc stack logging is not enabled for this process.

All Output

```
viewDidLoad(): Bounds: (0.0, 0.0, 0.0, 0.0)
viewDidLoad(): IntrinsicContentSize: (245.66666666666666, 22.33333333333332)
viewWillAppear(_:): Bounds: (0.0, 0.0, 0.0, 0.0)
viewWillAppear(_:): IntrinsicContentSize: (245.66666666666666, 22.33333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
viewWillAppear(_:): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewWillAppear(_:): IntrinsicContentSize: (390.0, 22.33333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.33333333333332)
(lldb)
```

```
final class HostingController<Content: View>: UIHostingController<Content> {
    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        view.setNeedsUpdateConstraints()
    }
}
```

```
final class HostingController<Content: View>: UIHostingController<Content> {
    override func viewDidLayoutSubviews() {
        super.viewDidLayoutSubviews()
        view.setNeedsUpdateConstraints()
    }
}
```

Navigation with SwiftUI

Navigation with SwiftUI

.popover

NavLink

.navigationDestination

NavigationView

NavigationStack

NavigationSplitView

.confirmationDialog

.alert

.sheet

.fullScreenCover

Navigation with SwiftUI

.popover

NavLink

NavigationView

NavigationStack

.confirmationDialog

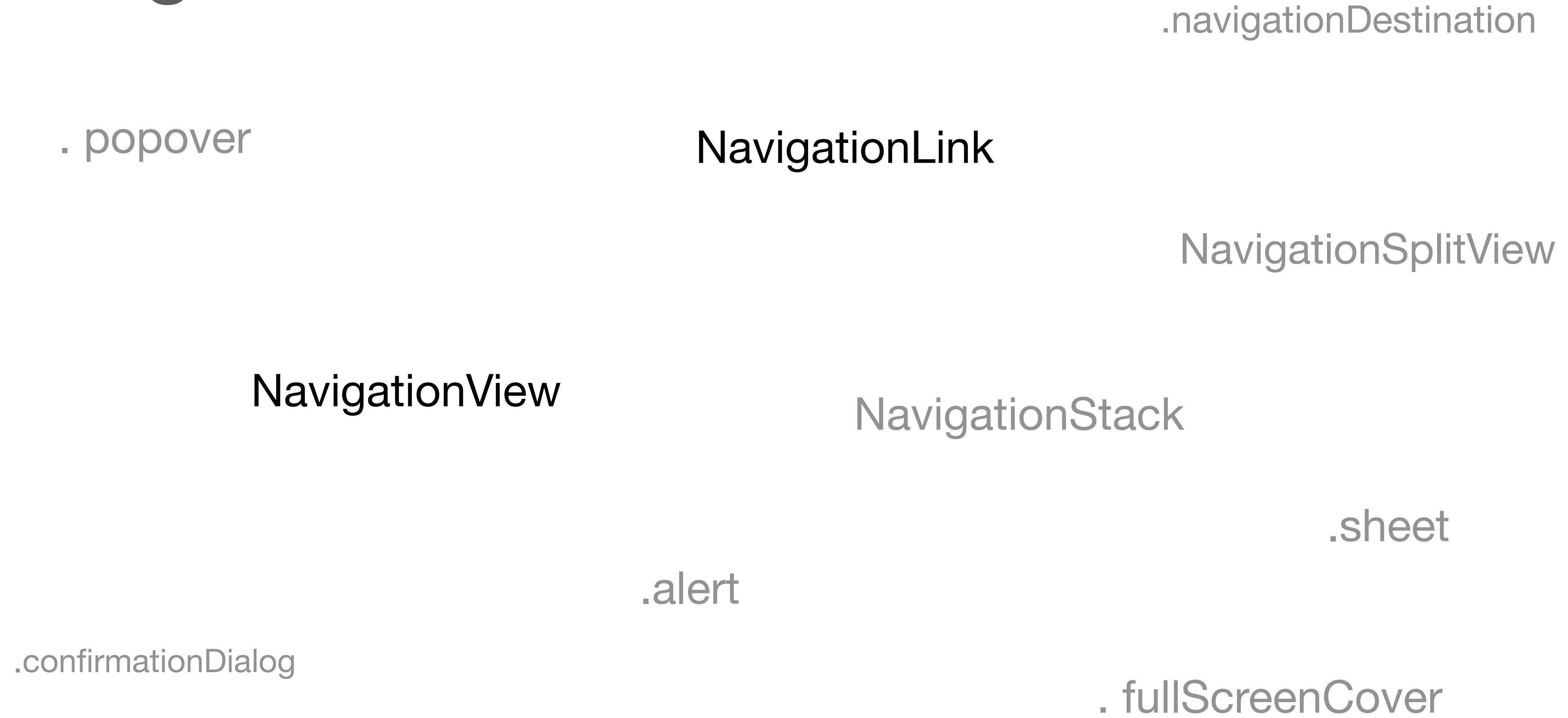
.alert

.sheet

.fullScreenCover

.navigationDestination

Navigation with SwiftUI



Navigation with SwiftUI

.popover

NavigationLink

.navigationDestination



.confirmationDialog

.alert

NavigationStack

.sheet

.fullScreenCover

Navigation with SwiftUI

.popover

NavigationLink

.navigationDestination



NavigationStack

NavigationSplitView

.confirmationDialog

.alert

.sheet

.fullScreenCover

NavLink

```
struct FrenchView: View {  
    var body: some View {  
        NavigationStack { // (prev NavigationView)  
            NavigationLink("Push view") {  
                Text("Bonjour")  
            }  
        }  
    }  
}
```

NavLink

```
struct FrenchView: View {  
    var body: some View {  
        NavigationStack { // (prev NavigationView)  
            NavigationLink("Push view") {  
                BonjourView()  
            }  
        }  
    }  
}
```

```
struct BonjourView: View {  
    init() { print("init") }  
    var body: some View {  
        Text("Bonjour")  
    }  
}
```

NavLink

```
struct FrenchView: View {  
    var body: some View {  
        NavigationStack { // (prev NavigationView)  
            NavigationLink("Push view") {  
                BonjourView()  
            }  
        }  
    }  
}
```

```
struct BonjourView: View {  
    init() { print("init") }  
    var body: some View {  
        Text("Bonjour")  
    }  
}
```

NavigationLink

struct
var

The screenshot shows a blog post on a website. The title is "MVP + Coordinators in SwiftUI (part 1)". Below the title is the date "April 27, 2020" and a duration of "15 minutes". A series of tags are listed: swiftui, coordinator, mvp, article, series, and part1. The main content area features a large image of a laptop displaying Xcode code. The code is for a Swift protocol named BaseCoordinator, which includes methods for getting and setting associated objects based on identifiers. The laptop is positioned against a brick wall.

Luis Ascorbe

Software Developer. Tech Lead. Speaker.
NSSpain Organizer.

Email GitHub Twitter LinkedIn

Blog About

```
1 // Created by Luis Ascorbe on 23/04/2020.
2 // Copyright © 2020 Luis Ascorbe. All rights reserved.
3
4 import Foundation
5
6 protocol BaseCoordinator: AssociatedObject {
7     associatedtype View
8     associatedtype P: BaseCoordinator
9     func start() -> View
10 }
11
12 extension BaseCoordinator { // Mixin Extension: Check out AssociatedObject.swift
13     fileprivate var identifier: UUID {
14         get {
15             guard let identifier: UUID = associatedObject(for: IdentifierKey) else {
16                 self.identifier = UUID()
17                 return self.identifier
18             }
19             return identifier
20         }
21     }
22     set {
23         setAssociatedObject(newValue, for: IdentifierKey)
24     }
25 }
26
27 var parent: P? {
28     get { associatedObject(for: ParentKey) }
29     set { setAssociatedObject(newValue, for: ParentKey) }
30 }
31
32 func coordinate< T: BaseCoordinator>(coordinator: T) -> some View {
33     coordinator.parent = self as! P
34     return coordinator.start()
35 }
36
37 private var IdentifierKey: UUID = UUID()
38 private var parentKey: UUID = UUID()
```

}

}

NavigationStack

and **.navigationDestination(for:, destination:)**

NavigationStack

```
NavigationStack(path: $viewModel.path) {  
    List {  
        NavigationLink(value: Destination.bonjour) {  
            Text("To Bonjour")  
        }  
    }  
.navigationDestination(for: Destination.self) { destination in  
    switch destination {  
        case .bonjour:  
            BonjourView()  
    }  
}  
}
```

NavigationStack

```
NavigationStack(path: $viewModel.path) {  
    List {  
        NavigationLink(value: Destination.bonjour) {  
            Text("To Bonjour")  
        }  
    }  
    .navigationDestination(for: Destination.self) { destination in  
        switch destination {  
            case .bonjour:  
                BonjourView()  
        }  
    }  
}
```

NavigationStack

```
NavigationStack(path: $viewModel.path) {  
    List {  
        NavigationLink(value: Destination.bonjour) {  
            Text("To Bonjour")  
        }  
    }  
    .navigationDestination(for: Destination.self) { destination in  
        switch destination {  
            case .bonjour:  
                BonjourView()  
        }  
    }  
}
```

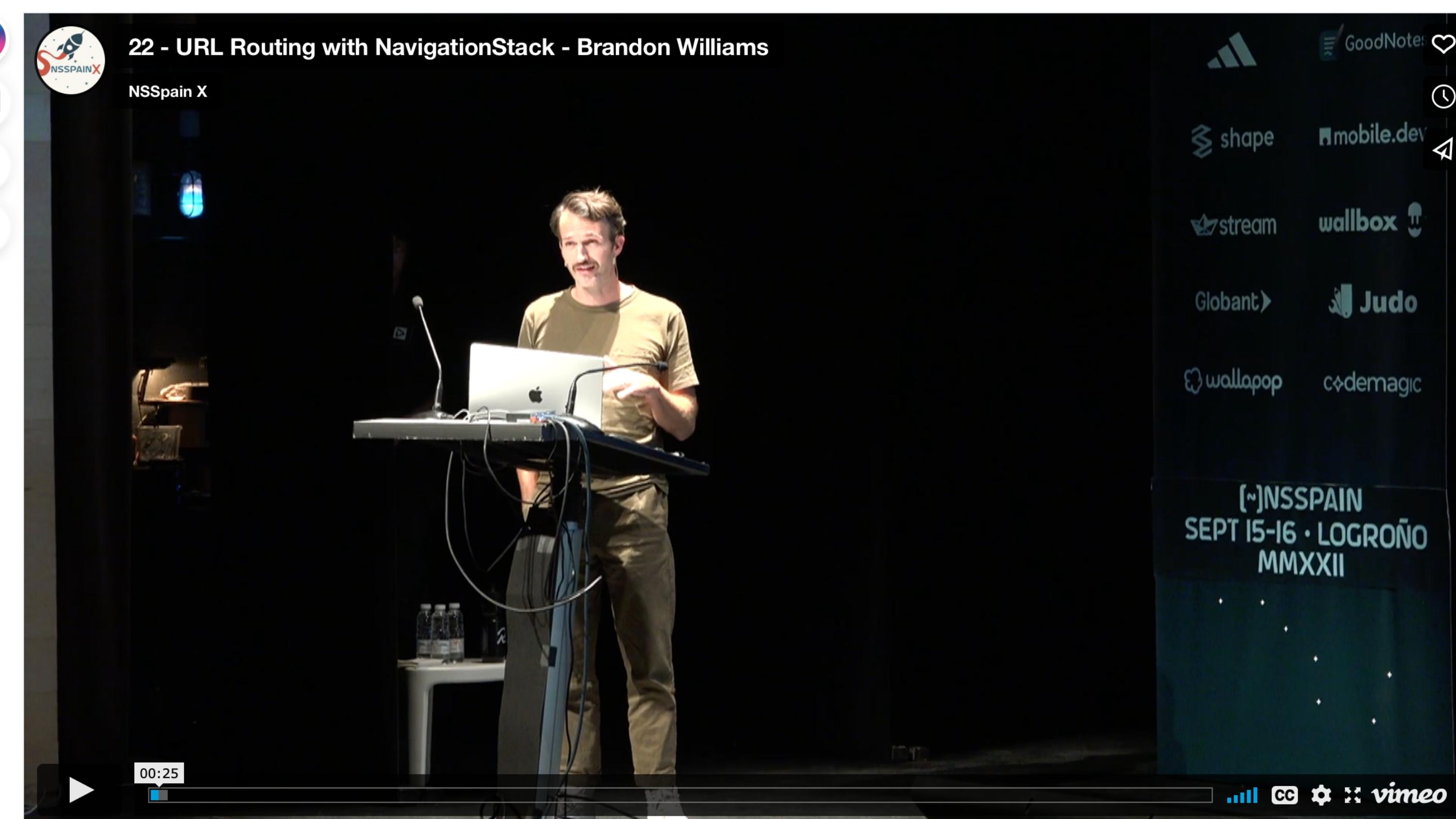
NavigationStack

```
NavigationStack(path: $viewModel.path) {  
    List {  
        NavigationLink(value: .bonjour)  
            Text("To Bonjour")  
    }  
    .navigationDestination(for: Destination.self) { destination in  
        switch destination {  
            case .bonjour:  
                BonjourView()  
        }  
    }  
}
```

```
enum Destination: Hashable {  
    case bonjour  
}  
class ViewModel: ObservableObject {  
    @Published var path: [Destination]  
}
```

Brandon Williams

@mbrandonw



A screenshot of a Vimeo video player. The video is titled "22 - URL Routing with NavigationStack - Brandon Williams". The player shows a man with a mustache standing behind a podium, speaking. The background is dark with stage lights. To the right of the video frame, there is a sidebar with various sponsor logos: adidas, GoodNote, shape, mobile.dev, stream, wallbox, Globant, Judo, wallapop, and cademagic. Below the video frame, the text "[~]NSSPAIN SEPT 15-16 • LOGROÑO MMXXII" is visible. At the bottom of the player, there is a progress bar showing "00:25". The top of the player has standard browser controls (back, forward, search) and a URL bar showing "vimeo.com/manage/videos/751580644". On the left side of the player, there are four circular icons: a color wheel, a photo, a GIF, and a link symbol.

22 - URL Routing with NavigationStack - Brandon Williams

After a brief overview of how SwiftUI's new NavigationStack API works, we'll explore how to build a router that can transform nebulous URLs into state that drives deep-linking in your

Storage 9%

vimeo.com/nsspain/swiftui-navigation

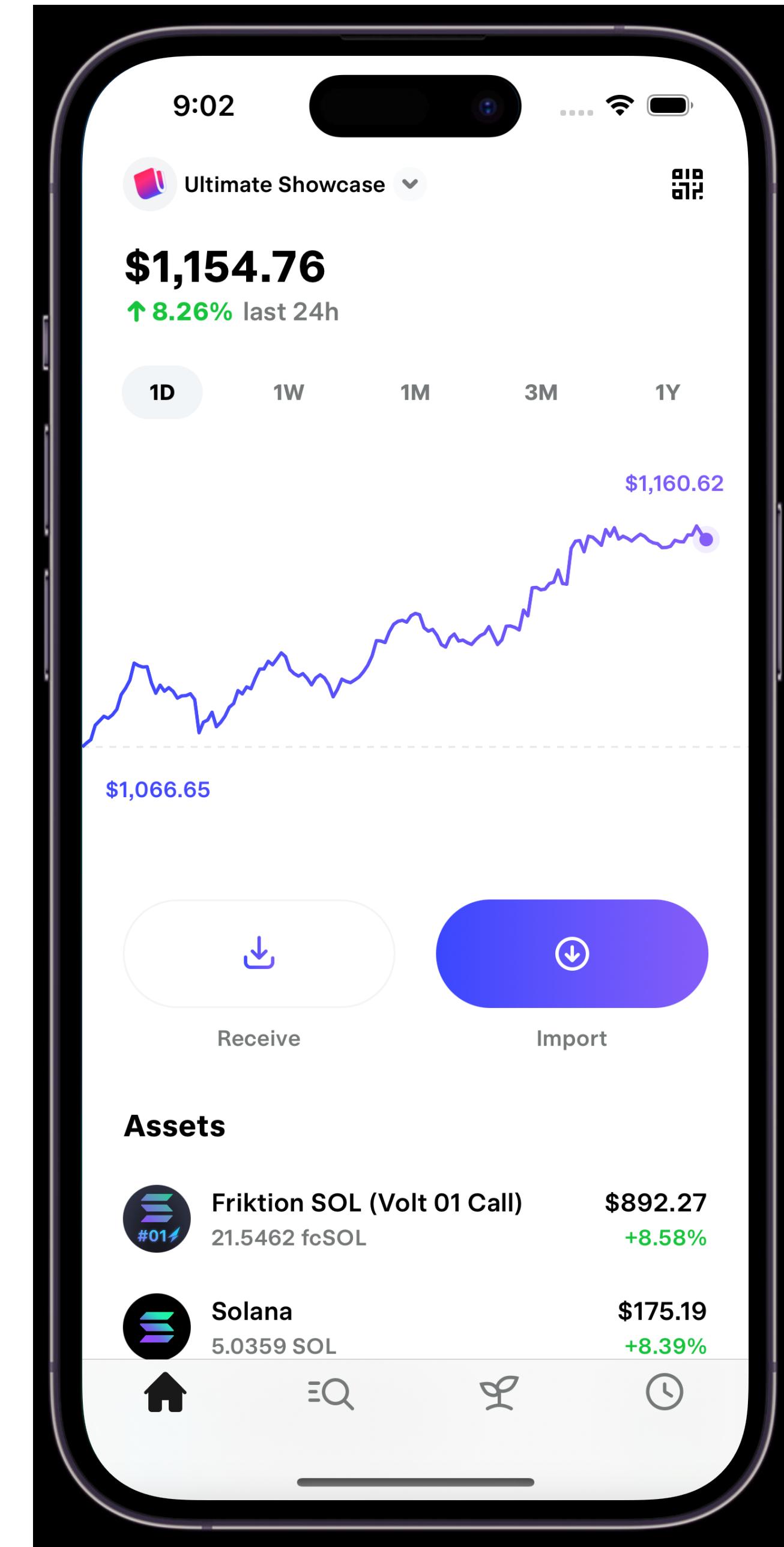
vimeo.com/nsspain/swiftui-navigation

The Composable Architecture

github.com/pointfreeco/swift-composable-architecture

Real life use case

ultimate.money



Which one to choose?

You should consider

You should consider

Navigation with UIKit if:

- Your project is UIKit
- You want to decouple the navigation logic from views
- You want to start using SwiftUI
- You can't switch your architecture

You should consider

Navigation with UIKit if:

- Your project is UIKit
- You want to decouple the navigation logic from views
- You want to start using SwiftUI
- You can't switch your architecture

Navigation with SwiftUI if:

- Your deployment target is iOS 14+
- You are starting a new project
- You want to use TCA

Merci

Luis Ascorbe @lascorbe