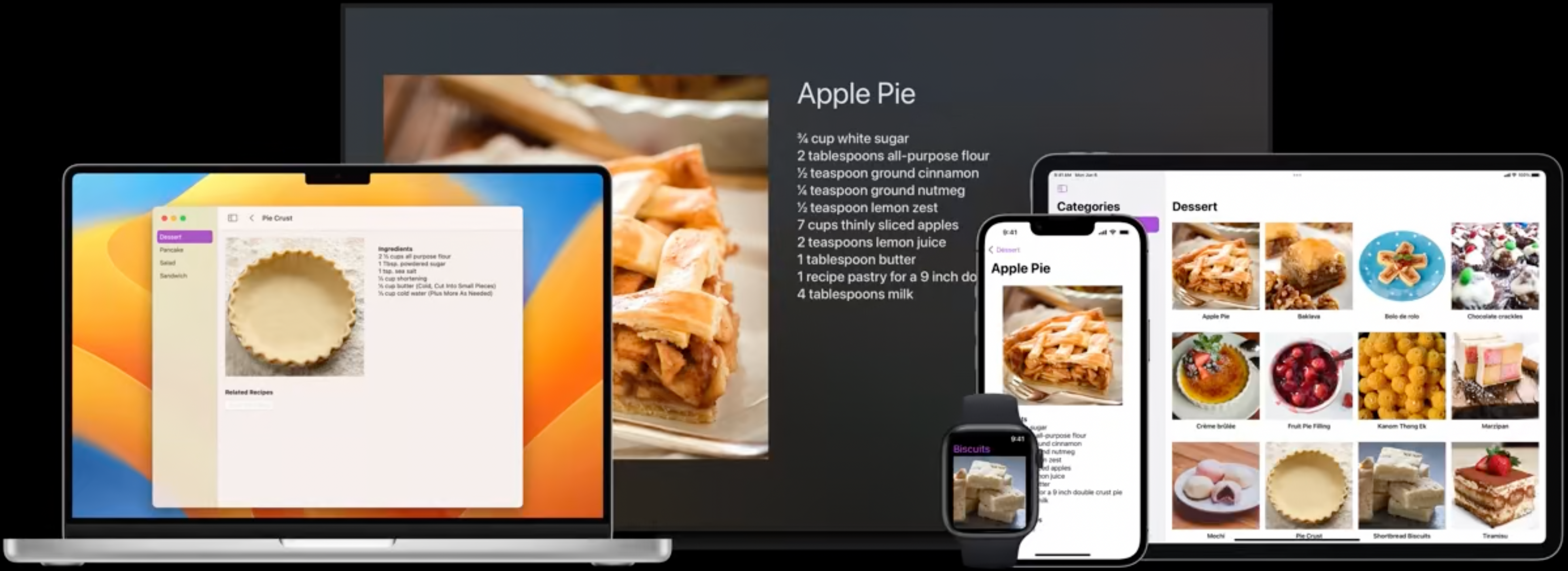
The Quimera: SwiftUI Navigation

FrenchKit 2022

@lascorbe



Ingredients
2 ½ cups all purpose flour
1 Tbsp. powdered sugar
1 tsp. sea salt
½ cup shortening
½ cup butter (Cold, Cut Into Small Pieces)
½ cup cold water (Plus More As Needed)

Apple Pie

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 do
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>

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

```swift
import SwiftUI

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

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

```swift
import SwiftUI

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

UIHostingController(rootView: FrenchView())
```

class UIHostingController<Content> where Content: View

```swift
import SwiftUI

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

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

class UIHostingC

```
class ViewModel: ObservableObject {

}
```

```
import SwiftUI

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

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

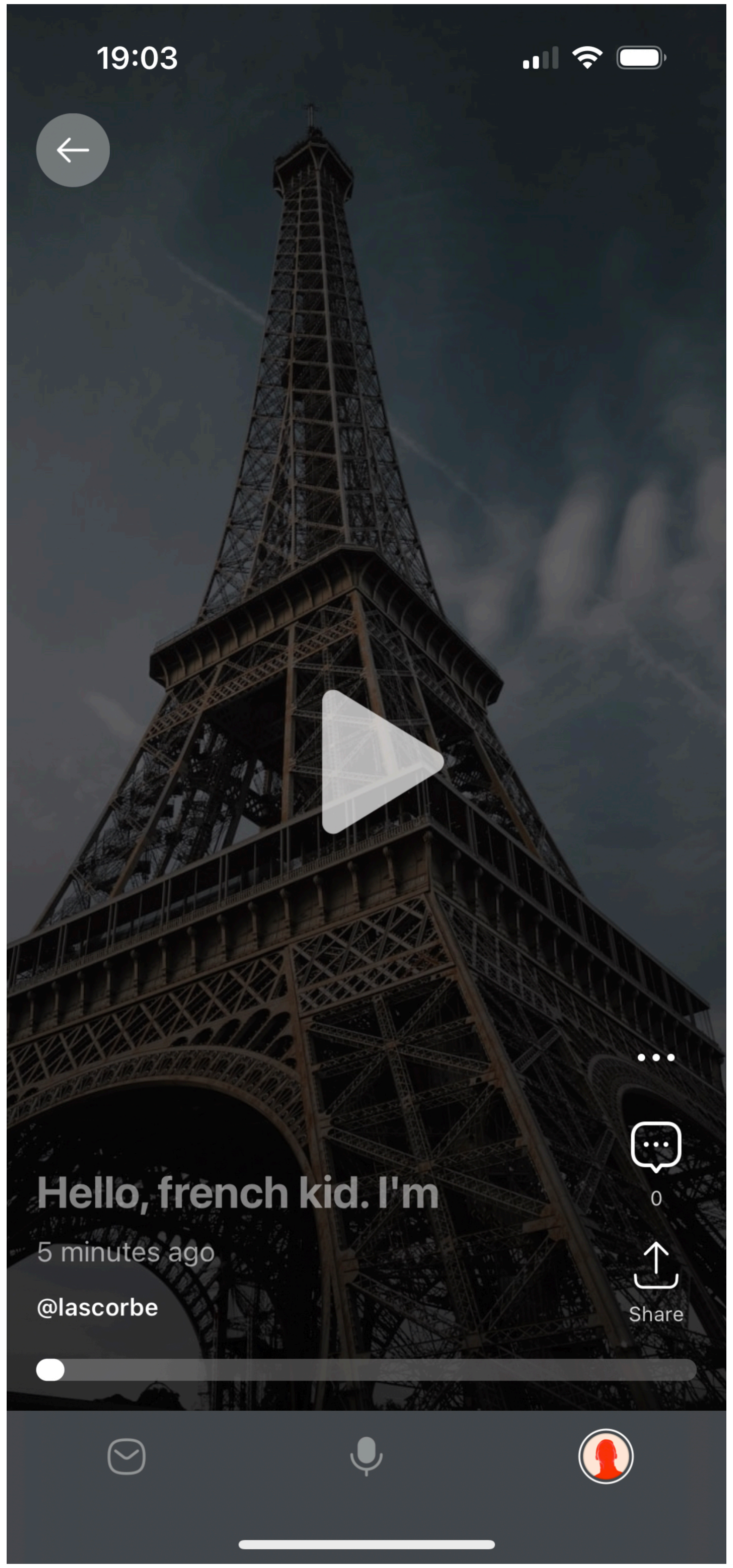
class UIHostingC

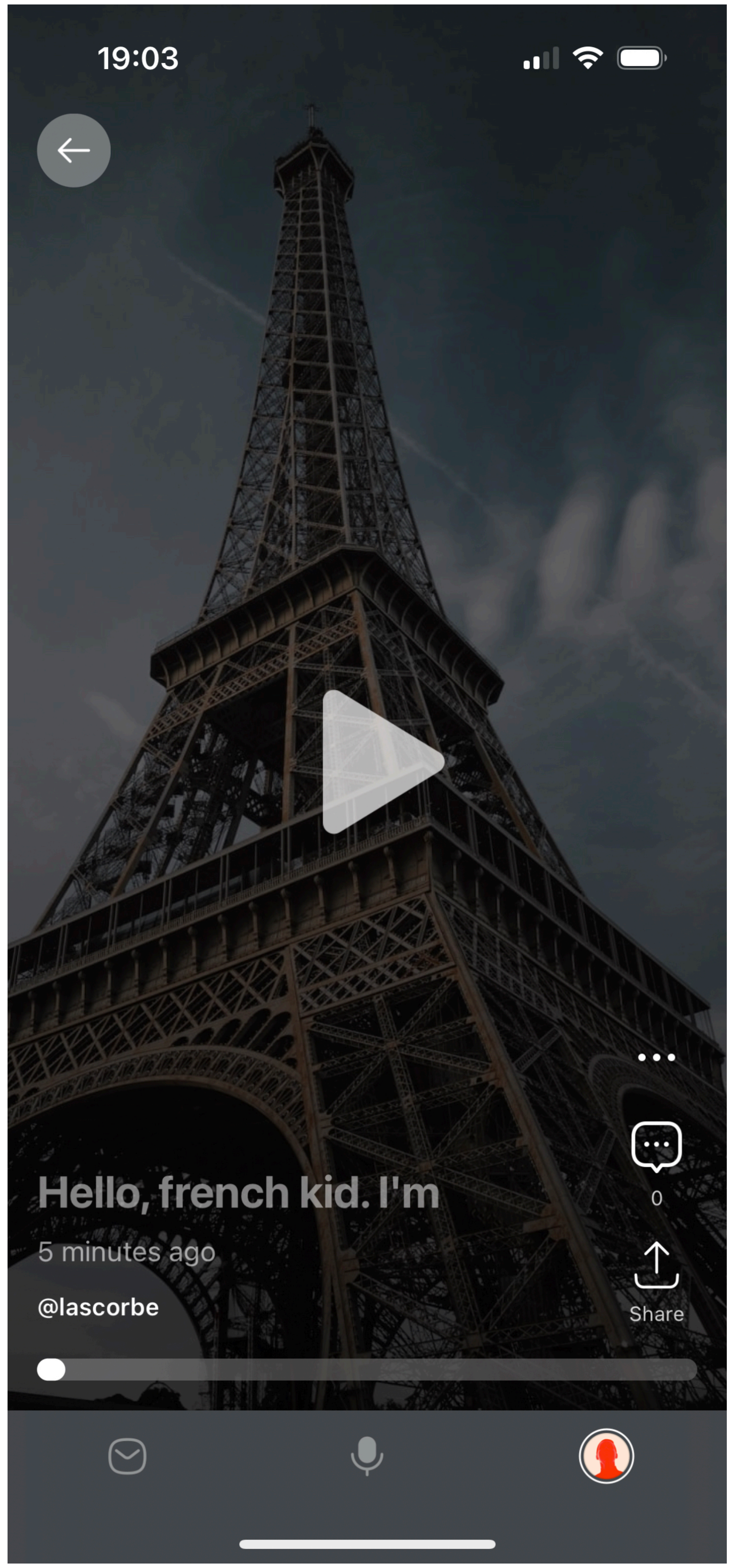
```
class ViewModel: ObservableObject {
    @Published var title: String
}
```

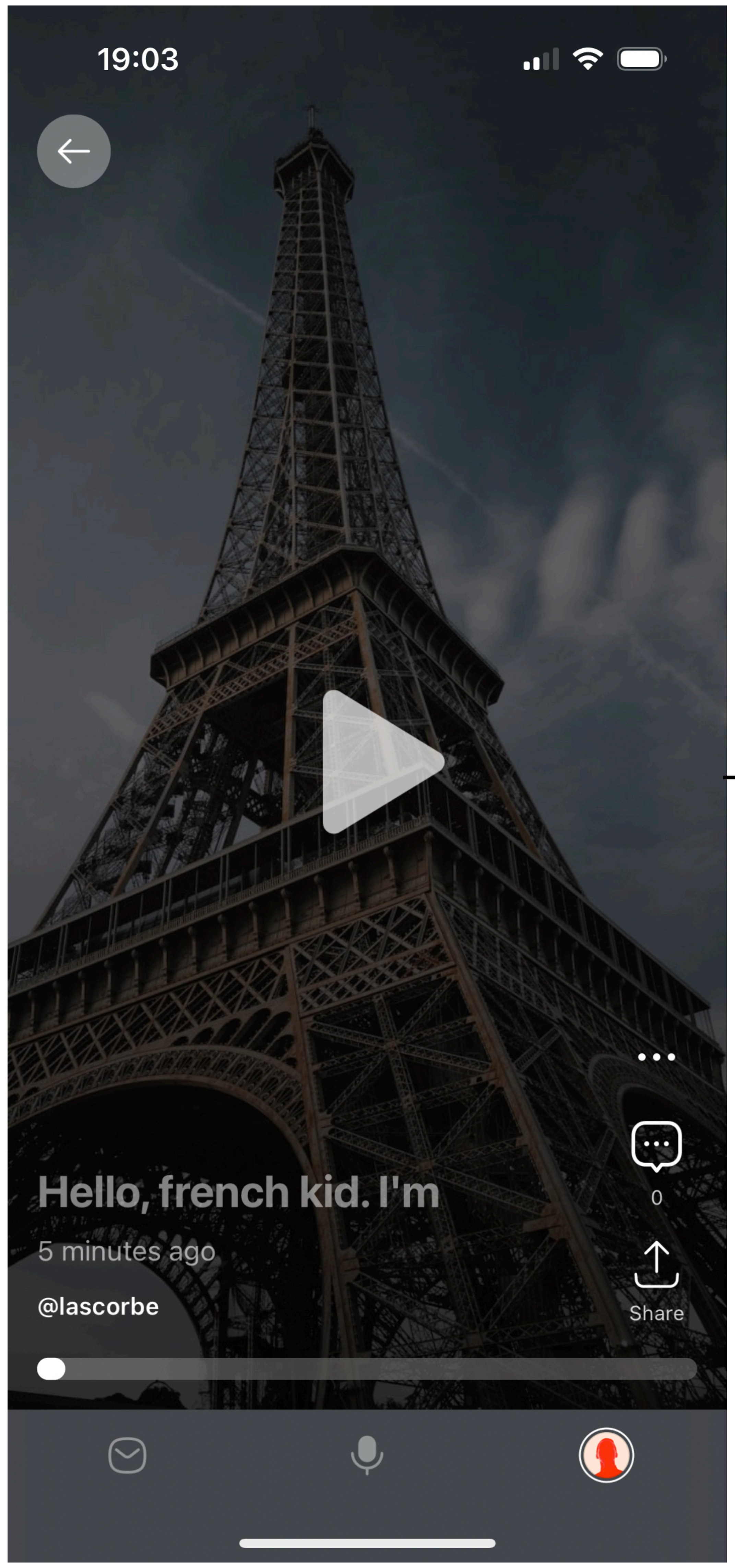
```
import SwiftUI

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

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

Real life use case

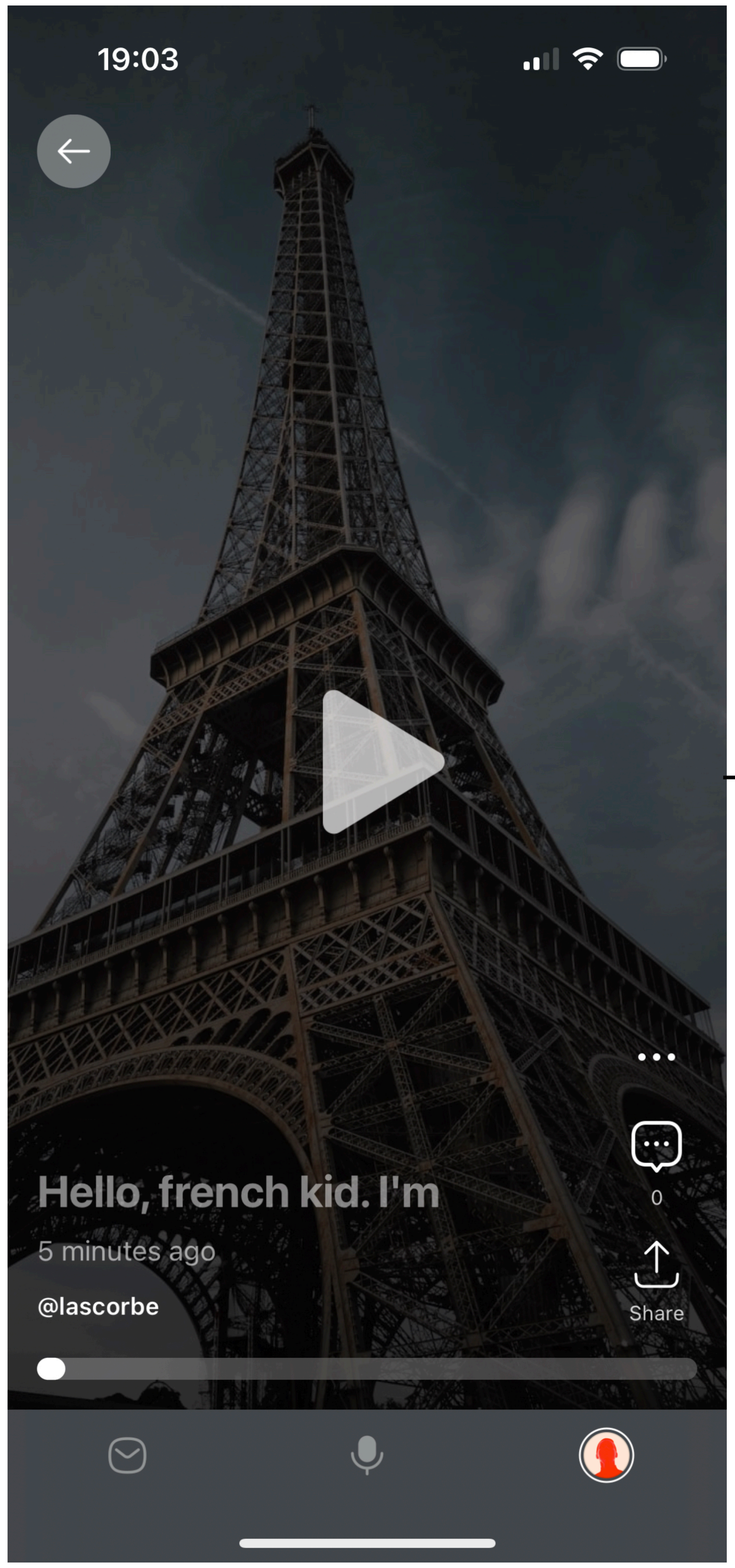
beams.fm




19:03
Hello, french kid. I'm
5 minutes ago
@lascorbe
0
Share

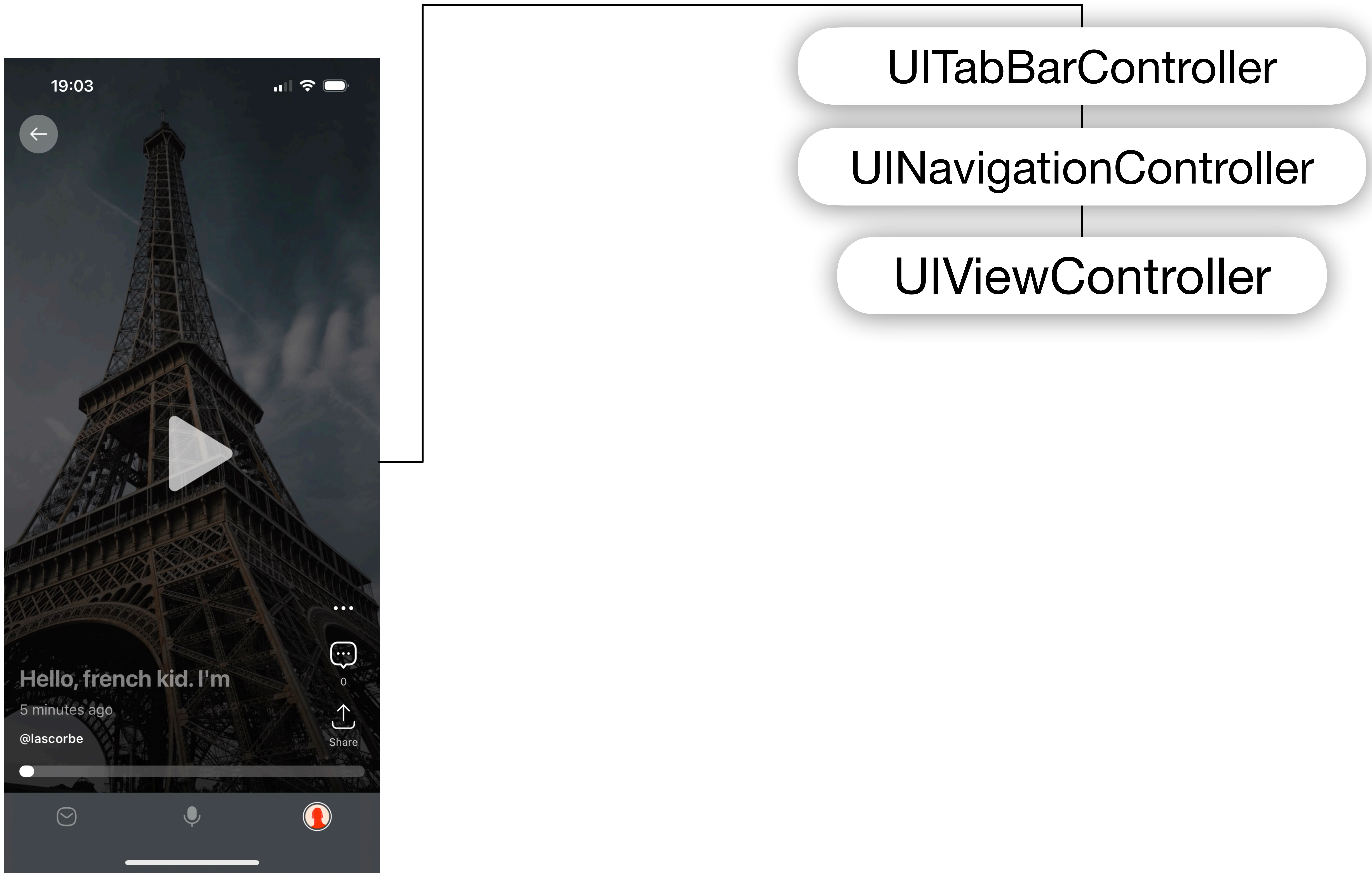


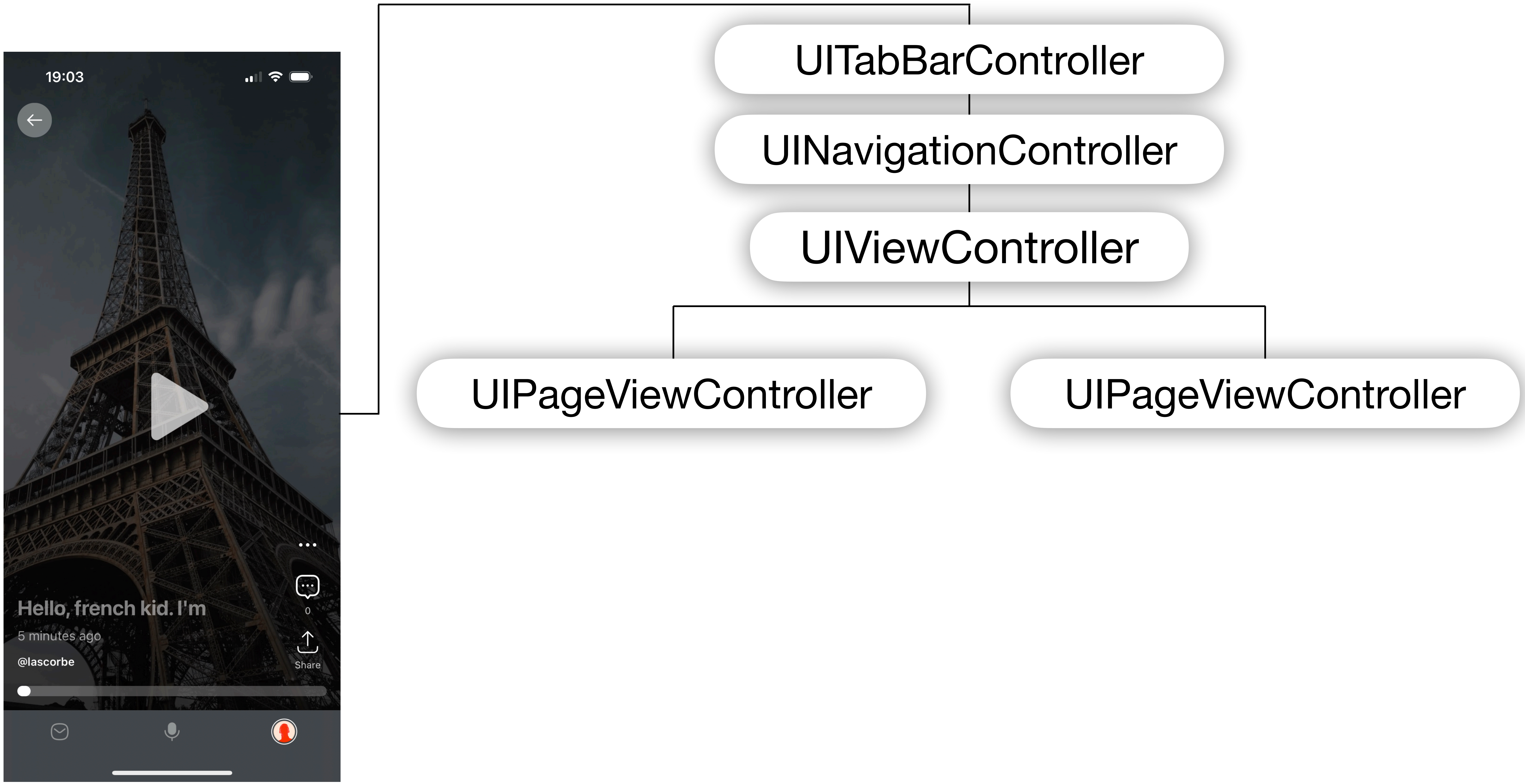
UITabBarController

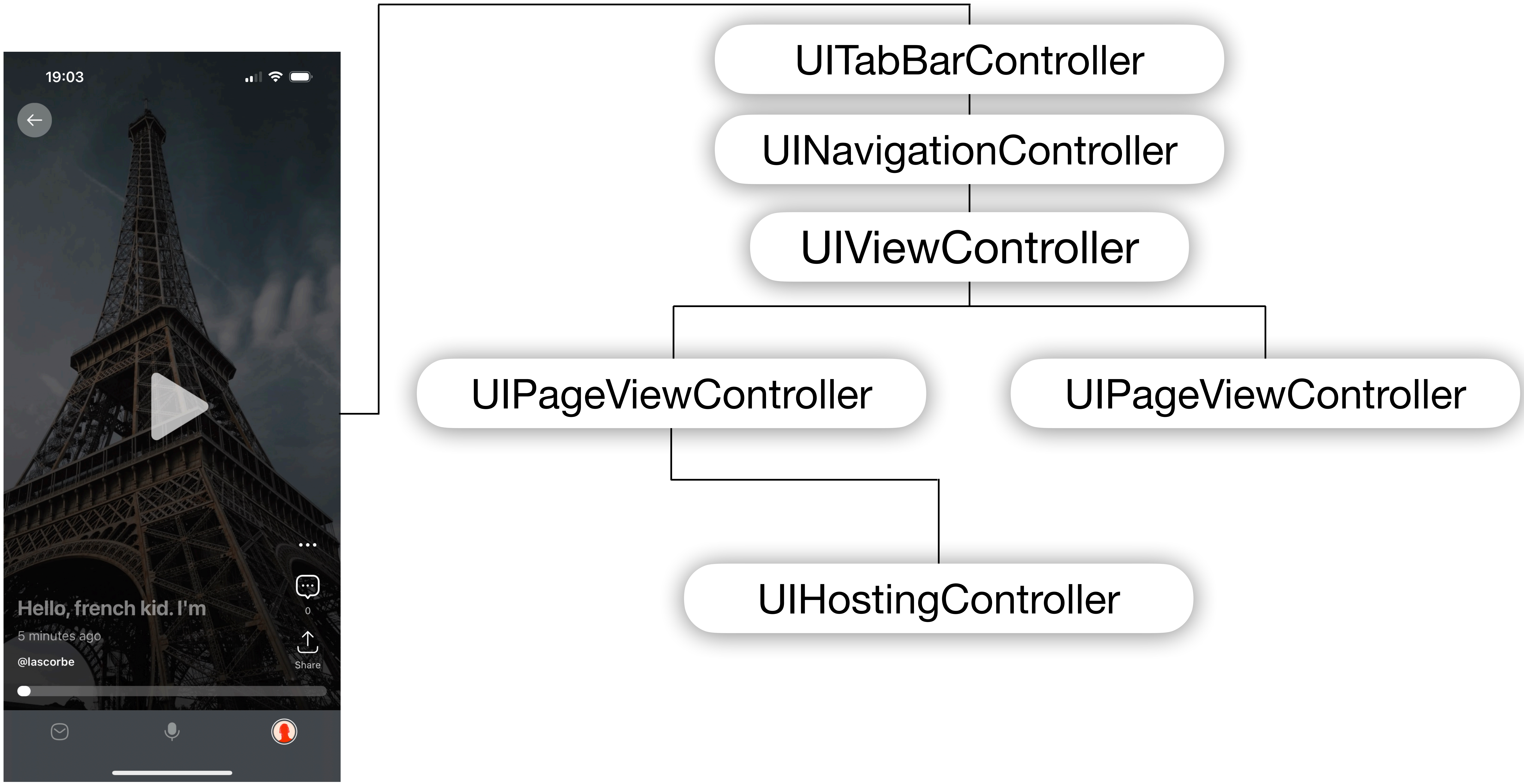


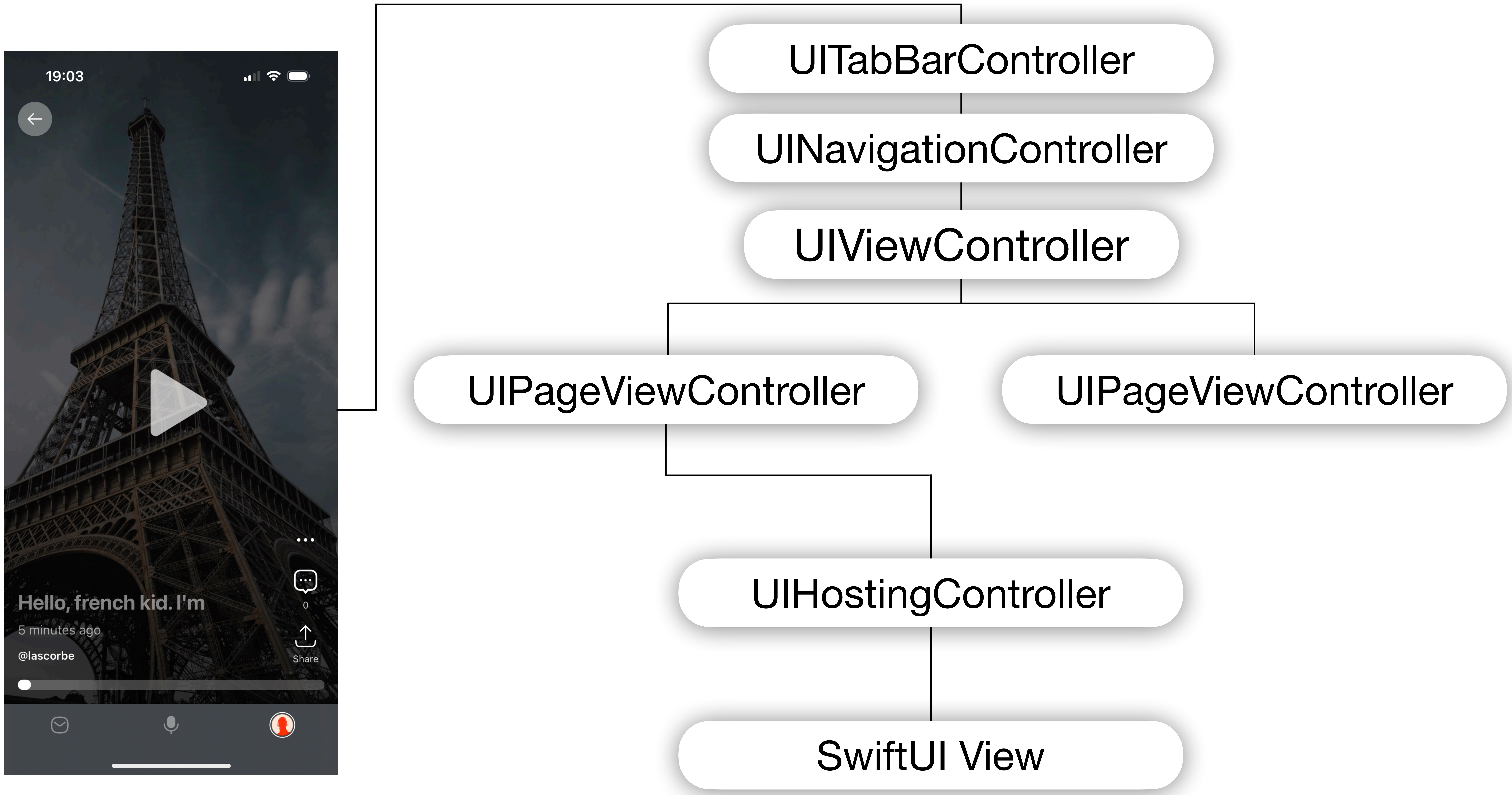
UITabBarController

UINavigationController


19:03
Hello, french kid. I'm
5 minutes ago
@lascorbe
0
Share
UITabBarController
UINavigationController
UIViewController


19:03
Hello, french kid. I'm
5 minutes ago
@lascorbe
0
Share
UITabBarController
UINavigationController
UIViewController
UIPageViewController
UIPageViewController


19:03
Hello, french kid. I'm
5 minutes ago
@lascorbe
0
Share
UITabBarController
UINavigationController
UIViewController
UIPageViewController
UIPageViewController
UIHostingController


19:03
Hello, french kid. I'm
5 minutes ago
@lascorbe
0
Share
UITabBarController
UINavigationController
UIViewController
UIPageViewController
UIPageViewController
UIHostingController
SwiftUI View

Why?

Why?

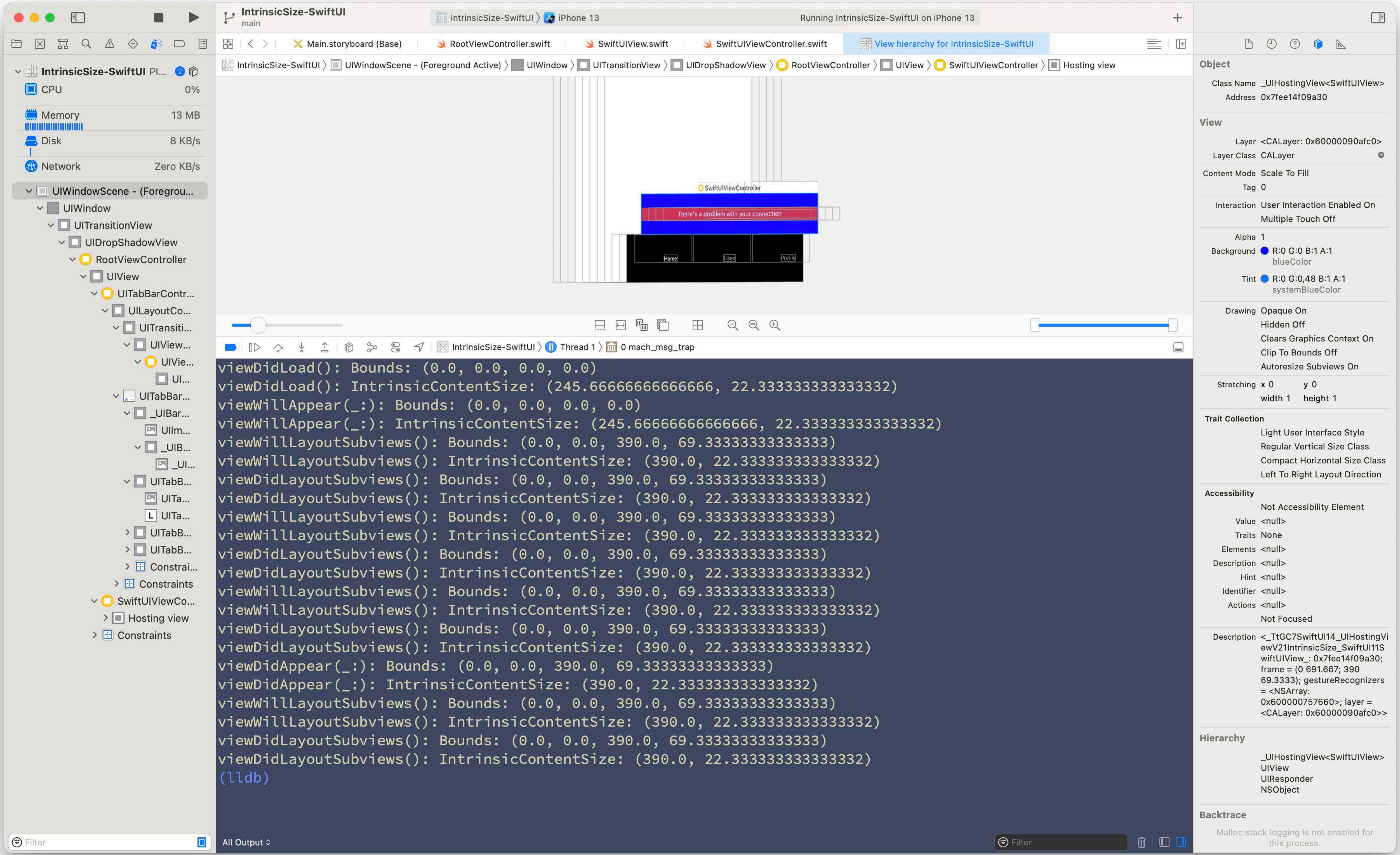
- All our navigation was working with UIKit

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



```
viewDidLoad(): Bounds: (0.0, 0.0, 0.0, 0.0)
viewDidLoad(): IntrinsicContentSize: (245.66666666666666, 22.333333333333332)
viewWillAppear(_:): Bounds: (0.0, 0.0, 0.0, 0.0)
viewWillAppear(_:): IntrinsicContentSize: (245.66666666666666, 22.333333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewDidAppear(_:): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewDidAppear(_:): IntrinsicContentSize: (390.0, 22.333333333333332)
viewWillLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewWillLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
viewDidLayoutSubviews(): Bounds: (0.0, 0.0, 390.0, 69.33333333333333)
viewDidLayoutSubviews(): IntrinsicContentSize: (390.0, 22.333333333333332)
(lldb)
```

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
Autoresize Subviews 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_SwiftUI11SwiftUIView_: 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.

```
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

.navigationDestination

. popover

NavigationLink

NavigationSplitView

NavigationView

NavigationStack

.sheet

.alert

.confirmationDialog

. fullScreenCover

Navigation with SwiftUI

.navigationDestination

. popover

NavigationLink

NavigationSplitView

NavigationView

NavigationStack

.sheet

.alert

.confirmationDialog

. fullScreenCover

Navigation with SwiftUI

.navigationDestination

. popover

NavigationLink

NavigationSplitView

NavigationView

NavigationStack

.sheet

.alert

.confirmationDialog

. fullScreenCover

Navigation with SwiftUI

.navigationDestination

. popover

NavigationLink

NavigationSplitView

NavigationView

DEPRECATED

NavigationStack

.sheet

.alert

.confirmationDialog

. fullScreenCover

Navigation with SwiftUI

.navigationDestination

. popover

NavigationLink

NavigationSplitView

NavigationView

DEPRECATED

NavigationStack

.sheet

.alert

.confirmationDialog

. fullScreenCover

NavigationLink

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

NavigationLink

```
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")
    }
}
```

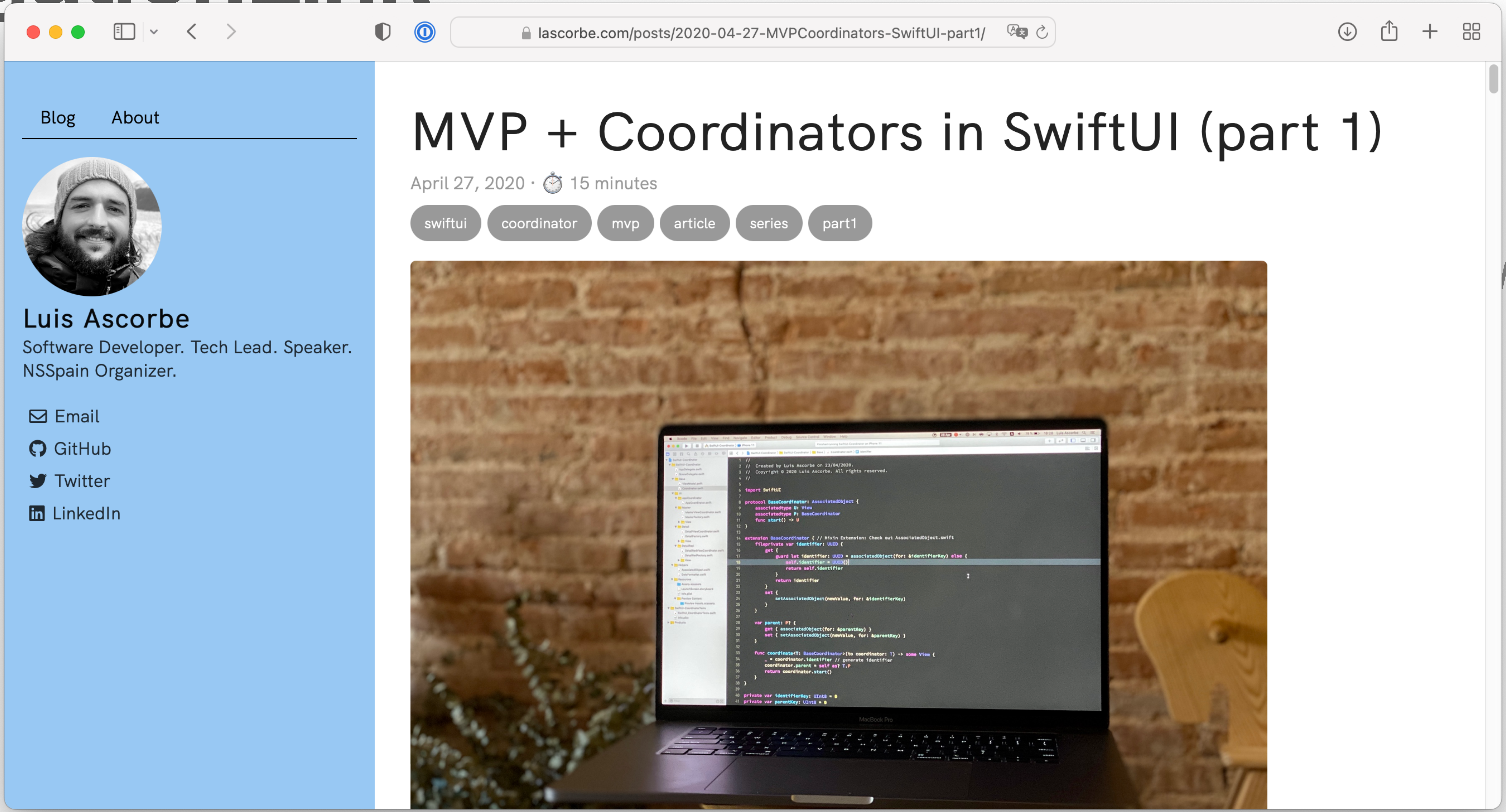
NavigationLink

```
struct 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
    va
                            w)

    }
}
```



lascorbe.com/posts/2020-04-27-MVPCoordinators-SwiftUI-part1/

Blog About

Luis Ascorbe

Software Developer. Tech Lead. Speaker. NSSpain Organizer.

Email
GitHub
Twitter
LinkedIn

MVP + Coordinators in SwiftUI (part 1)

April 27, 2020 · ⏱ 15 minutes

swiftui coordinator mvp article series part1

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()
        }
    }
}
```

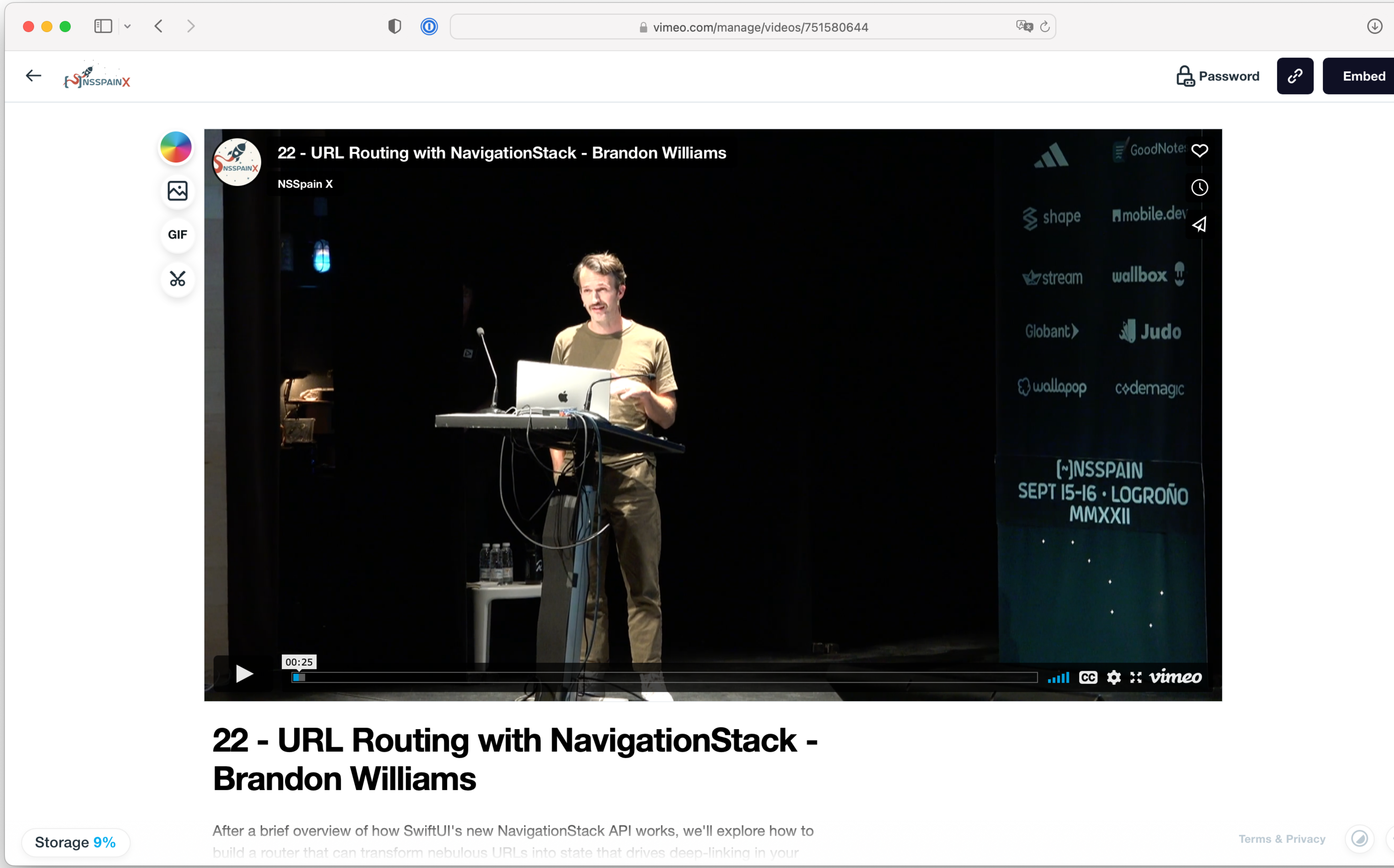
NavigationStack

```
NavigationStack(path: $viewMo
    List {
        NavigationLink(value:
            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



vimeo.com/manage/videos/751580644

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

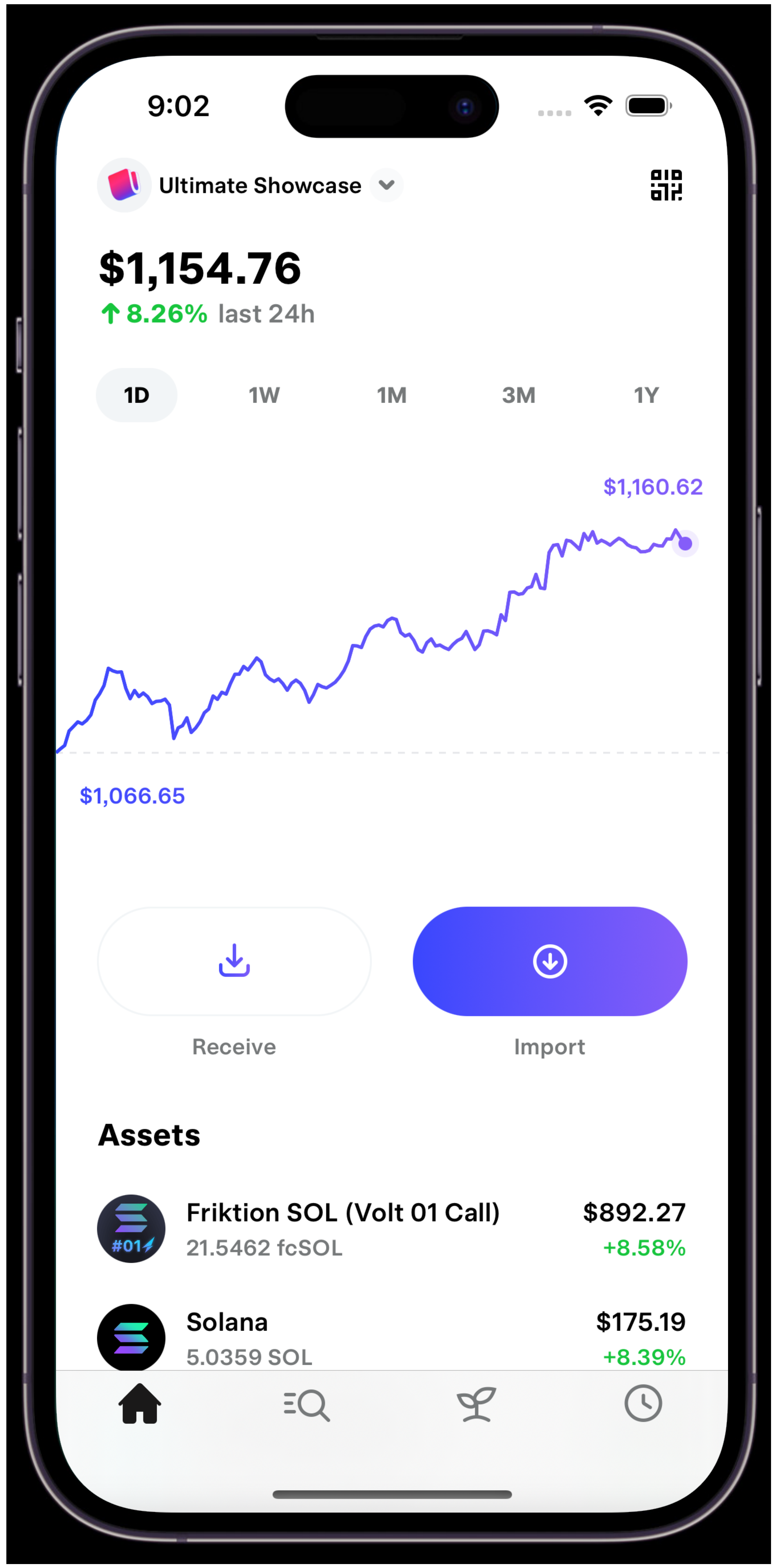
vimeo.com/nsspain/swiftui-navigation

The Composable Architecture

github.com/pointfreeco/swift-composable-architecture

Real life use case

ultimate.money


9:02

Ultimate Showcase

$1,154.76

↑ 8.26% last 24h

Assets

Friktion SOL (Volt 01 Call)	$892.27
21.5462 fcSOL	+8.58%
Solana	$175.19
5.0359 SOL	+8.39%

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