

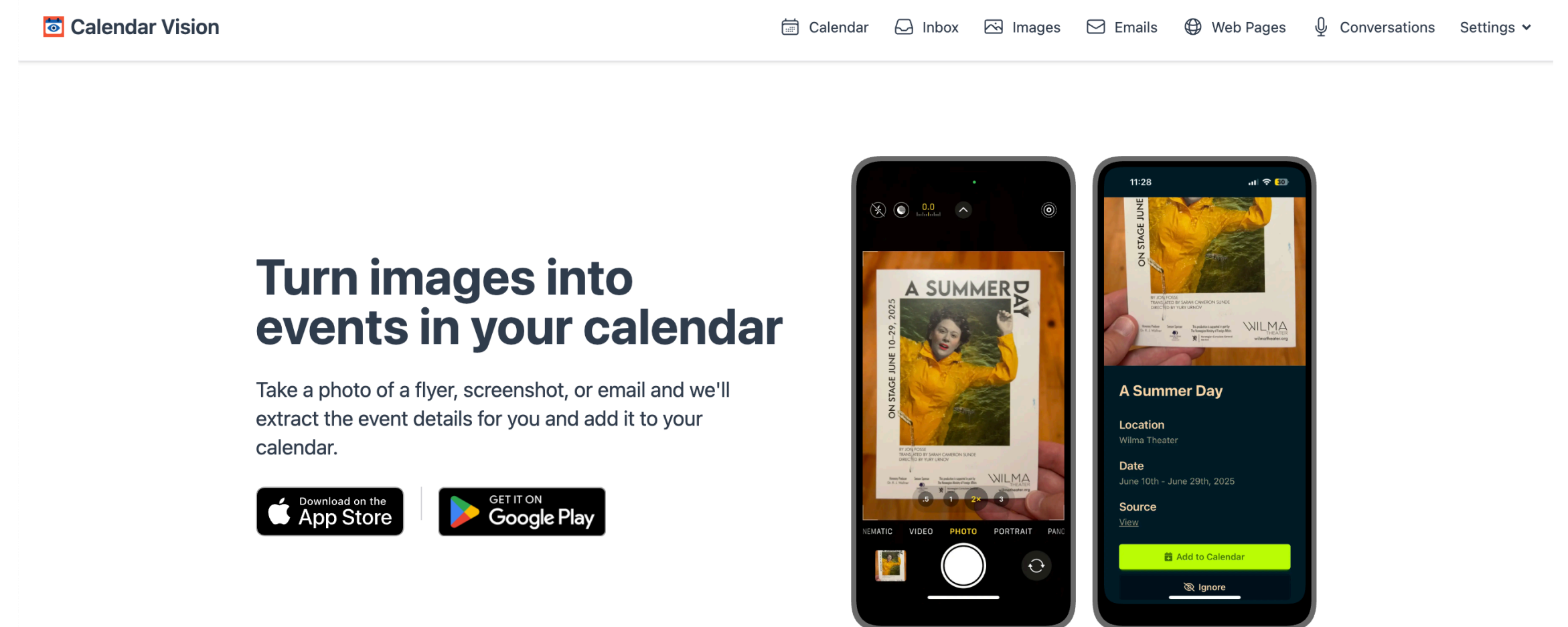
Introduction to Hotwire Native

Build iOS and Android apps with Ruby on Rails

Mike Dalton

Who am I?

- Based in Philadelphia
- Attended my first NYC.rb in October 2024
- Lead Engineer at Triumph
- Full Stack Ruby on Rails developer
- Using Hotwire Native on my side project



<https://calendarvision.app/>

Hotwire

Hotwire

- Created by 37signals
- Default front end “framework” in Rails 7+

“Hotwire is an alternative approach to building modern web applications without using much JavaScript by sending HTML instead of JSON over the wire.”

<https://hotwired.dev/>



Hotwire

Turbo

- Requires JavaScript but...
- No need to write your own JavaScript
- Behavior triggered by server-side responses

Hotwire

Turbo Drive

- Turbolinks successor
- Works “out of the box”
- Performs all requests in JS to prevent page reload

Hotwire

Turbo Frames

- Only update parts of a page
- Links or form submissions

```
<body>
  <div id="navigation">Links targeting the entire page</div>

  <turbo-frame id="message_1">
    <h1>My message title</h1>
    <p>My message content</p>
    <a href="/messages/1/edit">Edit this message</a>
  </turbo-frame>

  <turbo-frame id="comments">
    <div id="comment_1">One comment</div>
    <div id="comment_2">Two comments</div>

    <form action="/messages/comments">...</form>
  </turbo-frame>
</body>
```

<https://turbo.hotwired.dev/handbook/frames>

Hotwire

Turbo Streams

- Modify any part of the page
- More control, more code, more complexity

```
<turbo-stream action="append" target="messages">
  <template>
    <div id="message_1">
      This div will be appended to the element with the DOM ID "messages".
    </div>
  </template>
</turbo-stream>

<turbo-stream action="prepend" target="messages">
  <template>
    <div id="message_1">
      This div will be prepended to the element with the DOM ID "messages".
    </div>
  </template>
</turbo-stream>

<turbo-stream action="replace" target="message_1">
  <template>
    <div id="message_1">
      This div will replace the existing element with the DOM ID "message_1".
    </div>
  </template>
</turbo-stream>
```

<https://turbo.hotwired.dev/handbook/streams>

Hotwire

Stimulus

- Write custom JavaScript
- Like jQuery but more structured

Hotwire

Stimulus

Sprinkle your HTML with controller, target, and action attributes:

```
<!--HTML from anywhere-->
<div data-controller="hello">
  <input data-hello-target="name" type="text">

  <button data-action="click->hello#greet">
    Greet
  </button>

  <span data-hello-target="output">
  </span>
</div>
```

Write a compatible controller and watch Stimulus bring it to life:

```
// hello_controller.js
import { Controller } from "stimulus"

export default class extends Controller {
  static targets = [ "name", "output" ]

  greet() {
    this.outputTarget.textContent =
      `Hello, ${this.nameTarget.value}!`
  }
}
```

Mike

Greet

“Hello, Mike!”

Hotwire

Native

- More to come...

Mobile in 2025

So many choices

Mobile in 2025

Native

Android (Kotlin)

Developers

Essentials

Develop

More

Search

English

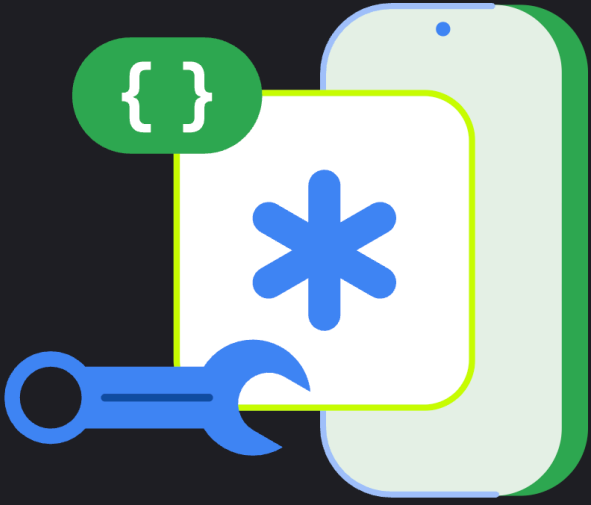
Android Studio

M

Develop for Android

The code samples, guides, and API reference you need—whether you're building for phones, watches, tablets, laptops, foldables, TVs, cars, or XR. Take advantage of these resources to develop your app faster and with higher quality.

I'm new to Android



iOS (Swift)

Developer

News

Discover

Design

Develop

Distribute

Support

Account

Search

Develop for iOS

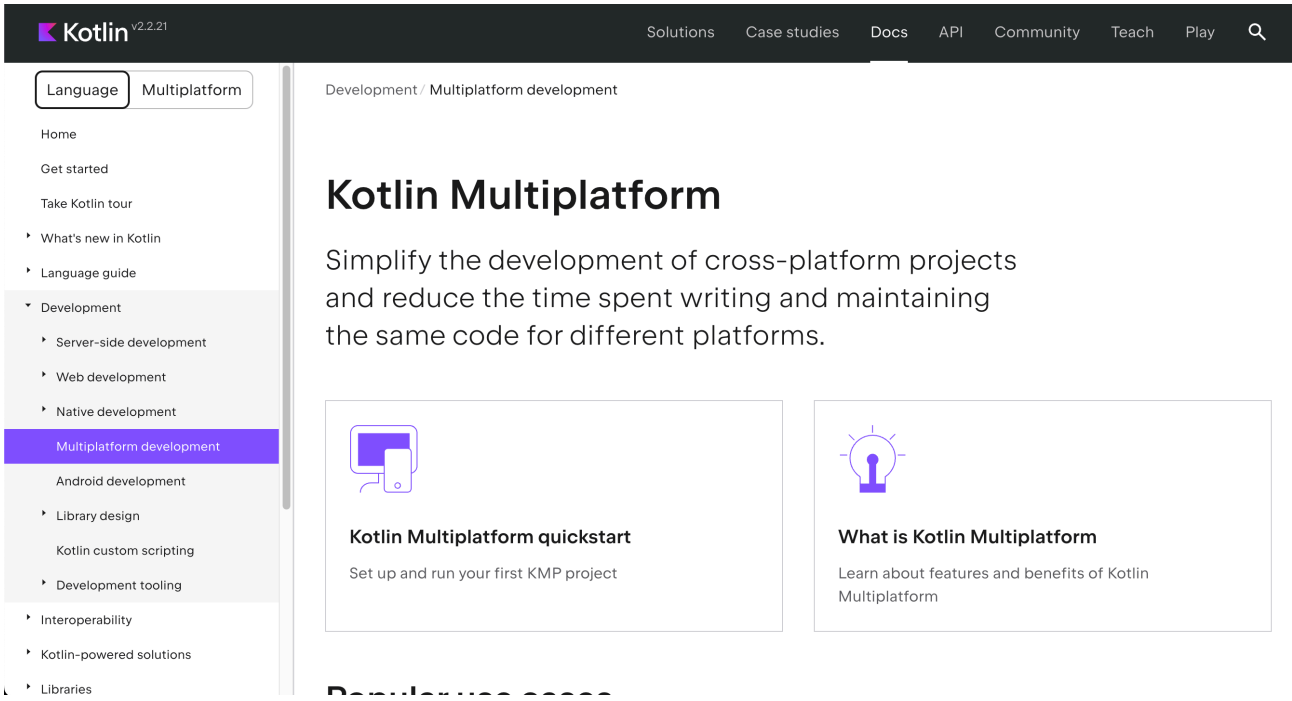
Create great apps and games for the world's most advanced mobile operating system. Take advantage of the new design, which elevates the content people care about most — and creates a unified design across Apple platforms. And with Apple Intelligence, you can bring personal intelligence into your apps to deliver new capabilities, all with great performance and built-in privacy.



Mobile in 2025

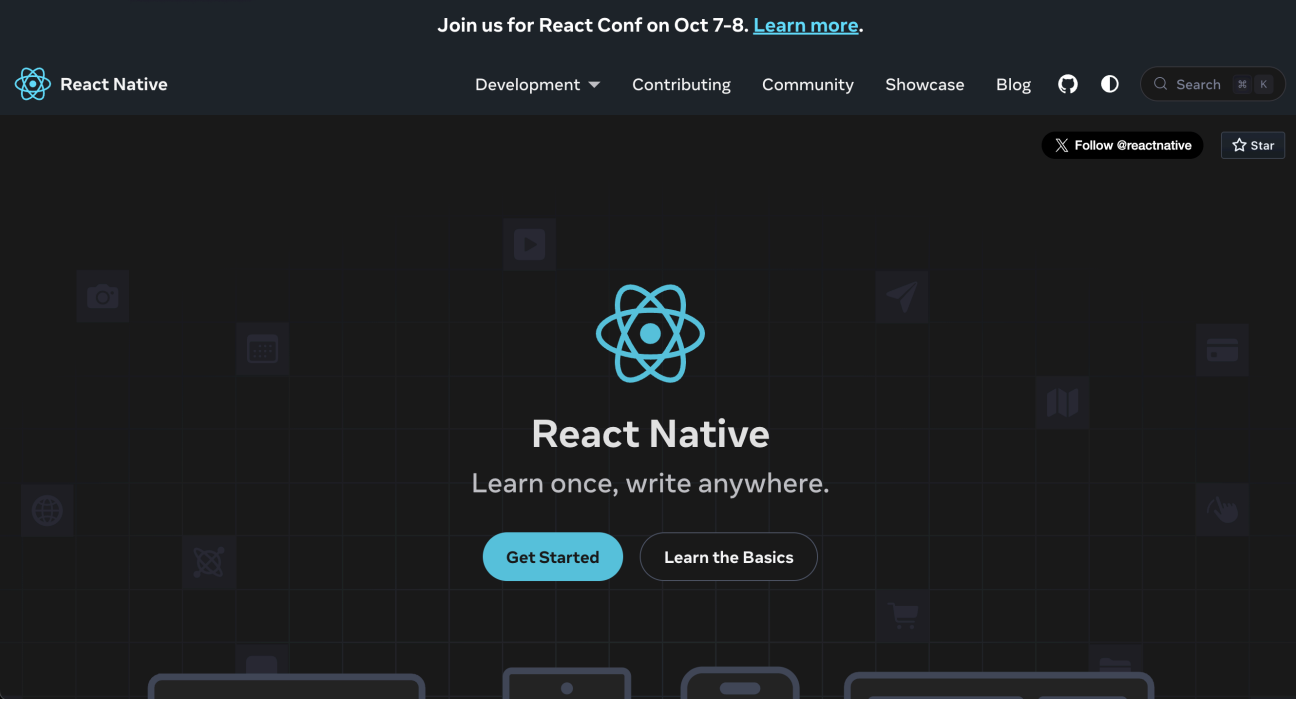
Crossplatform

Kotlin Multiplatform



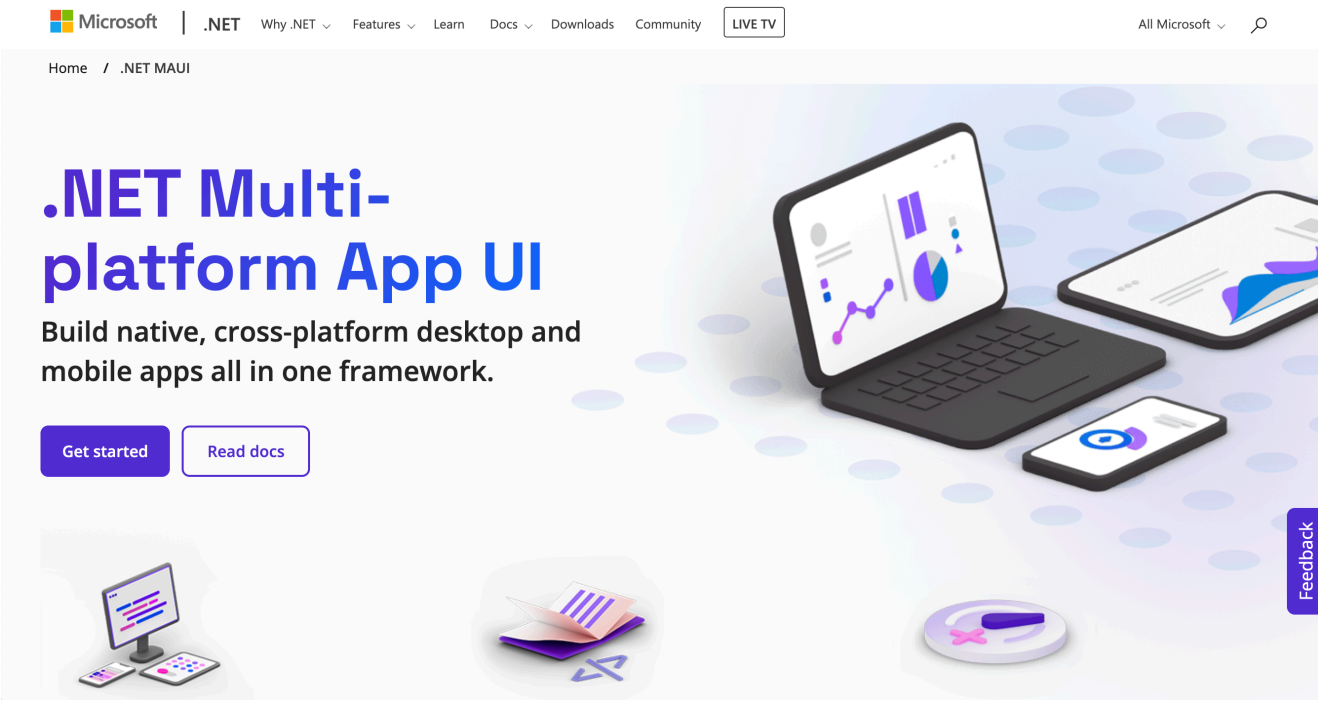
Kotlin

React Native



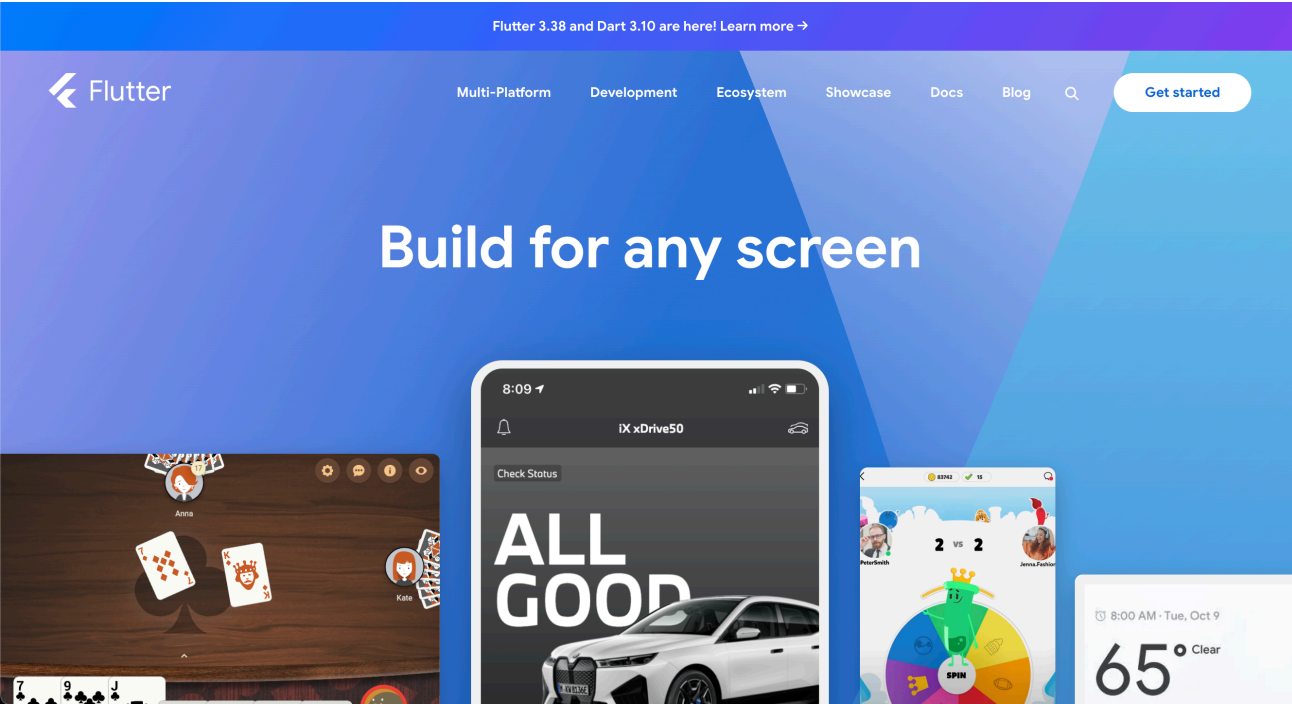
React

.NET MAUI



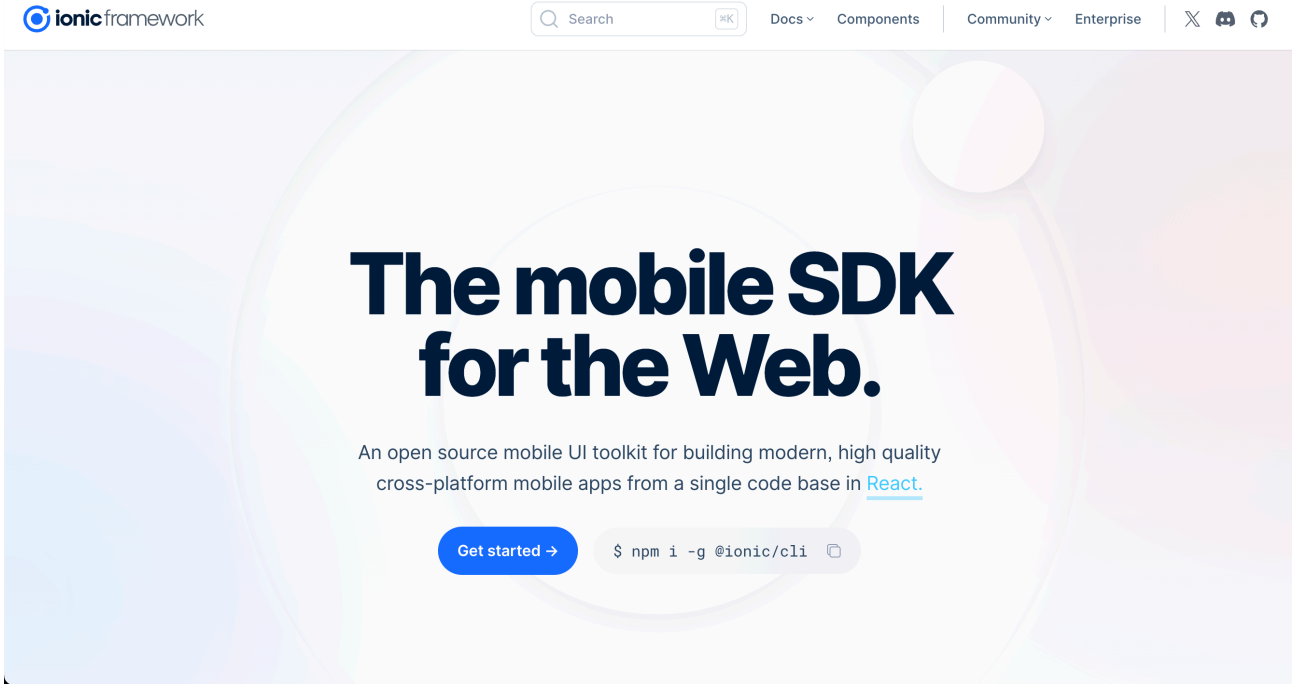
C#

Flutter



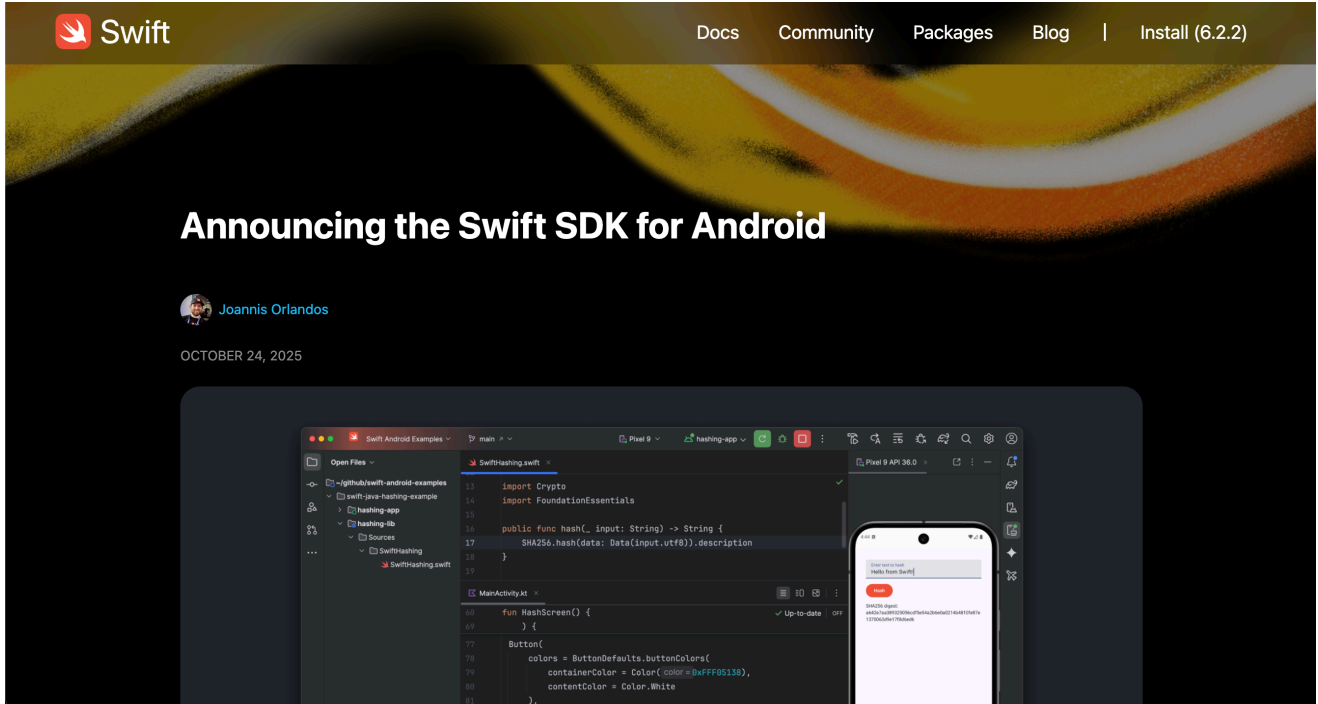
Dart

Ionic



Angular, React or Vue

Swift for Android



Swift

Hotwire Native

Hotwire Native

History

- Created by 37signals
- Turbo Native released in 2020
- Strada released in 2023
- Rebranded as Hotwire Native in 2024

Hotwire Native

Who should use it

- You have the need for a web app, iOS and Android app
- You want to build with Hotwire and Turbo
- You're comfortable being an early adopter

Hotwire Native

How does it work

- Embedded web browser
- Doesn't look like a browser
- iOS uses WKWebView
- Android uses WebView

Project Setup

Project Setup

- No generator (like `rails new`)
- Hotwire Native apps are Native apps

Project Setup

ios

1. Use the Xcode New Project Wizard
2. Add `hotwire-native-ios` package dependency
3. Replace `SceneDelegate.swift` with...

```
import HotwireNative
import UIKit

let rootURL = URL(string: "https://hotwire-native-demo.dev")!

class SceneDelegate: UIResponder, UIWindowSceneDelegate {
    var window: UIWindow?

    private let navigator = Navigator(configuration: .init(
        name: "main",
        startLocation: rootURL
    ))

    func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options connectionOptions: UIScene.ConnectionOptions) {
        window?.rootViewController = navigator.rootViewController
        navigator.start()
    }
}
```

SceneDelegate.swift

Project Setup

Android

1. Use the Android Studio New Project Wizard
2. Add `hotwire-native-android` dependencies to Gradle build files
3. Enable internet access in `AndroidManifest.xml`
4. Replace `MainActivity.kt` with...
5. Replace `activity_main.xml` with...

```
package com.example.myapplication

import android.os.Bundle
import android.view.View
import androidx.activity.enableEdgeToEdge
import dev.hotwire.navigation.activities.HotwireActivity
import dev.hotwire.navigation.navigator.NavigatorConfiguration
import dev.hotwire.navigation.util.applyDefaultImeWindowInsets

class MainActivity : HotwireActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        enableEdgeToEdge()
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        findViewById<View>(R.id.main_nav_host).applyDefaultImeWindowInsets()
    }

    override fun navigatorConfigurations() = listOf(
        NavigatorConfiguration(
            name = "main",
            startLocation = "https://hotwire-native-demo.dev",
            navigatorHostId = R.id.main_nav_host
        )
    )
}
```

MainActivity.kt

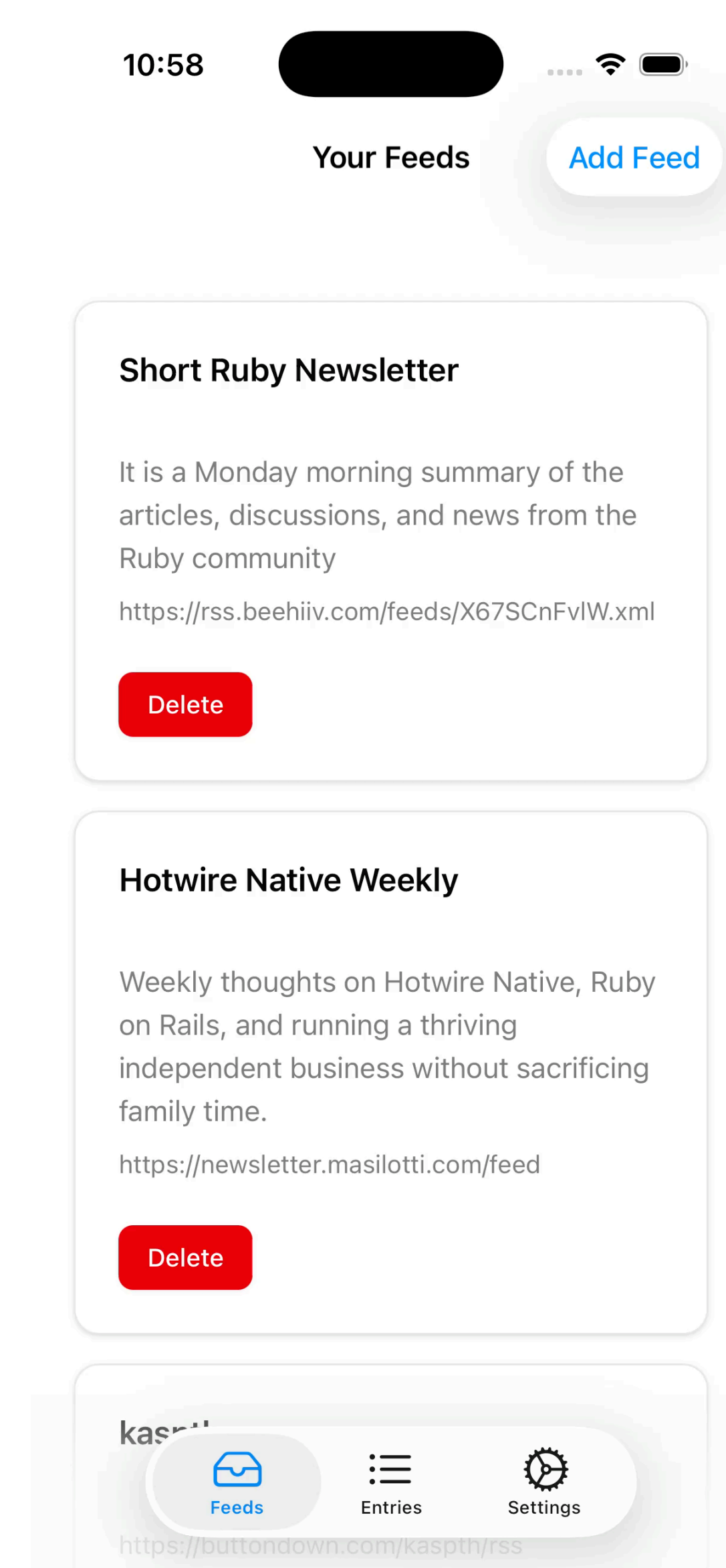

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main_nav_host"
    android:name="dev.hotwire.navigation.navigator.NavigatorHost"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:defaultNavHost="false" />
```

activity_main.xml

Screen Navigation

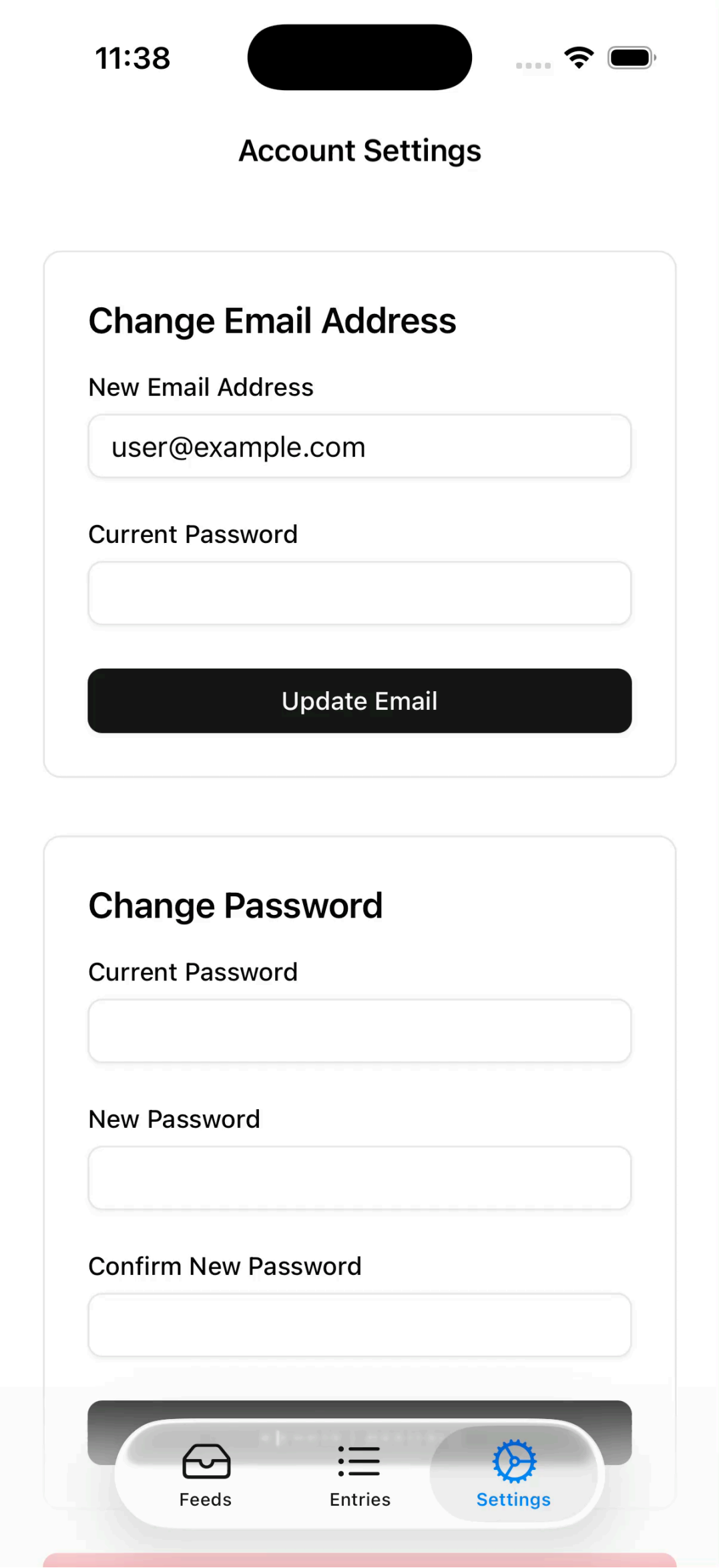
Screen Navigation

Push and Pop



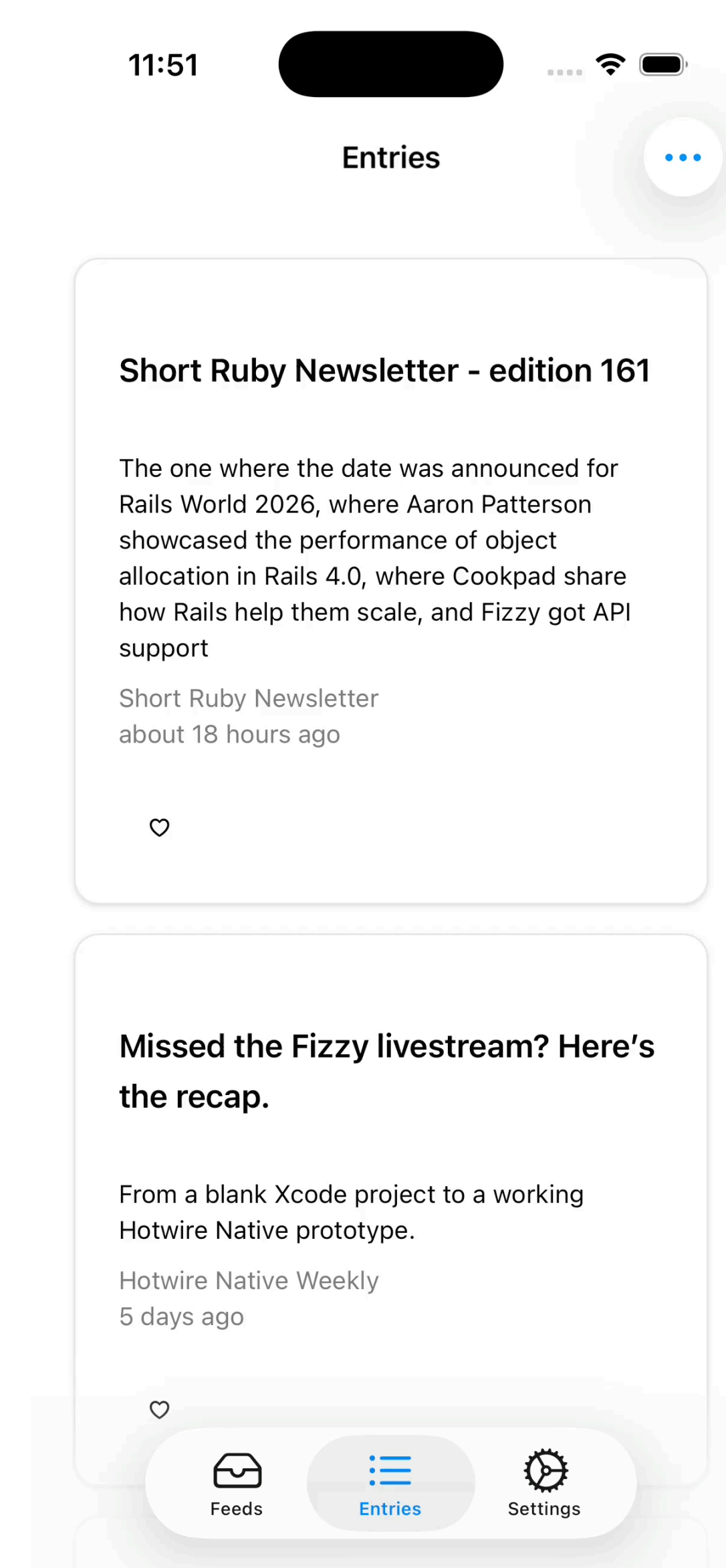
Screen Navigation

Replace



Screen Navigation

External Links

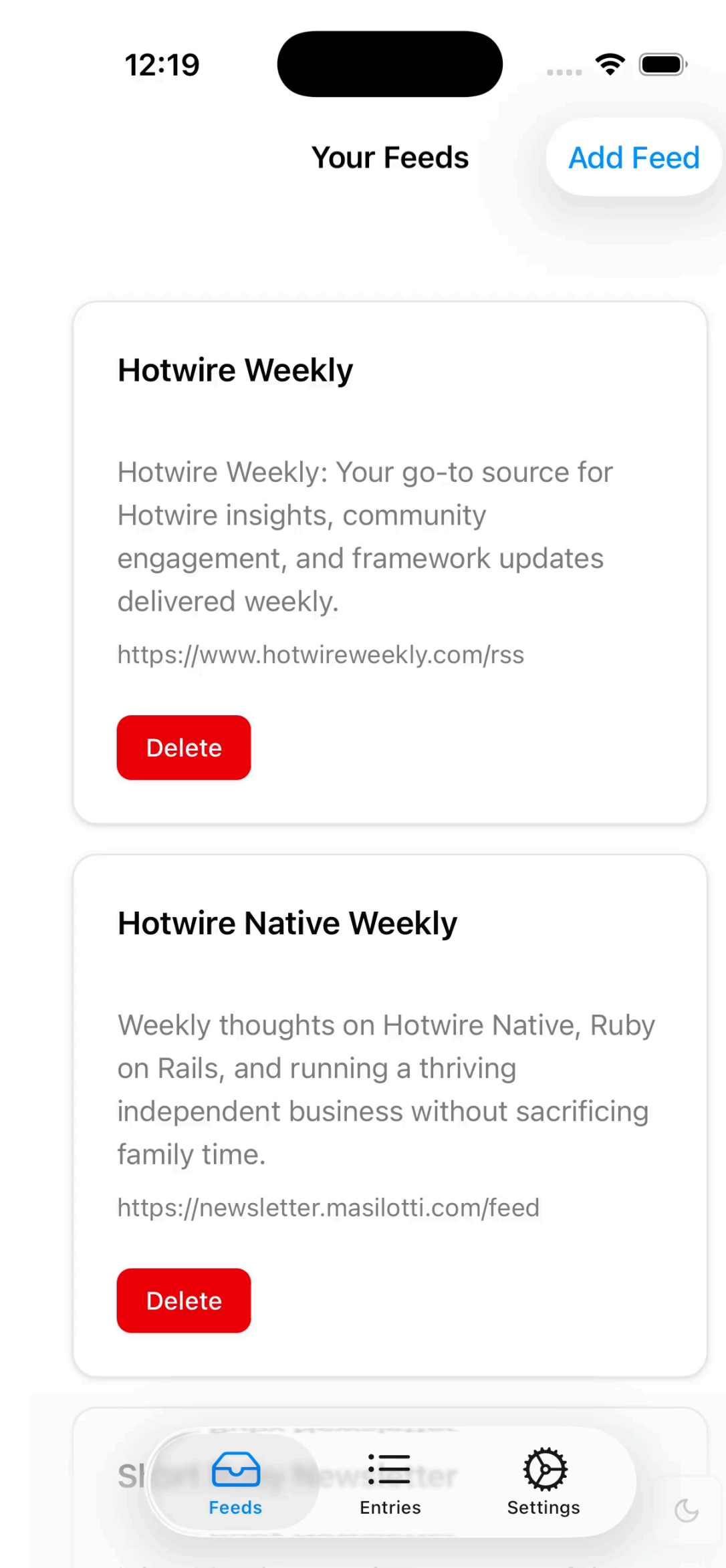


Path Configuration

Path Configuration

Basic Navigation

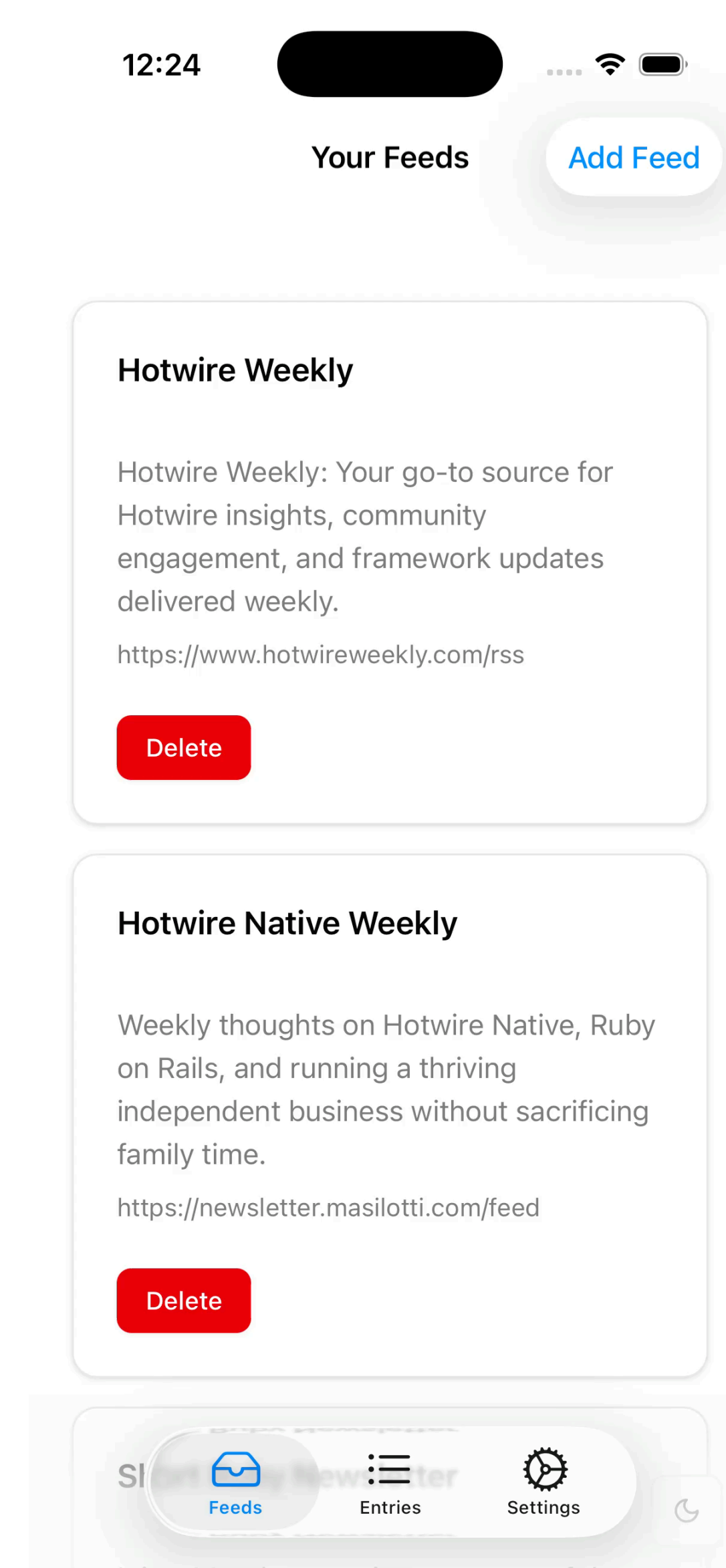
```
{
  "rules": [
    {
      "patterns": [
        ".*"
      ],
      "properties": {
        "context": "default",
        "pull_to_refresh_enabled": true
      }
    }
  ]
}
```



Path Configuration

Modal Navigation

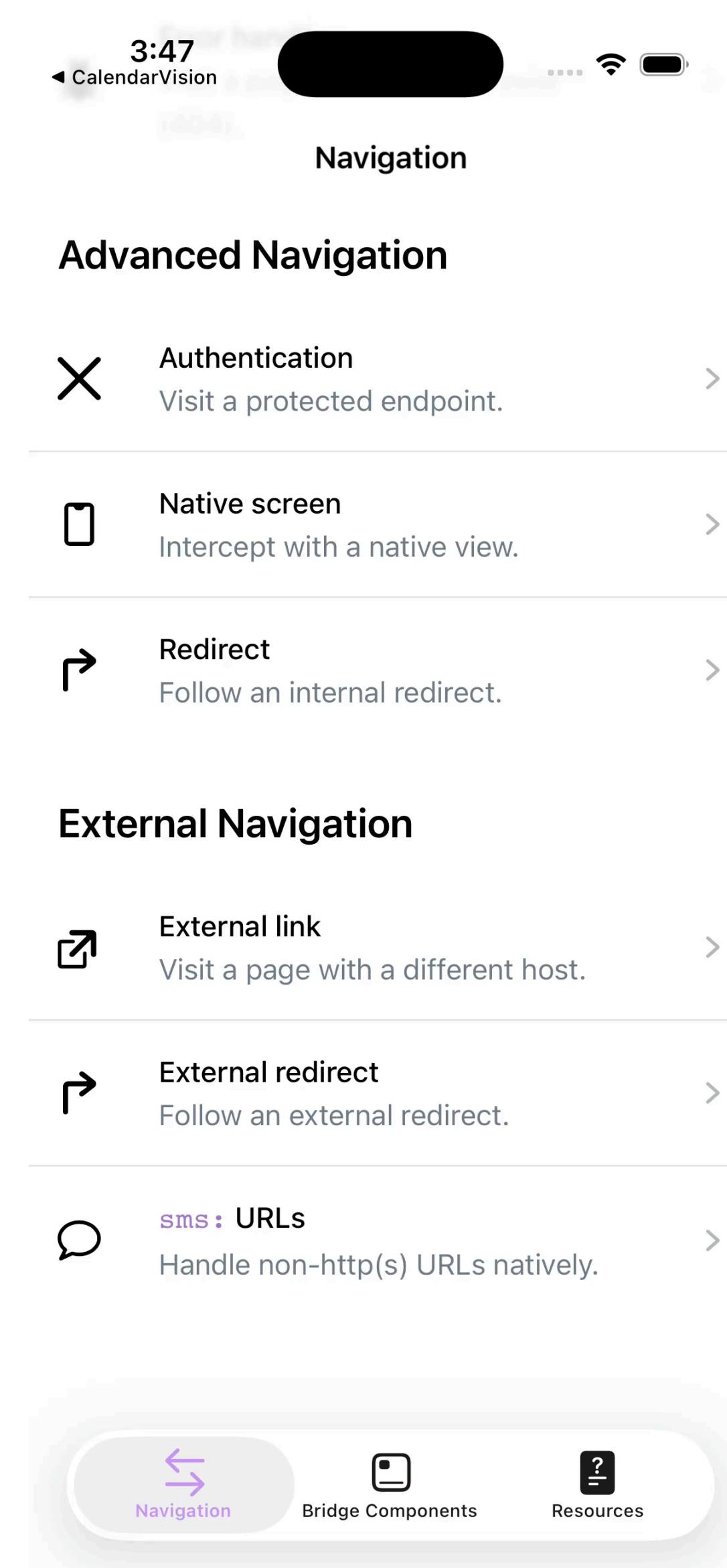
```
{
  "rules": [
    {
      "patterns": [
        "/new$",
        "/edit$"
      ],
      "properties": {
        "context": "modal",
        "pull_to_refresh_enabled": false
      }
    }
  ]
}
```



Path Configuration

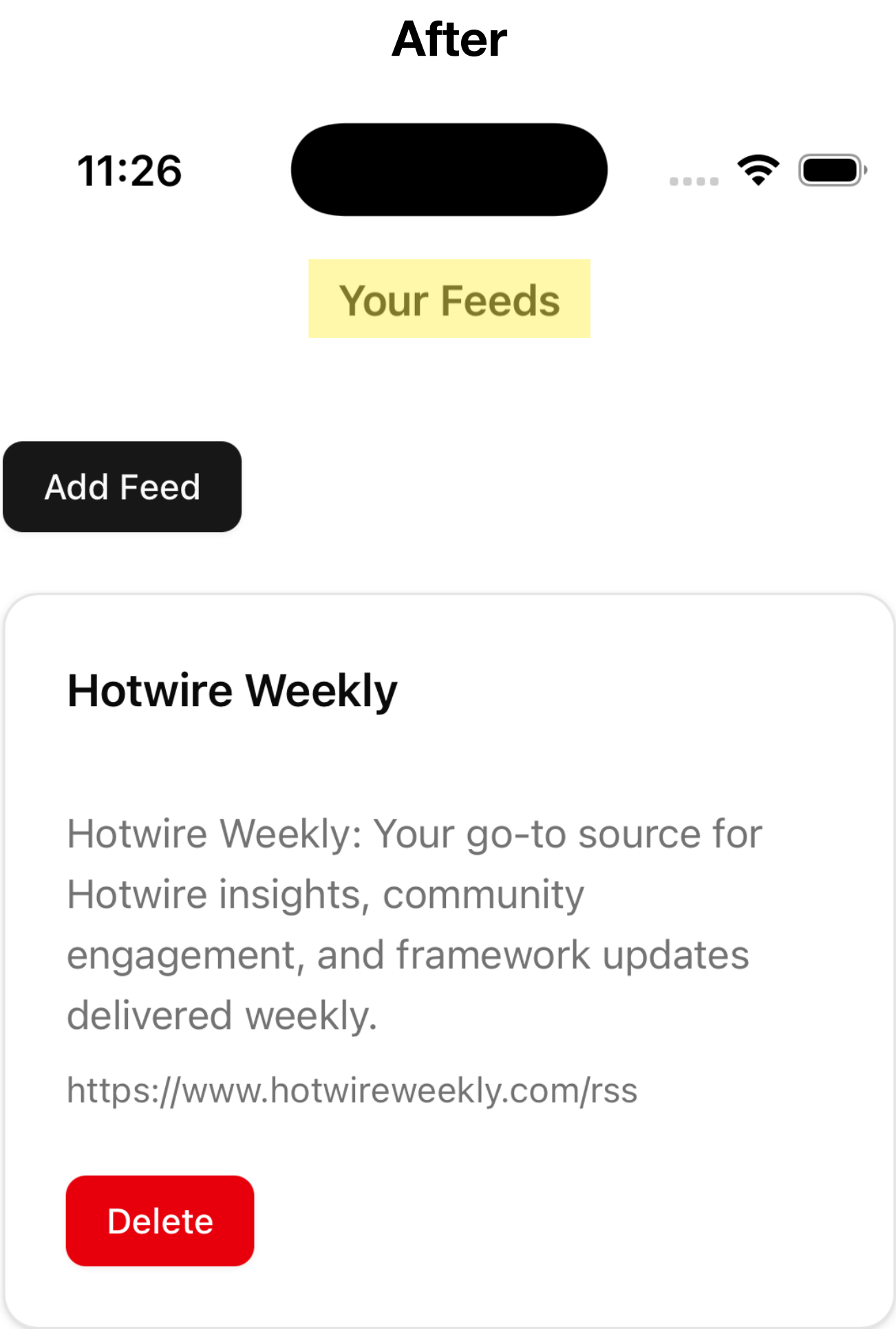
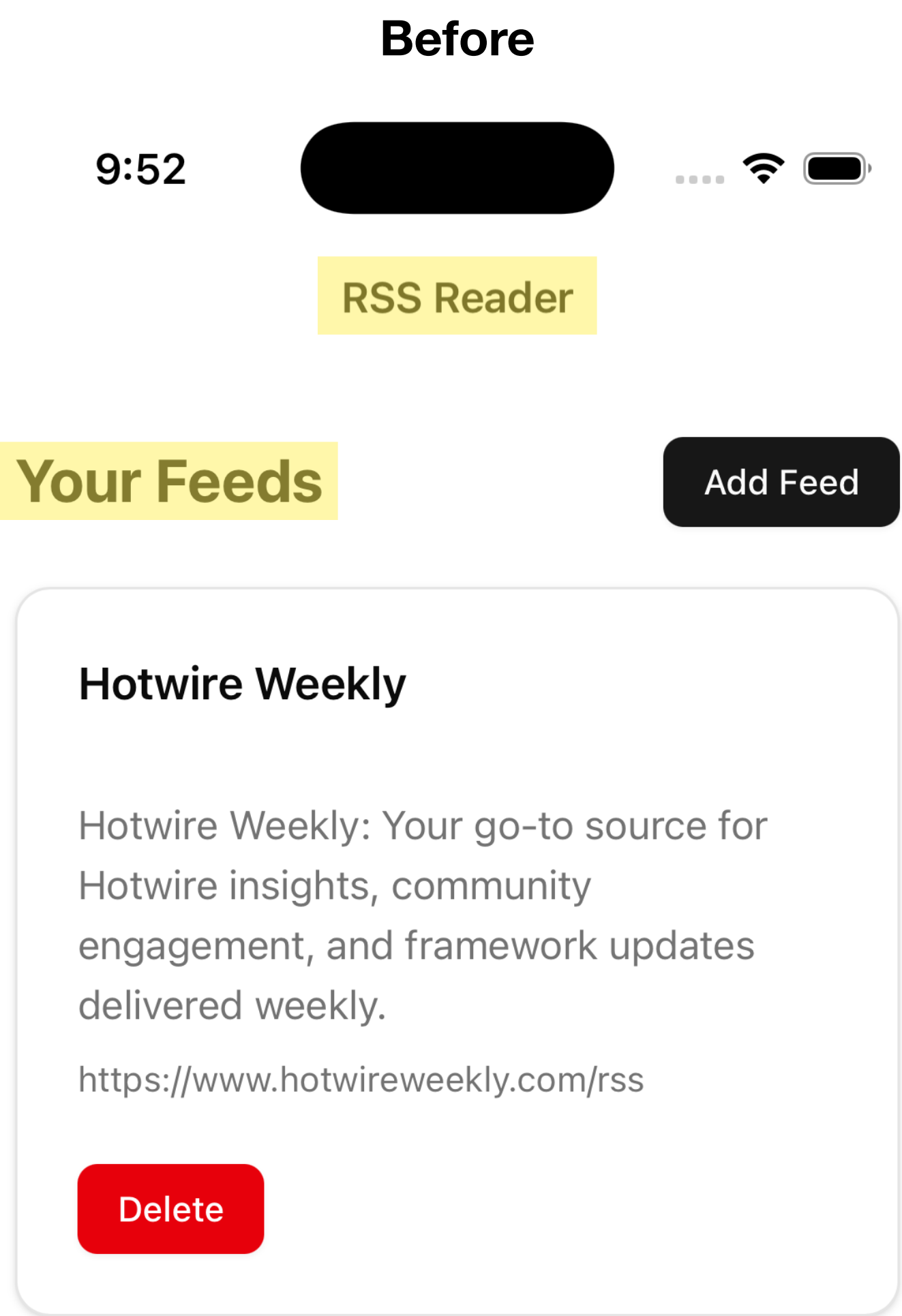
Native Screens

```
{
  "rules": [
    {
      "patterns": [
        "/numbers$"
      ],
      "properties": {
        "view_controller": "numbers"
      }
    }
  ]
}
```



Navigation Bar Title

Navigation Bar Title



Navigation Bar Title

Implementation steps

- ❑ Add page-specific `title` tag in Rails code base
- ❑ Hide `h1` tag in Rails code base

Navigation Bar Title

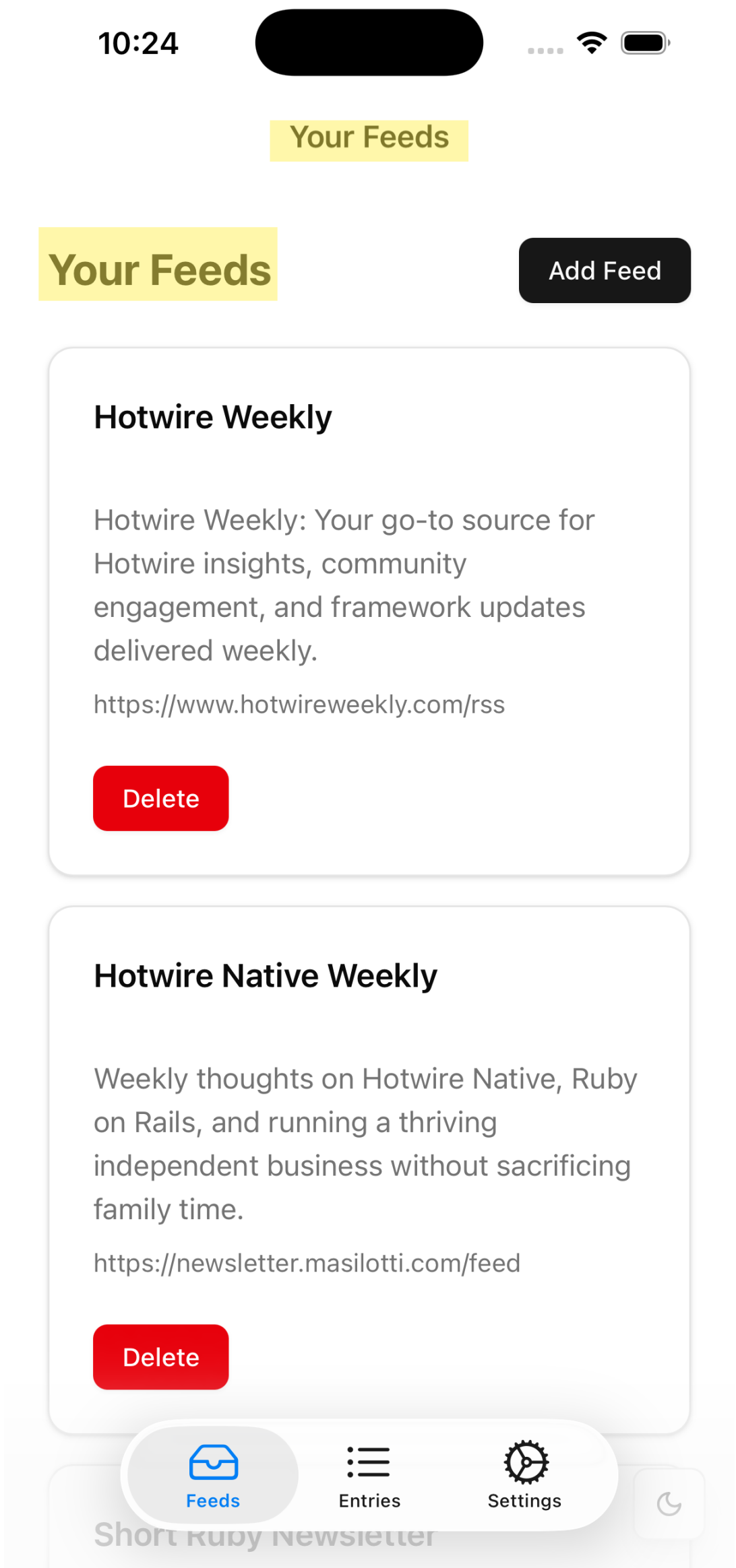
Add title tag

```
<%= content_for :title, "Your Feeds" %>

<div class="max-w-4xl mx-auto p-6 space-y-6">
  <div class="flex justify-between items-center">
    <h1 class="text-2xl font-bold">Your Feeds</h1>
    <%= link_to "Add Feed", new_feed_path, class: "btn-primary" %>
  </div>

  ...
</div>
```

index.html.erb



Navigation Bar Title

Implementation steps

- ☒ ~~Add page-specific title tag in Rails code base~~
- ☐ Hide h1 tag in Rails code base

Navigation Bar Title

Add hotwire-native CSS variant

```
<!DOCTYPE html>
<%= tag.html(
  data: {
    hotwire_native: hotwire_native_app?,
  },
) do %>
  <head>
    <title><%= content_for(:title) || "RSS Reader" %></title>
    ...
<% end %>
```

application.html.erb

```
@variant hotwire-native {
  html[data-hotwire-native="true"] & {
    @slot
  }
}
```

application.css

Navigation Bar Title

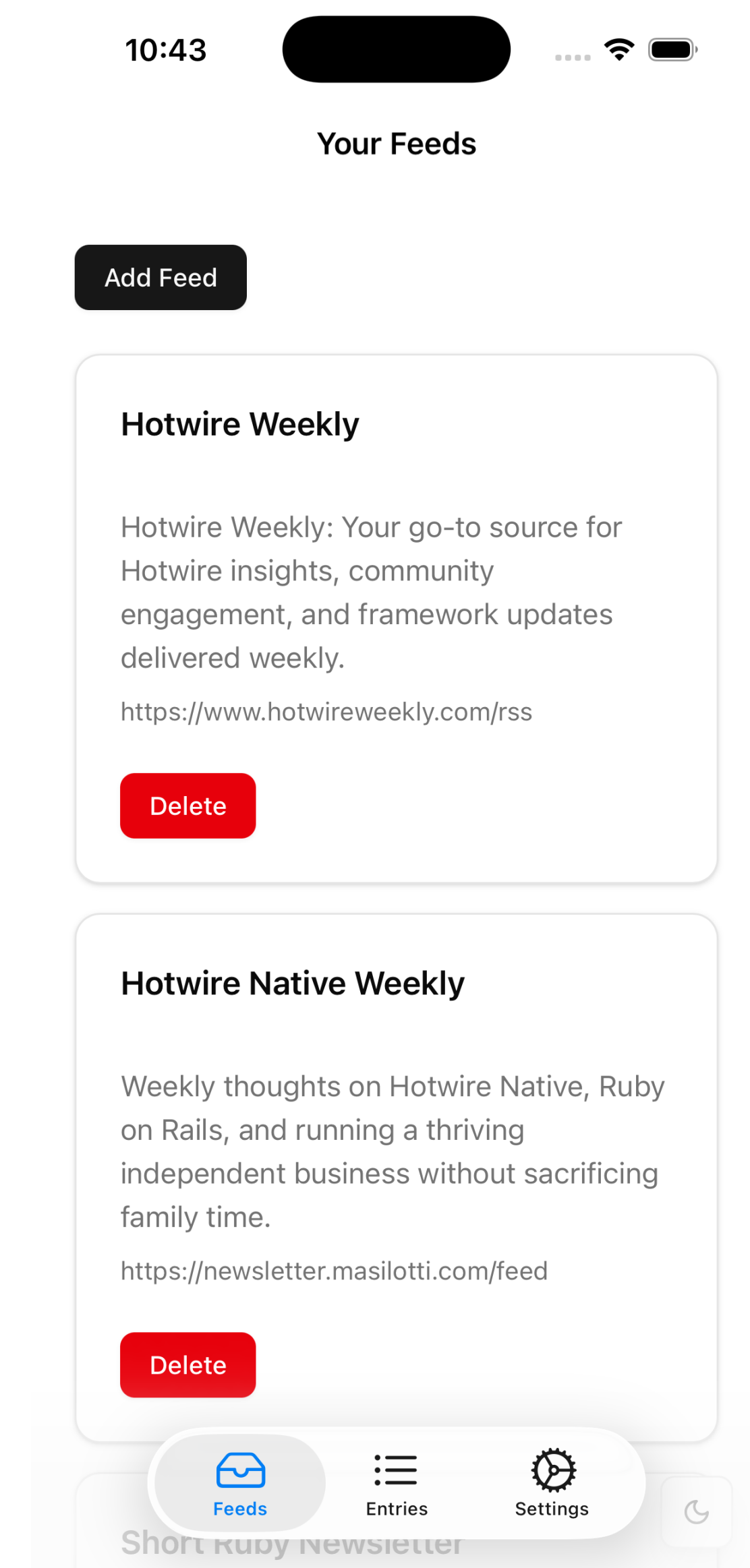
Hide h1 on Hotwire Native apps

```
<%= content_for :title, "Your Feeds" %>

<div class="max-w-4xl mx-auto p-6 space-y-6">
  <div class="flex justify-between items-center">
    <h1 class="hotwire-native:hidden text-2xl font-bold">Your Feeds</h1>
    <%= link_to "Add Feed", new_feed_path, class: "btn-primary" %>
  </div>

  ...
</div>
```

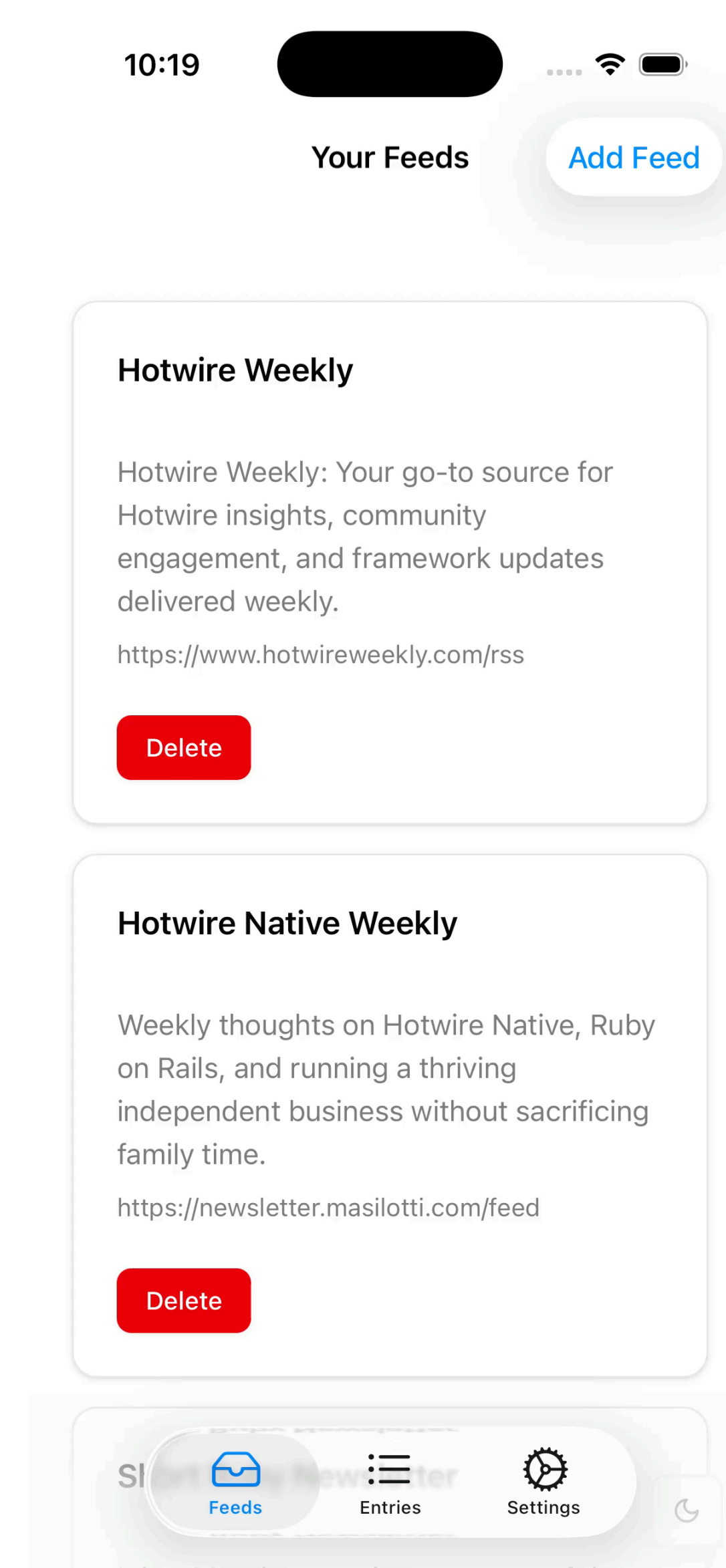
index.html.erb



Native Tab Bar

Native Tab Bar

- Built in to Hotwire Native
- Each tab is a...
 - separate web view
 - separate navigation stack



Native Tab Bar

Implementation steps

- ❑ Add native tab bar to iOS code base
- ❑ Add native tab bar to Android code base
- ❑ Hide web-based navigation in Rails code base

Native Tab Bar

ios

```
let baseUrl = URL(string: "http://localhost:3000")!

extension HotwireTab {
    static let all: [HotwireTab] = {
        var tabs: [HotwireTab] = [
            .feeds,
            ...
        ]

        return tabs
    }()

    static let feeds = HotwireTab(
        title: "Feeds",
        image: .init(systemName: "tray")!,
        url: baseUrl.appending(path: "/feeds")
    )

    ...
}
```

Tabs.swift

Native Tab Bar

ios

```
class SceneController: UIResponder {  
    ...  
  
    private lazy var tabBarController = HotwireTabBarController(navigatorDelegate: self)  
}  
  
extension SceneController: UIWindowSceneDelegate {  
    func scene(_ scene: UIScene, willConnectTo session: UISceneSession,  
              options connectionOptions: UIScene.ConnectionOptions) {  
        ...  
  
        tabBarController.load(HotwireTab.all)  
    }  
}
```

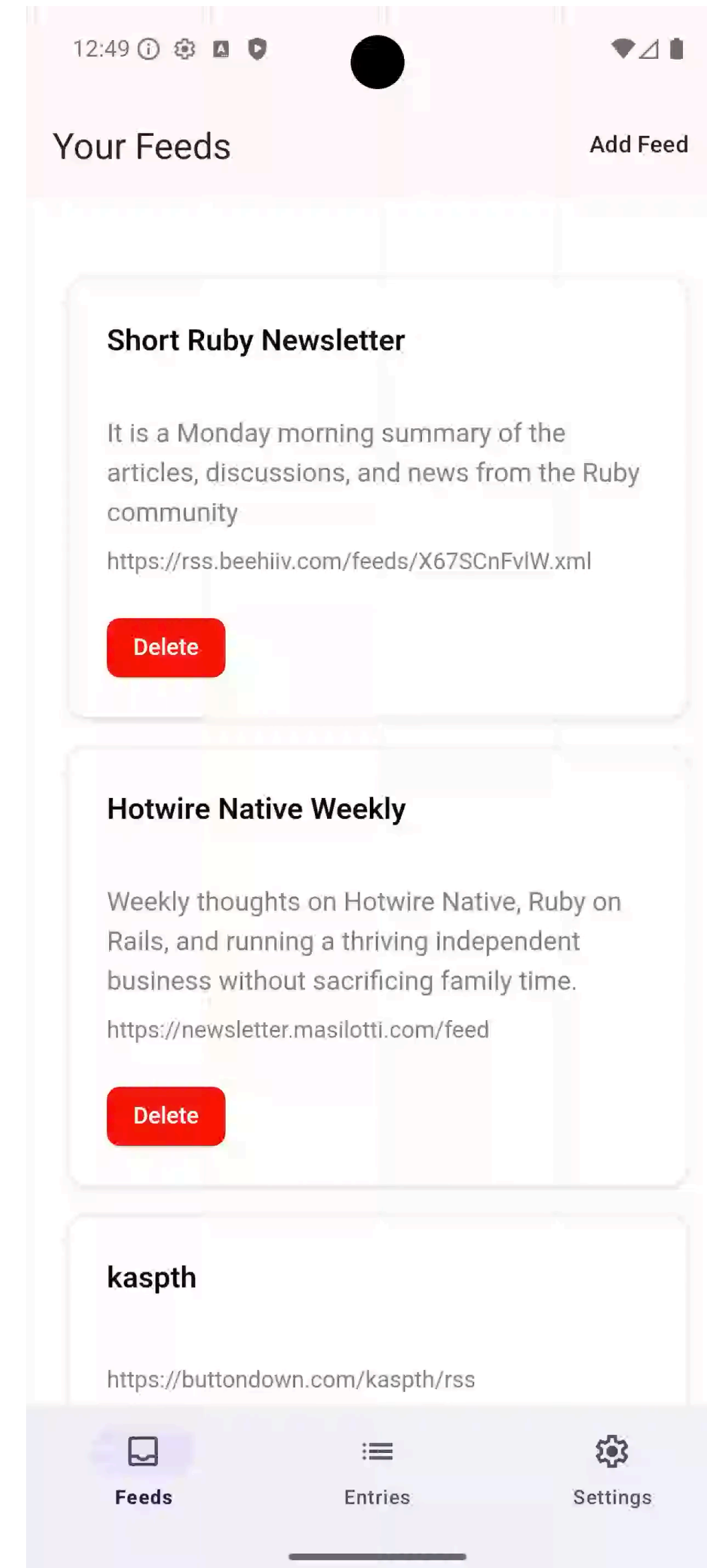
SceneController.swift

Native Tab Bar

Implementation steps

- ☒ ~~Add native tab bar to iOS code base~~
- ☐ Add native tab bar to Android code base
- ☐ Hide web-based navigation in Rails code base

Native Tab Bar Android



Native Tab Bar

Android

```
private val feeds = HotwireBottomTab(  
    title = "Feeds",  
    iconResId = R.drawable.inbox_24px,  
    configuration = NavigatorConfiguration(  
        name = "feeds",  
        navigatorHostId = R.id.feeds_nav_host,  
        startLocation = "$baseUrl/feeds"  
    )  
)  
  
...  
  
val mainTabs = listOf(  
    feeds,  
    ...  
)
```

Tabs.kt

Native Tab Bar Android

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    ...>

    <androidx.fragment.app.FragmentContainerView
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:id="@+id/feeds_nav_host"
        android:name="dev.hotwire.navigation.navigator.NavigatorHost"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:defaultNavHost="false"
        app:layout_constraintBottom_toTopOf="@id/bottom_nav"
        app:layout_constraintTop_toTopOf="parent" />

    ...

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/bottom_nav"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:labelVisibilityMode="labeled"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

activity_main.xml
```

Native Tab Bar

Android

```
class MainActivity : HotwireActivity() {  
    private lateinit var bottomNavController: HotwireBottomNavController  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        ...  
  
        initializeBottomTabs()  
    }  
  
    private fun initializeBottomTabs() {  
        val bottomNavigationView = findViewById<BottomNavigationView>(R.id.bottom_nav)  
        bottomNavController = HotwireBottomNavController(this, bottomNavigationView)  
        bottomNavController.load(mainTabs, 0)  
    }  
}
```

MainActivity.kt

Native Tab Bar

Implementation steps

- ☒ ~~Add native tab bar to iOS code base~~
- ☒ ~~Add native tab bar to Android code base~~
- ☐ Hide web-based navigation in Rails code base

Native Tab Bar

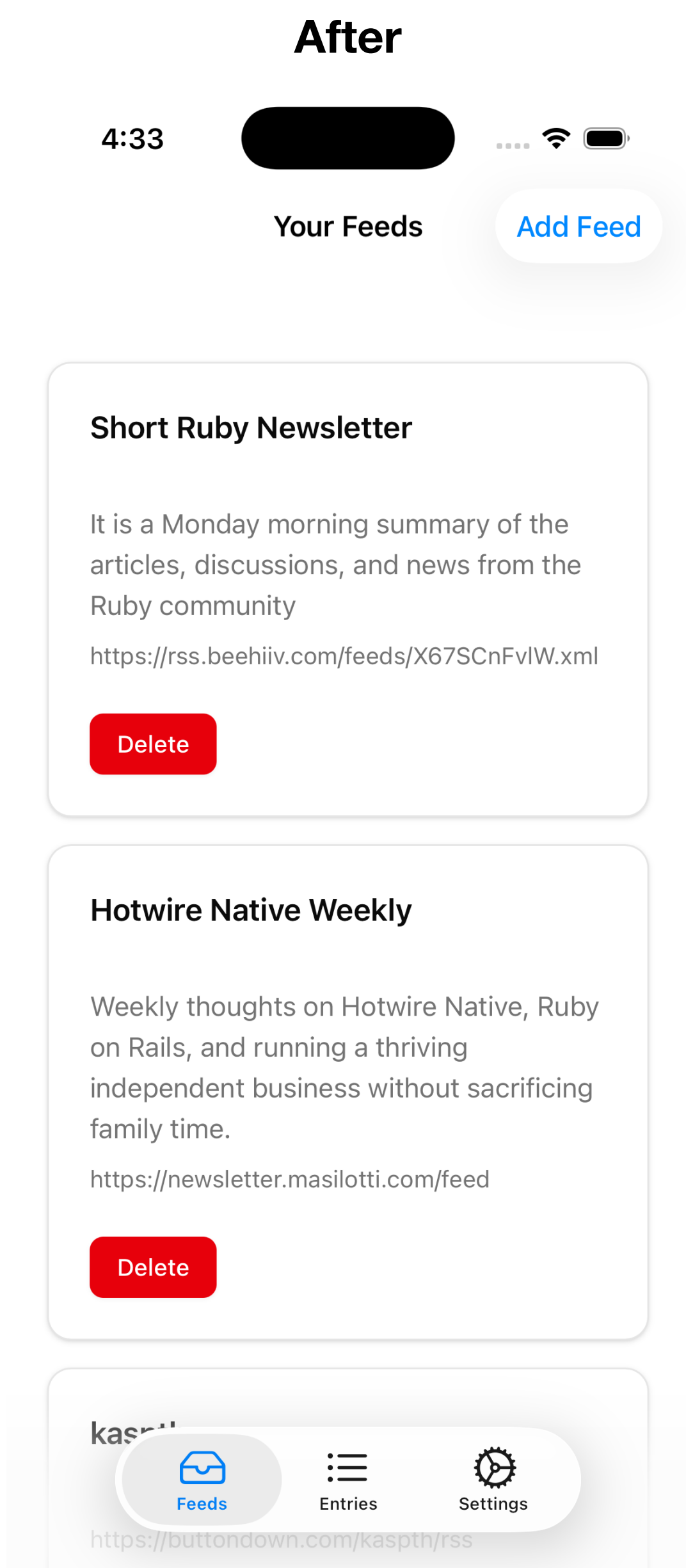
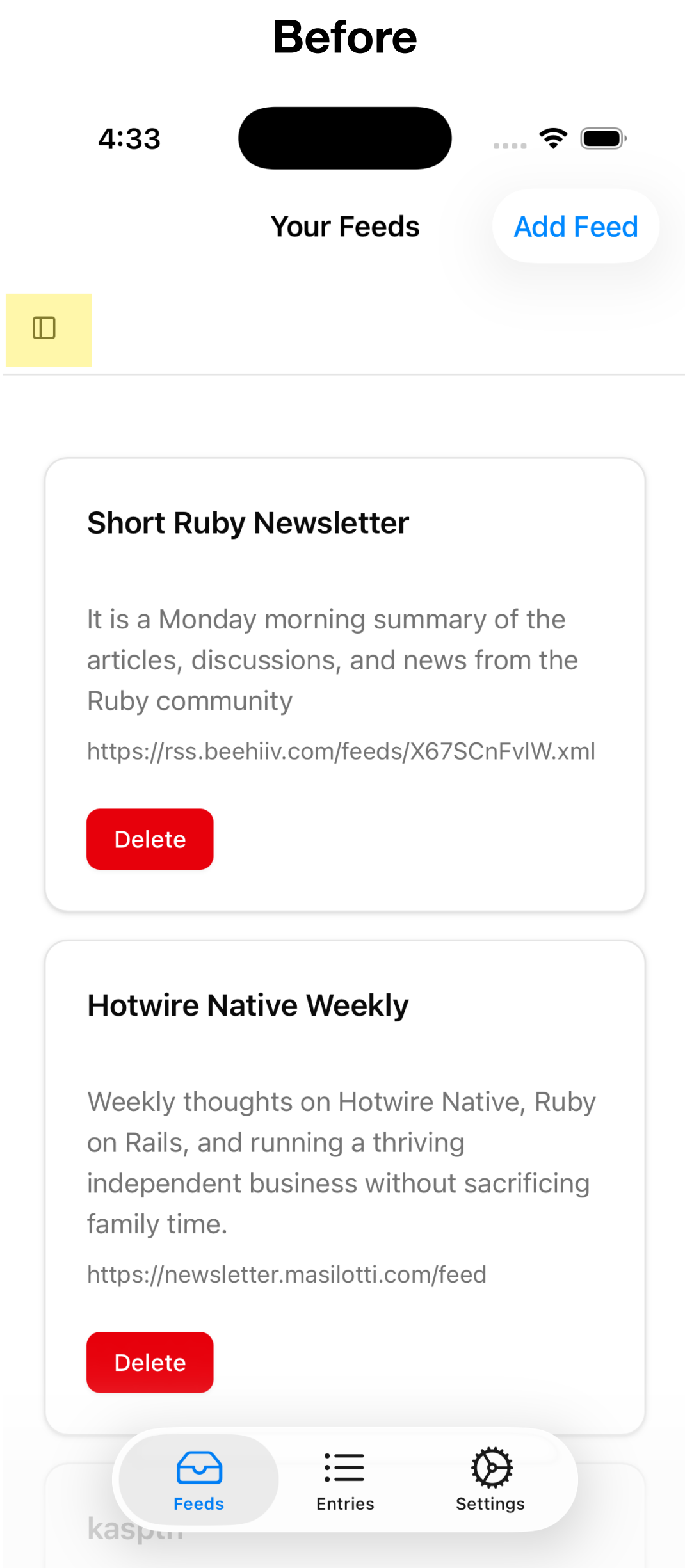
Hide Web Navigation

```
<body>
  ...

  <main>
    <% if authenticated? %>
      <header class="hotwire-native:hidden flex ..." >
        <button...>
        </button>
      </header>
    <% end %>

    <%= yield %>
  </main>

  ...
</body>
```



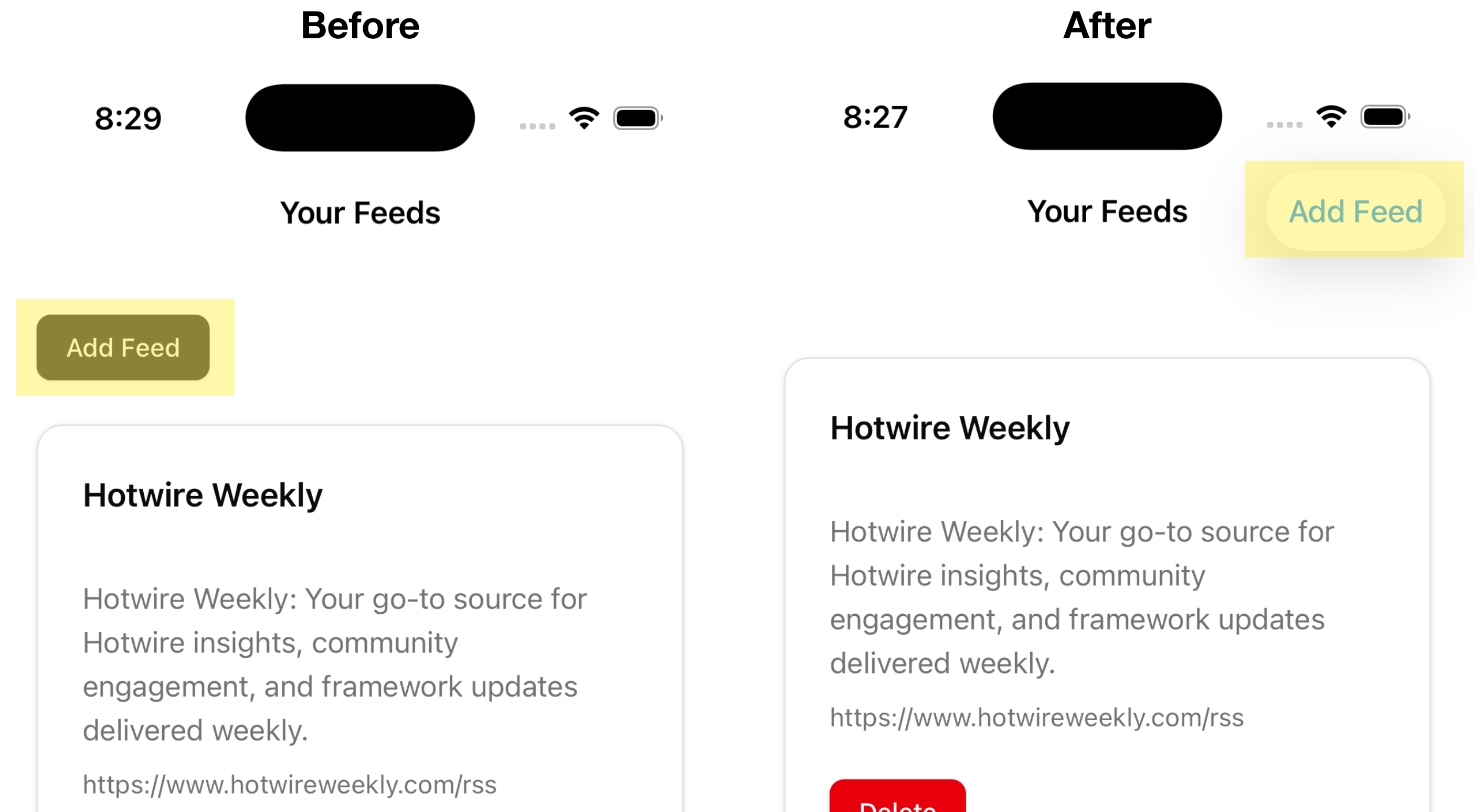
Bridge Components

Bridge Components

- Formerly called Strada
- Three parts
 - Stimulus controller
 - iOS component
 - Android fragment

Bridge Components

Navigation Bar Button



Bridge Components

Navigation Bar Button

- ❑ Add Stimulus controller
- ❑ Update the `link_to` helper to use the Stimulus controller
- ❑ Add iOS component
- ❑ Add Android component

Bridge Components

Navigation Bar Button

```
import { BridgeComponent } from "@hotwired/hotwire-native-bridge"

export default class extends BridgeComponent {
  static component = "button"

  connect() {
    super.connect()

    const element = this.bridgeElement
    const title = element.bridgeAttribute("title")
    this.send("connect", {title}, () => {
      this.element.click()
    })
  }
}
```

app/javascript/controllers/bridge/button_controller.js

Bridge Components

Navigation Bar Button

```
<%= link_to "Add Feed", new_feed_path,  
  class: "hotwire-native:hidden btn-primary",  
  data: {  
    controller: "bridge--button",  
    bridge_title: "Add Feed",  
  } %>
```

app/views/feeds/index.html.erb

Bridge Components

Navigation Bar Button

```
import HotwireNative
import UIKit

final class ButtonComponent: BridgeComponent {
    override class var name: String { "button" }

    override func onReceive(message: Message) {
        guard let viewController else { return }
        addButton(via: message, to: viewController)
    }

    private var viewController: UIViewController? {
        delegate?.destination as? UIViewController
    }

    private func addButton(via message: Message, to viewController: UIViewController) {
        guard let data: MessageData = message.data() else { return }

        let action = UIAction { [unowned self] _ in
            self.reply(to: "connect")
        }
        let item = UIBarButtonItem(title: data.title, primaryAction: action)
        viewController.navigationItem.rightBarButtonItem = item
    }
}

private extension ButtonComponent {
    struct MessageData: Decodable {
        let title: String
    }
}
```

ButtonComponent.swift

Bridge Components

Navigation Bar Button

```
import HotwireNative
import UIKit

@main
class AppDelegate: UIResponder, UIApplicationDelegate {
    func application(_ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        Hotwire.registerBridgeComponents([
            ButtonComponent.self
        ])
        return true
    }
}
```

AppDelegate.swift

Bridge Components

Navigation Bar Button

```
class ButtonComponent(  
    name: String,  
    private val delegate: BridgeDelegate<HotwireDestination>  
) : BridgeComponent<HotwireDestination>(name, delegate) {  
  
    override fun onReceive(message: Message) {  
        // Handle incoming messages based on the message `event`.  
        when (message.event) {  
            "connect" -> handleConnectEvent(message)  
            else -> Log.w("ButtonComponent", "Unknown event for message: $message")  
        }  
    }  
  
    private fun handleConnectEvent(message: Message) {  
        val data = message.data<MessageData>() ?: return  
  
        // Write native code to display a native submit button in the  
        // toolbar displayed in the delegate.destination. Use the  
        // incoming data.title to set the button title.  
    }  
  
    private fun performButtonClick(): Boolean {  
        return replyTo("connect")  
    }  
  
    // Use kotlinx.serialization annotations to define a serializable  
    // data class that represents the incoming message.data json.  
    @Serializable  
    data class MessageData(  
        @SerializedName("title") val title: String  
    )  
}
```

ButtonComponent.kt

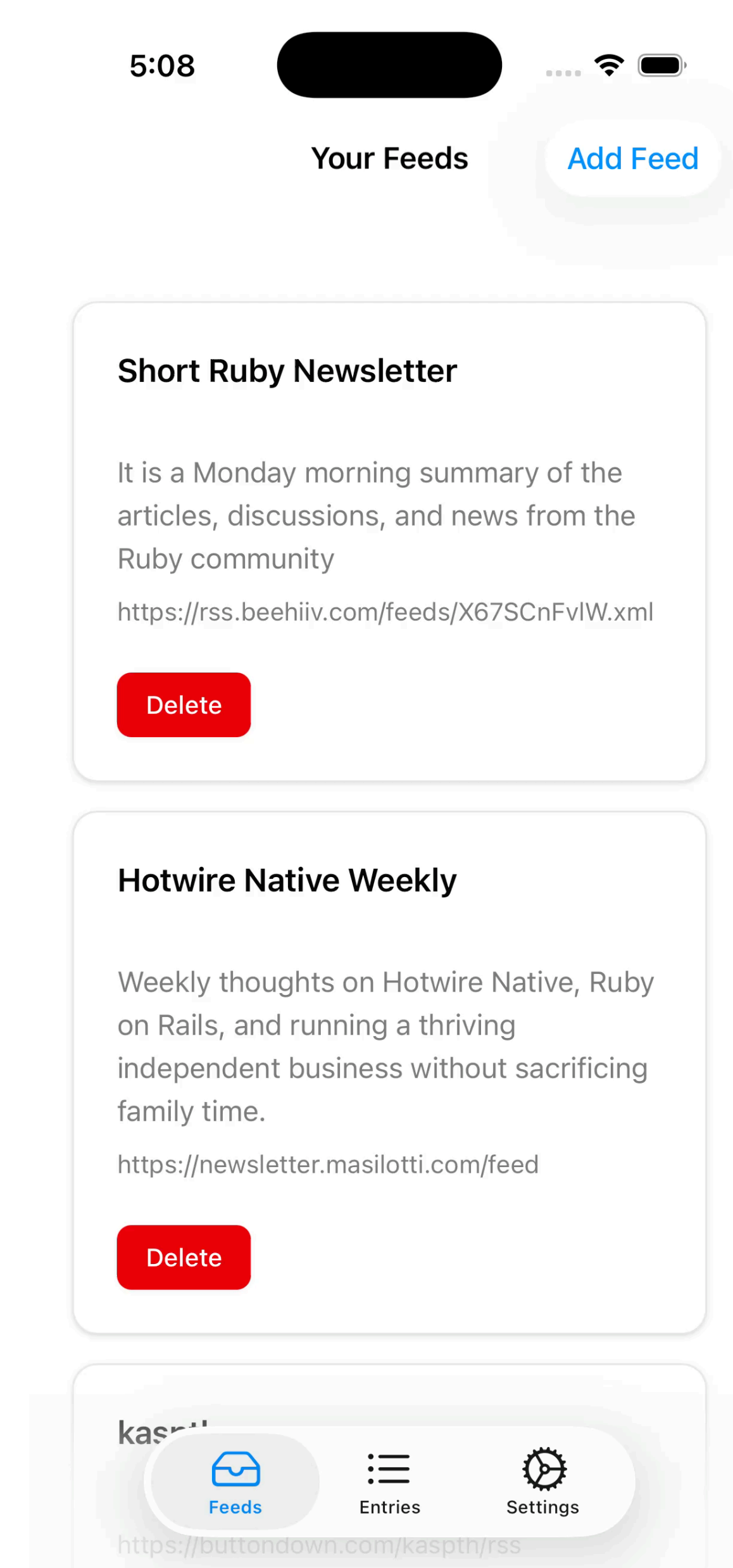
Bridge Components

Navigation Bar Button

```
Hotwire.registerBridgeComponents(  
  BridgeComponentFactory("button", ::ButtonComponent)  
)
```

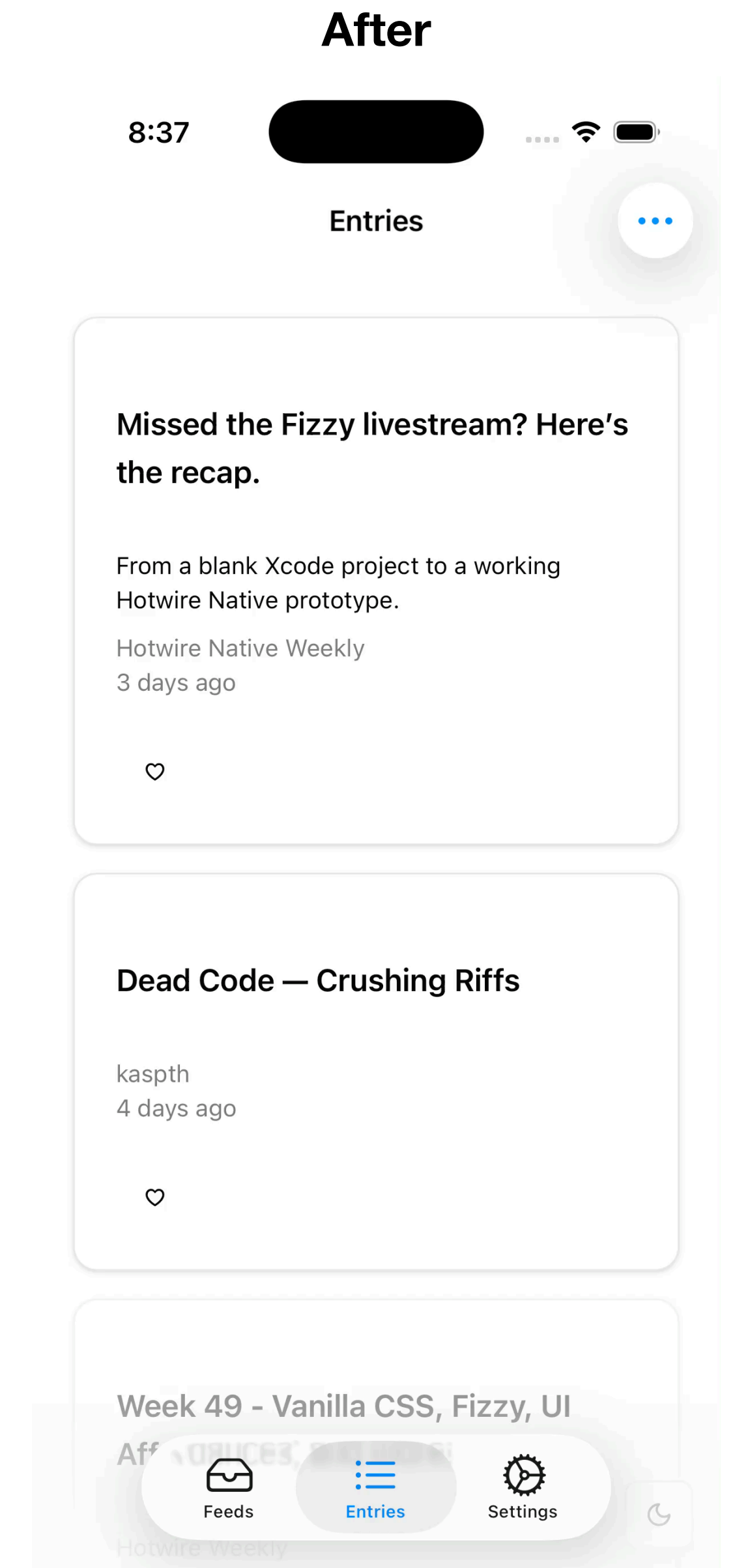
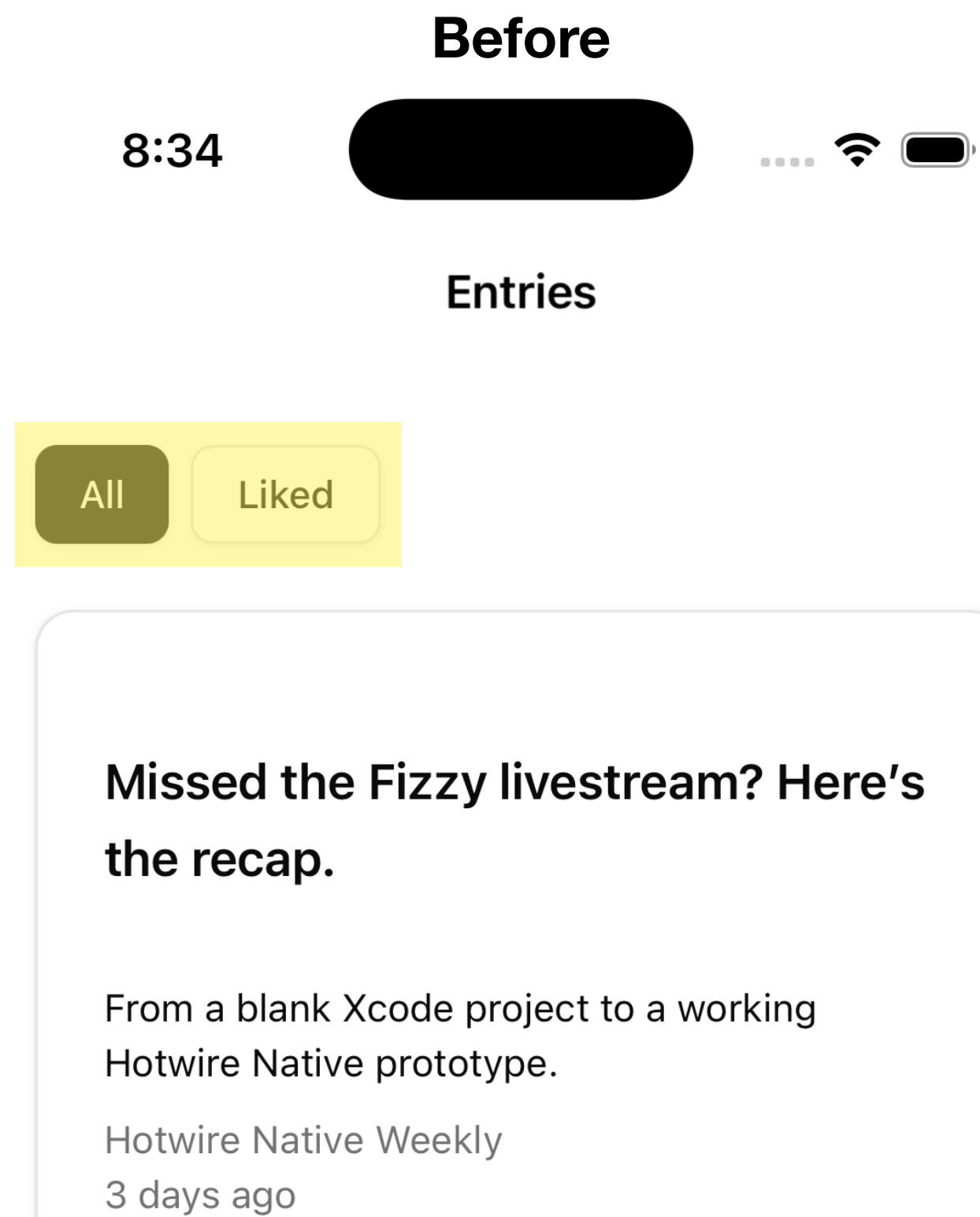

Bridge Components

Navigation Bar Button



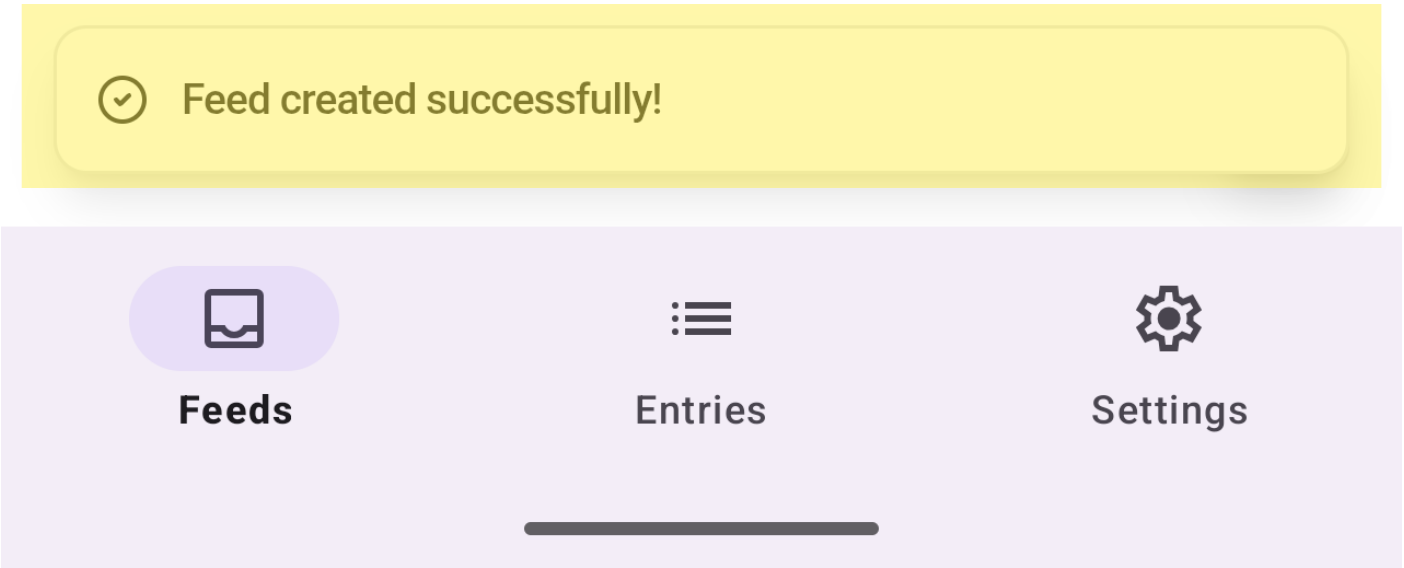
Bridge Components

Navigation Bar Menu

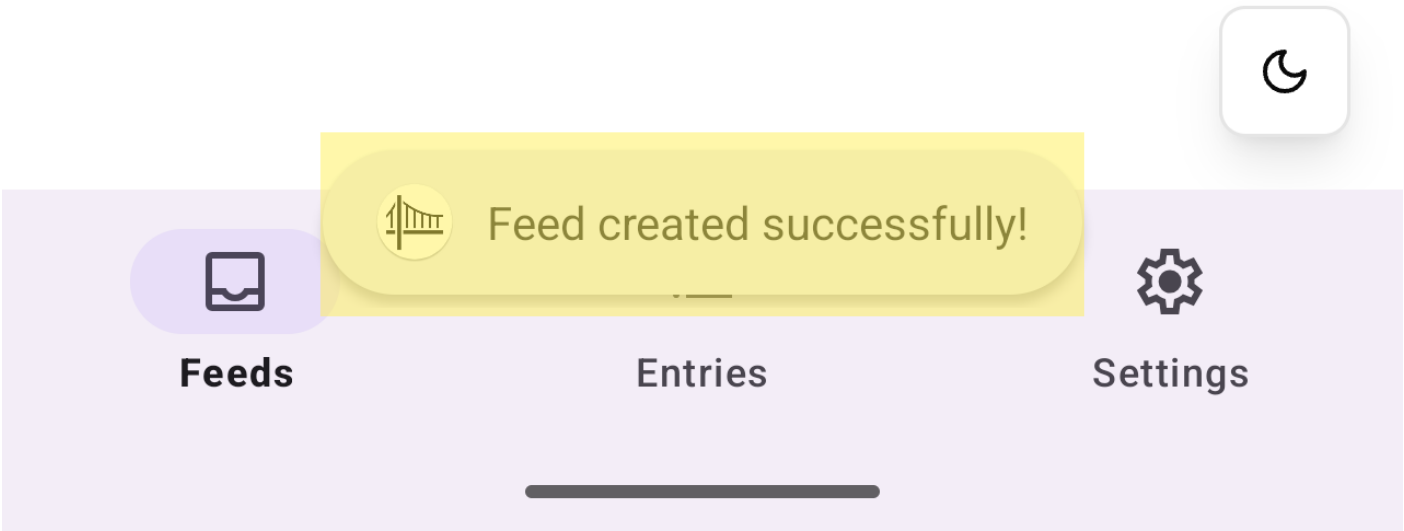


Bridge Components Toast Messages

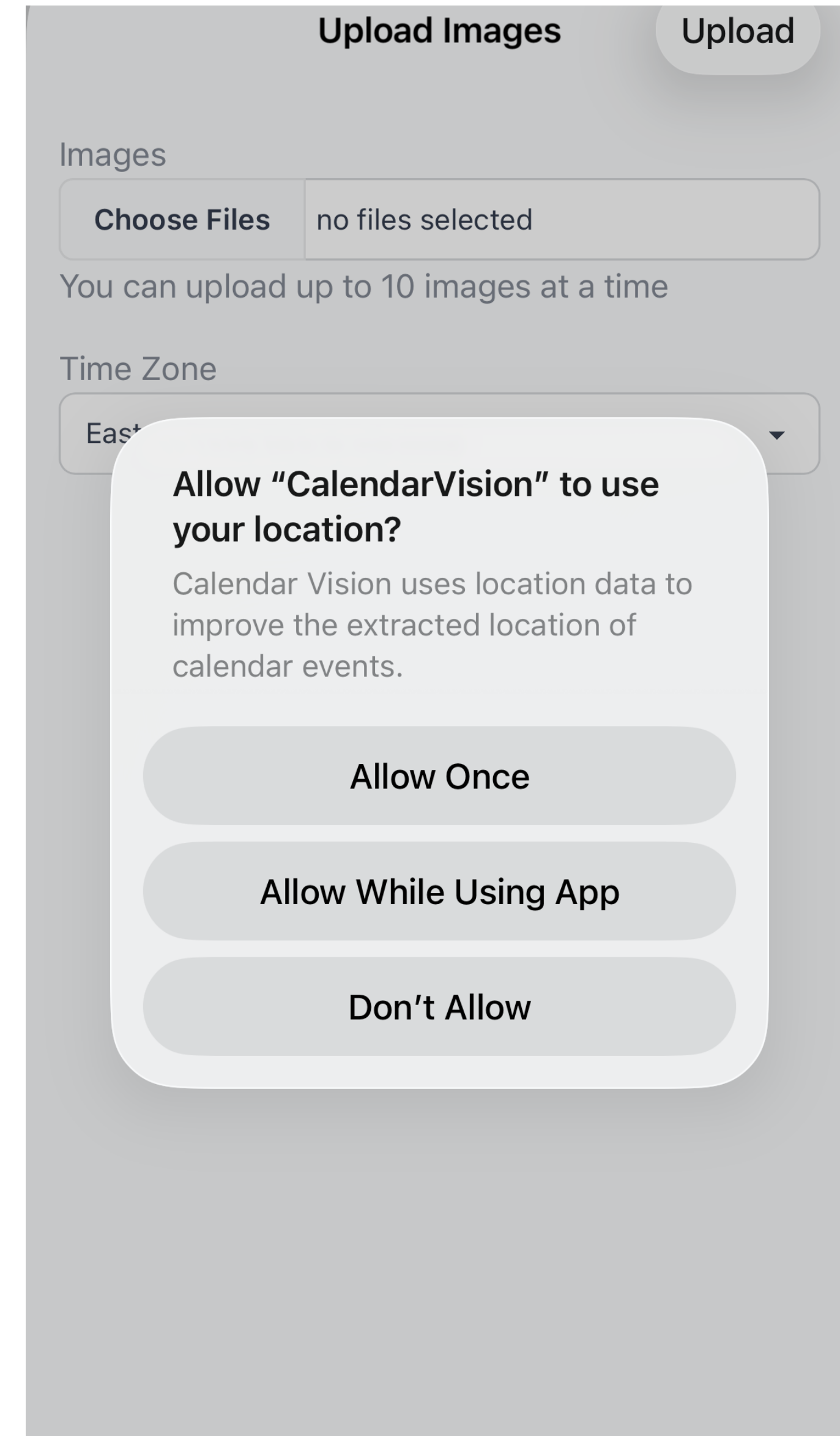
Before



After



Bridge Components Request Permissions



Bridge Components

Joe Masilotti's library

Components

Newsletter

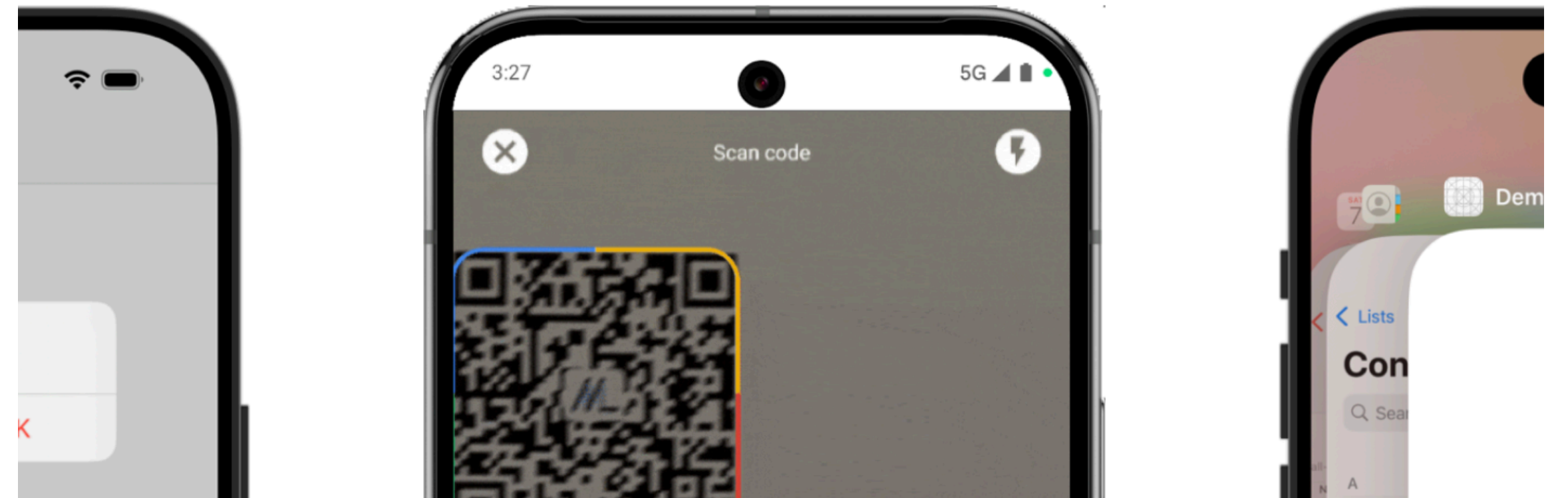
Installation

PRO

Bridge Components

Native integrations for Hotwire Native apps

Build iOS and Android features with HTML,
no Swift or Kotlin required.



<https://masilotti.com/bridge-components/>

Supporting OAuth

Supporting OAuth

- Sign in with Apple
- Sign in with Google
- Sign in with ...

Password

[Forgot your password?](#)

Sign in

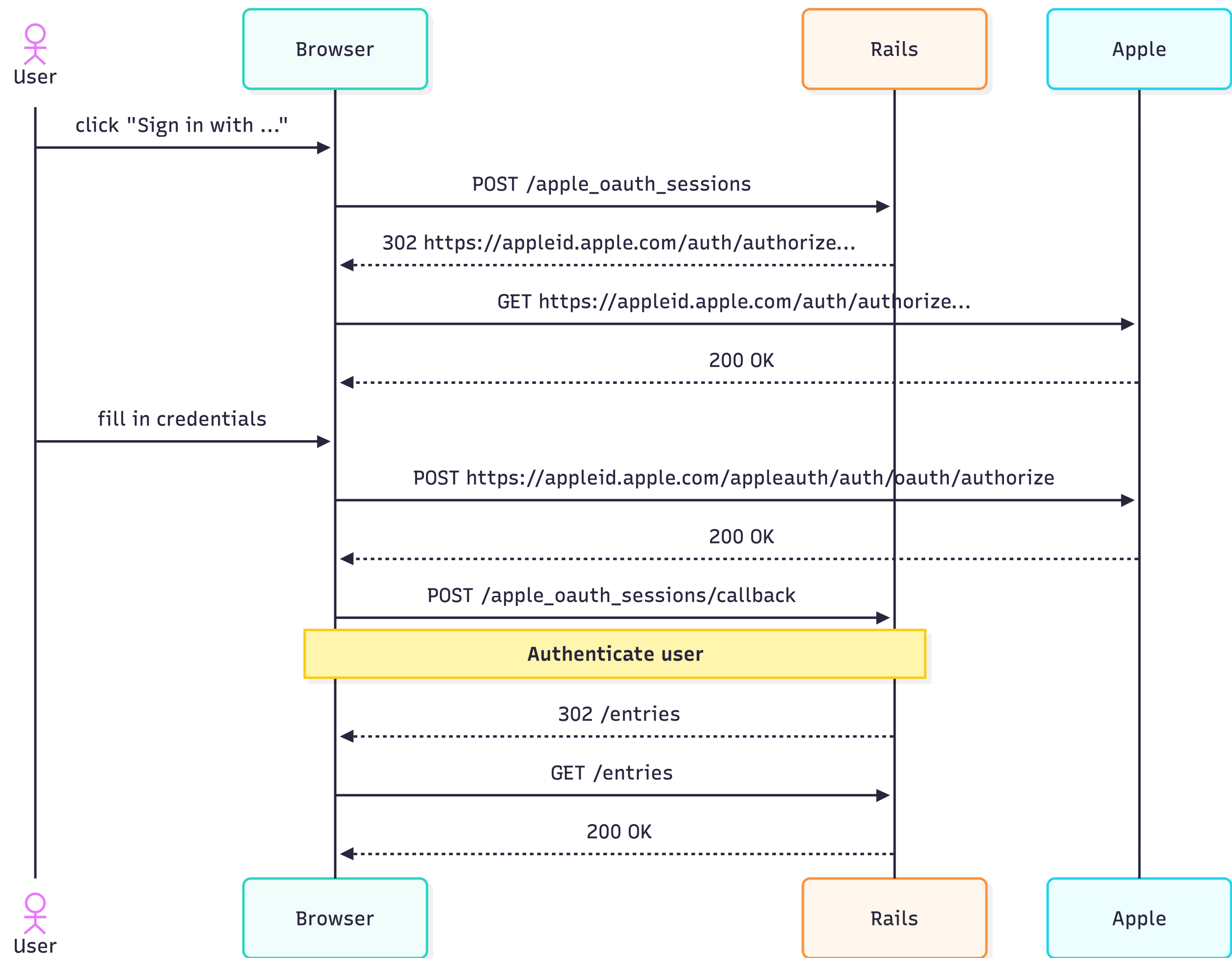
OR CONTINUE WITH

Sign in with Apple

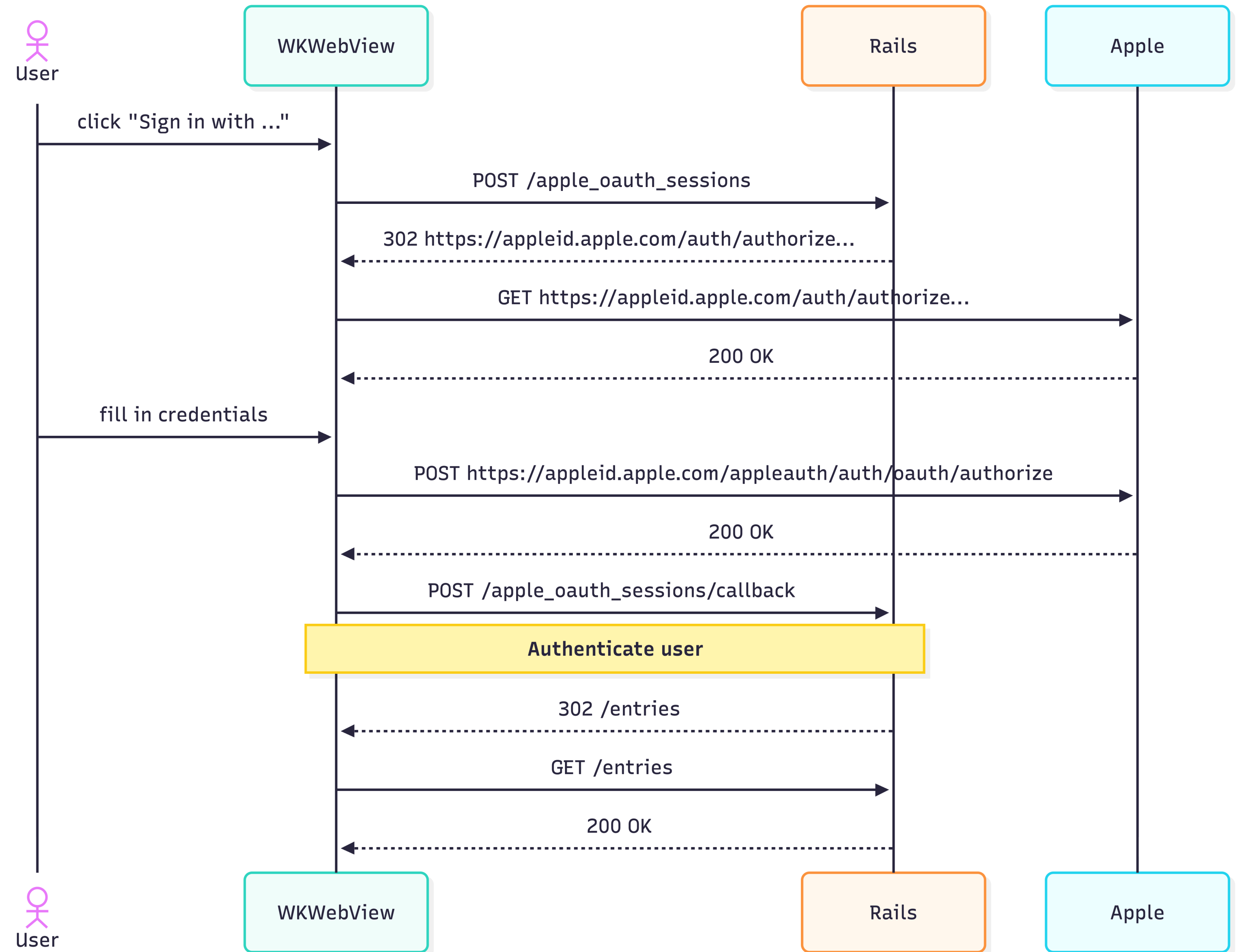
Sign in with Google

[Don't have an account? Sign up](#)

Supporting OAuth Web



Supporting OAuth Native (Web view)



 Search all articles...

Search

ANDROID

Upcoming security changes to Google's OAuth 2.0 authorization endpoint in embedded webviews

JUNE 29, 2021

 Share

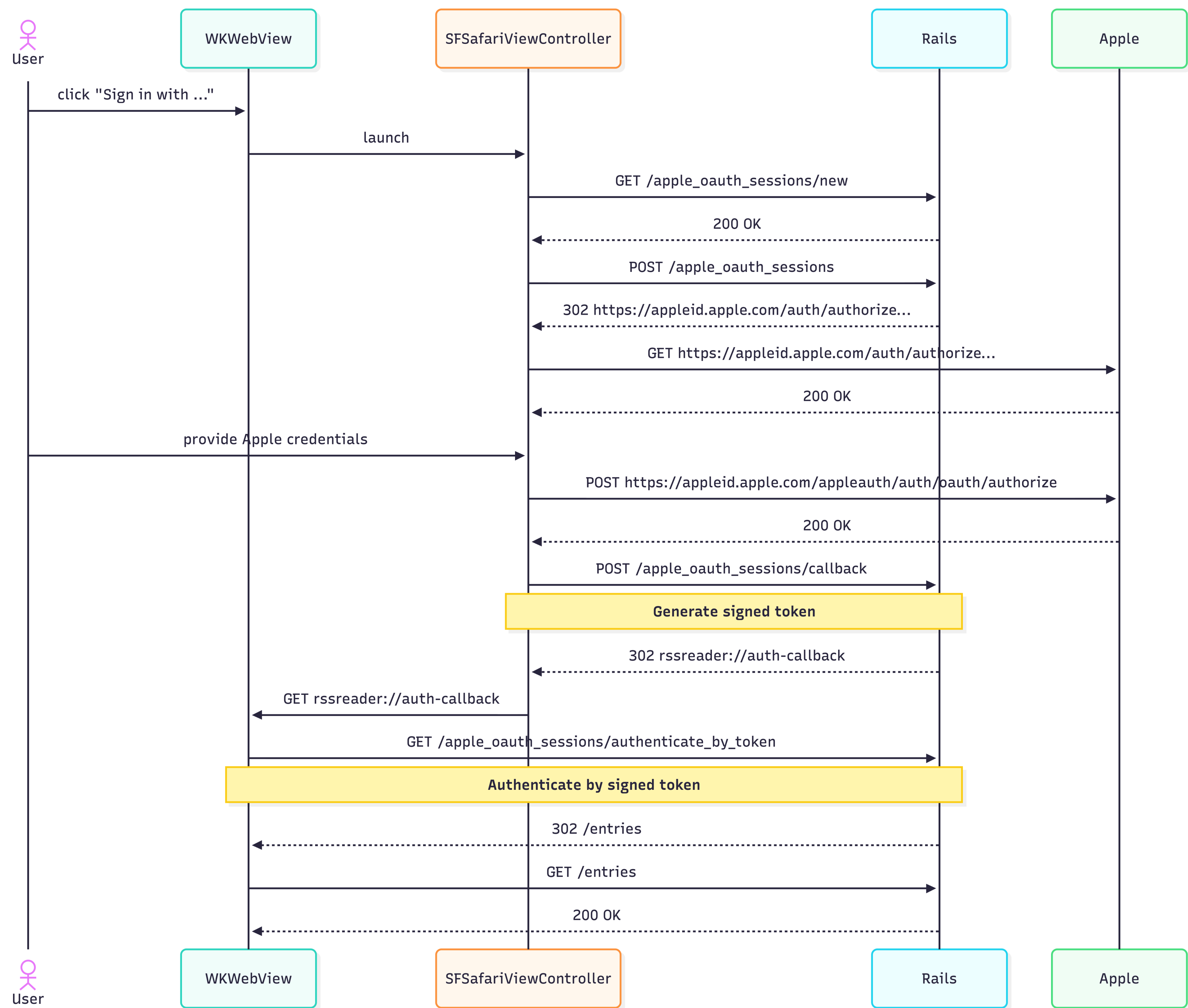
Posted by Badi Azad, Group Product Manager (@badiazad)

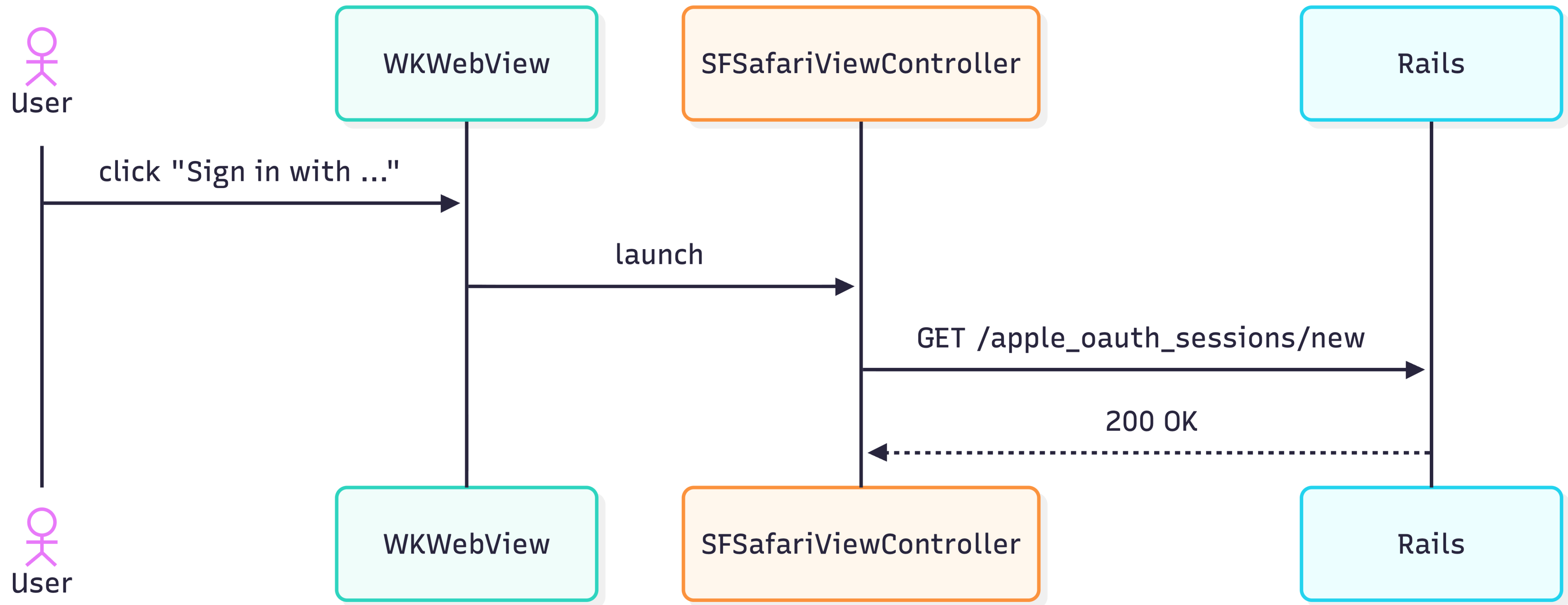
The Google Identity team is continually working to improve Google Account security and create a safer and more secure experience for our users. As part of that work, we recently introduced a new [secure browser policy](#) prohibiting Google OAuth requests in embedded browser libraries commonly referred to as embedded webviews. All embedded webviews will be blocked starting on **September 30**,

...[Google] introduced a new secure browser policy prohibiting Google OAuth requests in embedded browser libraries commonly referred to as embedded webviews. All embedded webviews will be blocked...

<https://developers.googleblog.com/upcoming-security-changes-to-googles-oauth-20-authorization-endpoint-in-embedded-webviews/>

Supporting OAuth Native (External)





- ❑ Create Stimulus part of Bridge Component
- ❑ Update “Sign in with ...” button to use Bridge component
- ❑ Create view in Rails to initiate OAuth process
- ❑ Create iOS Bridge Component

```

import { BridgeComponent } from "@hotwired/hotwire-native-bridge"

export default class extends BridgeComponent {
  static component = "sign-in-with-oauth"
  static values = {
    startPath: String
  }

  interceptSubmit(event) {
    event.preventDefault()

    const startPath = this.startPathValue
    this.send("click", { startPath })
  }
}

```

app/javascript/controllers/bridge/sign_in_with_oauth_controller.js

```

<%= form_with(
  url: apple_oauth_sessions_path,
  method: :post,
  data: {
    controller: "bridge--sign-in-with-oauth",
    action: "submit->bridge--sign-in-with-oauth#interceptSubmit",
    bridge__sign_in_with_oauth_start_path_value: new_apple_oauth_sessions_path
  }) do |form| %>
  <%= form.submit "Sign in with Apple", class: "btn-outline w-full" %>
<% end %>

```

app/views/shared/_sign_in_with_apple.html.erb

- ☒ ~~Create Stimulus part of Bridge Component~~
- ☒ ~~Update “Sign in with ...” button to use Bridge component~~
- ☐ Create view in Rails to initiate OAuth process
- ☐ Create iOS Bridge Component


```

class AppleOauthSessionsController < ApplicationController
  skip_before_action :verify_authenticity_token
  allow_unauthenticated_access

  def new
    render :new, layout: false
  end

  ...
end

```

app/controllers/apple_oauth_sessions_controller.rb

```

<%= form_with(
  url: apple_oauth_sessions_path,
  method: :post,
  data: {
    controller: "form-submit",
    turbo: false
  }) do |form| %>
  <%= form.hidden_field :platform, value: params[:platform] %>
<% end %>

```

app/views/apple_oauth_sessions/new.html.erb

```

import { Controller } from "@hotwired/stimulus"

export default class extends Controller {
  connect() {
    this.element.requestSubmit()
  }
}

```

app/javascript/controllers/form_submit_controller.js

- ☒ ~~Create Stimulus part of Bridge Component~~
- ☒ ~~Update “Sign in with ...” button to use Bridge component~~
- ☒ ~~Create view in Rails to initiate OAuth process~~
- ☐ Create iOS Bridge Component


```

class SignInWithOauthComponent: BridgeComponent {
    ...

    private var safariViewController: SFSafariViewController?

    override func onReceive(message: Message) {
        guard let event = Event(rawValue: message.event) else { return }

        switch event {
        case .click:
            onClick(message: message)
        }
    }

    private func onClick(message: Message) {
        guard let data: MessageData = message.data(),
              let startUrl = URL(string: "\(baseUrl)\(data.startPath)") else { return }

        launchSafariViewController(with: startUrl)
    }

    private func launchSafariViewController(with url: URL) {
        let safariVC = SFSafariViewController(url: url)
        safariVC.modalPresentationStyle = .pageSheet
        self.safariViewController = safariVC

        viewController.present(safariVC, animated: true)
    }
}

private extension SignInWithOauthComponent {
    enum Event: String {
        case click
    }

    struct MessageData: Decodable {
        let startPath: String
    }
}

```

SignInWithOauthComponent.swift

```

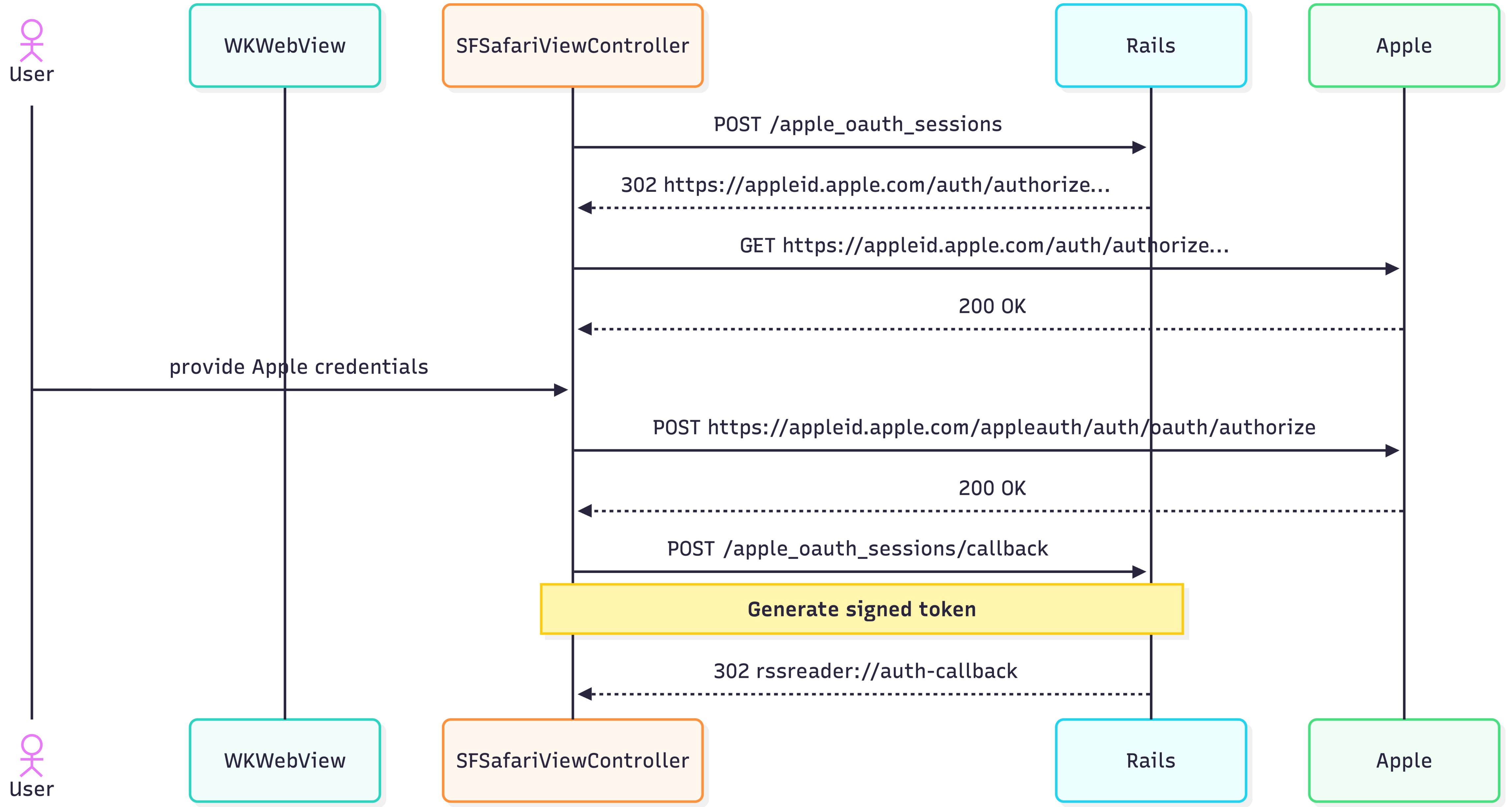
@main
class AppDelegate: UIResponder, UIApplicationDelegate {
    func application(
        ...
    ) -> Bool {
        ...

        Hotwire.registerBridgeComponents([
            SignInWithOauthComponent.self,
        ])

        ...
    }
}

```

AppDelegate.swift

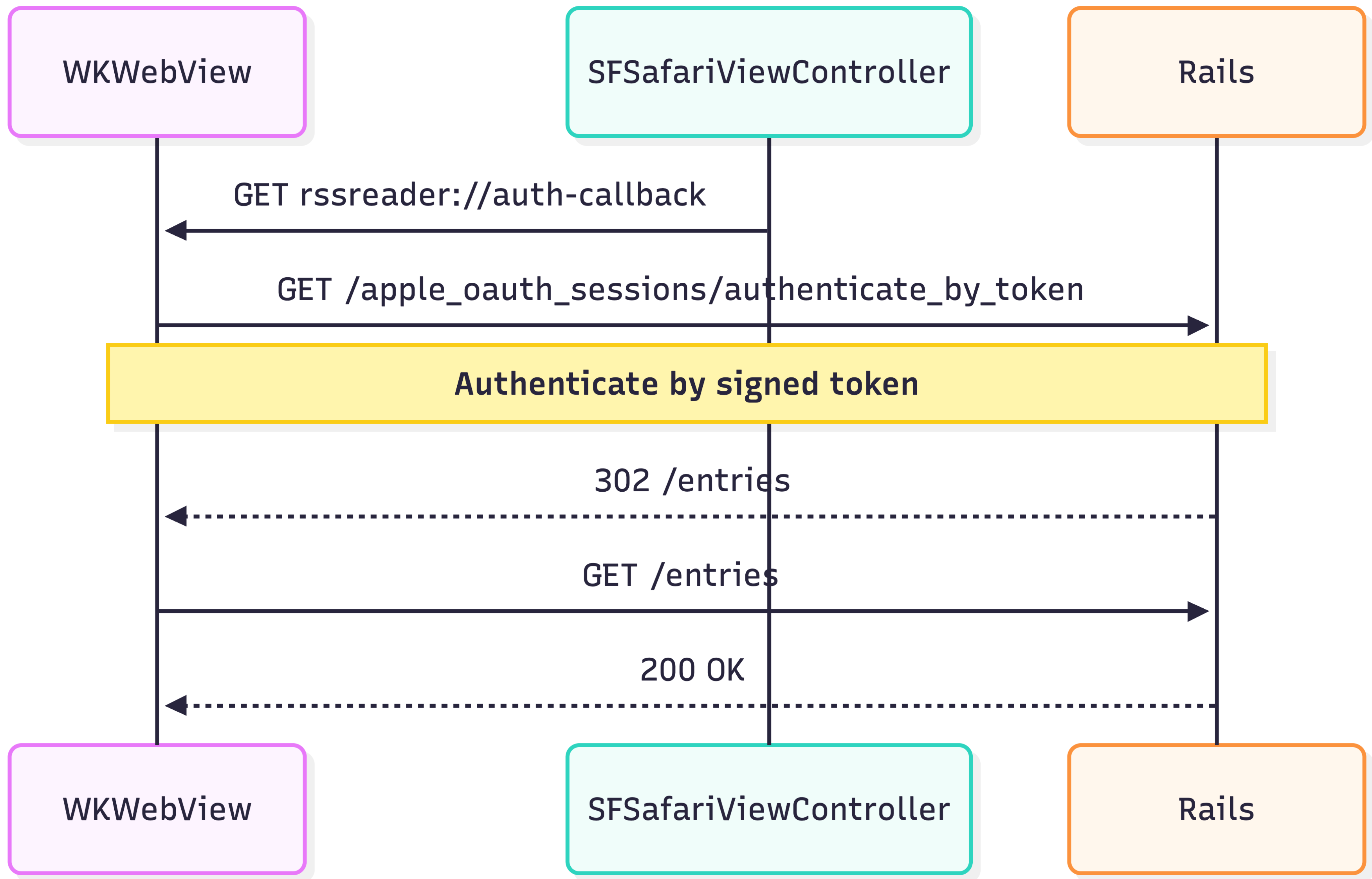


```
class AppleOauthSessionsController < ApplicationController
  ...

  def callback
    user_info = authenticate_with_apple
    user = create_user(user_info)
    unless user.persisted?
      redirect_to new_session_path, alert: "Unable to sign in. Please try again."
      return
    end

    token = user.signed_id(purpose: :native_auth, expires_in: 5.minutes)
    redirect_to "rssreader://auth-callback?token=#{token}&platform=#{platform}", allow_other_host: true
  end
end
```

app/controllers/apple_oauth_sessions_controller.rb



- ❑ Update Bridge Component so it passes the token authentication path
- ❑ Add the token authentication endpoint to Rails
- ❑ Implement OAuth callback in iOS app

```
<%= form_with(
  url: apple_oauth_sessions_path,
  method: :post,
  data: {
    controller: "bridge--sign-in-with-oauth",
    action: "submit->bridge--sign-in-with-oauth#interceptSubmit",
    bridge__sign_in_with_oauth_start_path_value: new_apple_oauth_sessions_path(platform: "native"),
    bridge__sign_in_with_oauth_token_auth_path_value: authenticate_by_token_apple_oauth_sessions_path
  }) do |form| %>
  <%= form.submit "Sign in with Apple", class: "btn-outline w-full" %>
<% end %>
```

app/views/apple_oauth_sessions/new.html.erb

```
import { BridgeComponent } from "@hotwired/hotwire-native-bridge"

export default class extends BridgeComponent {
  static component = "sign-in-with-oauth"
  static values = {
    startPath: String,
    tokenAuthPath: String,
  }

  interceptSubmit(event) {
    event.preventDefault()

    const startPath = this.startPathValue
    const tokenAuthPath = this.tokenAuthPathValue
    this.send("click", { startPath, tokenAuthPath })
  }
}
```

app/javascript/controllers/bridge/sign_in_with_oauth_controller.js


```

class SignInWithOauthComponent: BridgeComponent {
    ...

    private var tokenAuthPath: String?

    override func onReceive(message: Message) {
        guard let event = Event(rawValue: message.event) else { return }

        switch event {
        case .click:
            onClick(message: message)
        }
    }

    private func onClick(message: Message) {
        ...

        self.tokenAuthPath = data.tokenAuthPath

        launchSafariViewController(with: startUrl)
    }

    ...
}

private extension SignInWithOauthComponent {
    enum Event: String {
        case click
    }

    struct MessageData: Decodable {
        let startPath: String
        let tokenAuthPath: String
    }
}

```

SignInWithOauthComponent.swift

- ☒ ~~Update Bridge Component so it passes the token authentication path~~
- ☐ Add the token authentication endpoint to Rails
- ☐ Implement OAuth callback in iOS app


```
class AppleOauthSessionsController < ApplicationController
  ...

  def authenticate_by_token
    user = User.find_signed(params[:token], purpose: :native_auth)
    if user
      sign_in_and_redirect_user(user)
    else
      redirect_to welcome_path, alert: "Unable to sign in. Please try again."
    end
  end
end
```

app/controllers/apple_oauth_sessions_controller.rb

- ☒ ~~Update Bridge Component so it passes the token authentication path~~
- ☒ ~~Add the token authentication endpoint to Rails~~
- ☐ Implement OAuth callback in iOS app

```

class SignInWithOauthComponent: BridgeComponent {
    ...

    private func launchSafariViewController(with url: URL) {
        ...

        NotificationCenter.default.addObserver(
            self,
            selector: #selector(handleAuthCompletion),
            name: .signInWithOauthCompleted,
            object: nil
        )
    }

    @objc private func handleAuthCompletion(_ notification: Notification) {
        NotificationCenter.default.removeObserver(self, name: .signInWithOauthCompleted, object: nil)

        let token = notification.userInfo?["token"] as? String

        safariViewController?.dismiss(animated: true) { [weak self] in
            self?.safariViewController = nil
            self?.authenticateWithToken(token)
        }

        private func authenticateWithToken(_ token: String?) {
            guard let webViewController = delegate?.destination as? HotwireWebViewController,
                  let webView = webViewController.visitableView.webView else { return }

            if let token = token, let tokenAuthPath = tokenAuthPath {
                guard let tokenLoginUrl = URL(string: "\(baseUrl)\(tokenAuthPath)") else { return }

                var components = URLComponents(url: tokenLoginUrl, resolvingAgainstBaseUrl: false)
                components?.queryItems = [URLQueryItem(name: "token", value: token)]

                if let url = components?.url {
                    webView.load(URLRequest(url: url))
                } else {
                    webView.reload()
                }
            }
        }
    }

    ...

    extension Notification.Name {
        static let signInWithOauthCompleted = Notification.Name("signInWithOauthCompleted")
    }
}

```



```

extension SceneController: UIWindowSceneDelegate {
    func scene(
        _ scene: UIScene, connectionOptions: UIWindowScene.ConnectionOptions
    ) {
        ...

        if let urlContext = connectionOptions.urlContexts.first {
            handleIncomingURL(urlContext.url)
        }

        func scene(_ scene: UIScene, openURLContexts URLContexts: Set<UIOpenURLContext>) {
            guard let url = URLContexts.first?.url else { return }
            handleIncomingURL(url)
        }

        private func handleIncomingURL(_ url: URL) {
            guard let host = url.host else { return }

            switch host {
            case "auth-callback":
                let components = URLComponents(url: url, resolvingAgainstBaseURL: false)
                let token = components?.queryItems?.first(where: { $0.name == "token" })?.value

                NotificationCenter.default.post(
                    name: .signInWithOAuthCompleted,
                    object: nil,
                    userInfo: token != nil ? ["token": token!] : nil
                )
            default:
                ...
            }
        }
    }
}

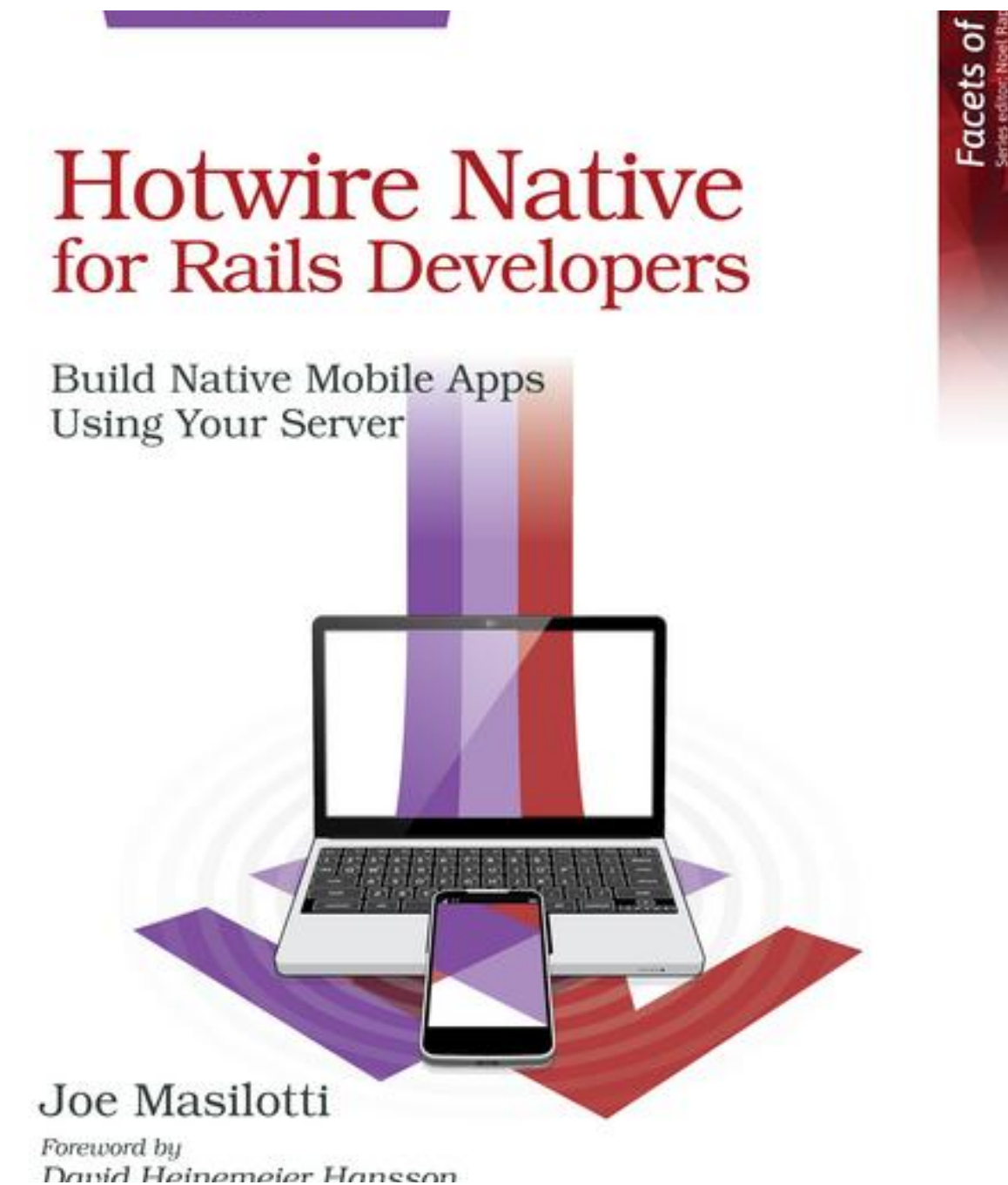
```

Summary

- Screen Navigation
- Path Configuration
- Navigation Bar Title
- Native Tab Bar
- Bridge Components
- OAuth

Resources

- Hotwire Native Handbook by 37signals
- Hotwire Native for Rails Developers book by Joe Masilotti
- Learn Hotwire course by Chris Oliver and William Kennedy



⚡ **Learn Hotwire**

Thanks!

Questions?