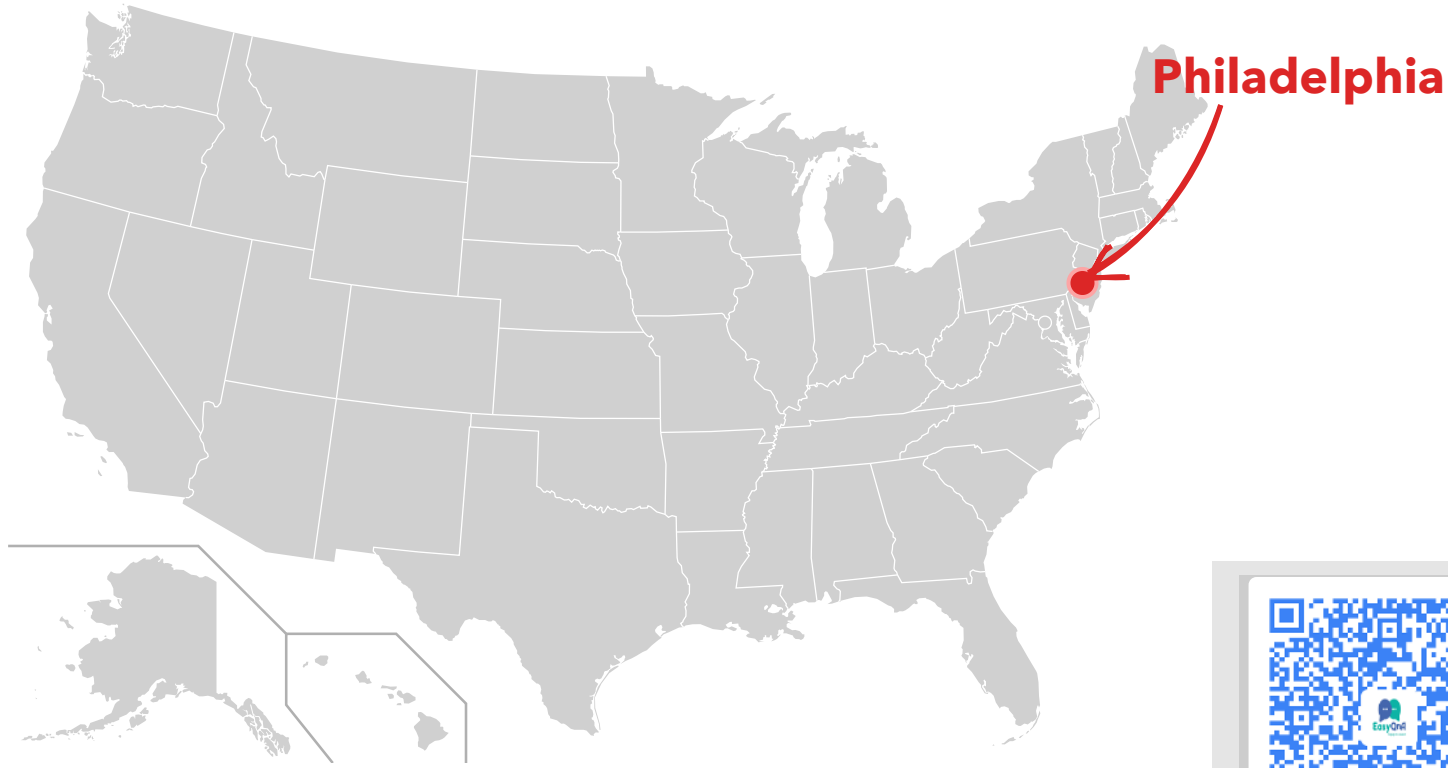
Introduction to Hotwire Native

Build iOS and Android apps with Ruby on Rails

Mike Dalton

Hi, I'm Mike

I’m from Philadelphia




DENNY'S
MUFFLER CENT


JOE NIAGARA
MUSIC UNLIMITED
RECORDS TAPES ACCESSORIES

I work at  Triumph


TIGER

COLLECT WHAT YOU LOVE

Unleash your inner collector.

EXPLORE


TRIUMPH
M696
Uber

Wondering why everything looks different? Learn more

Triumph

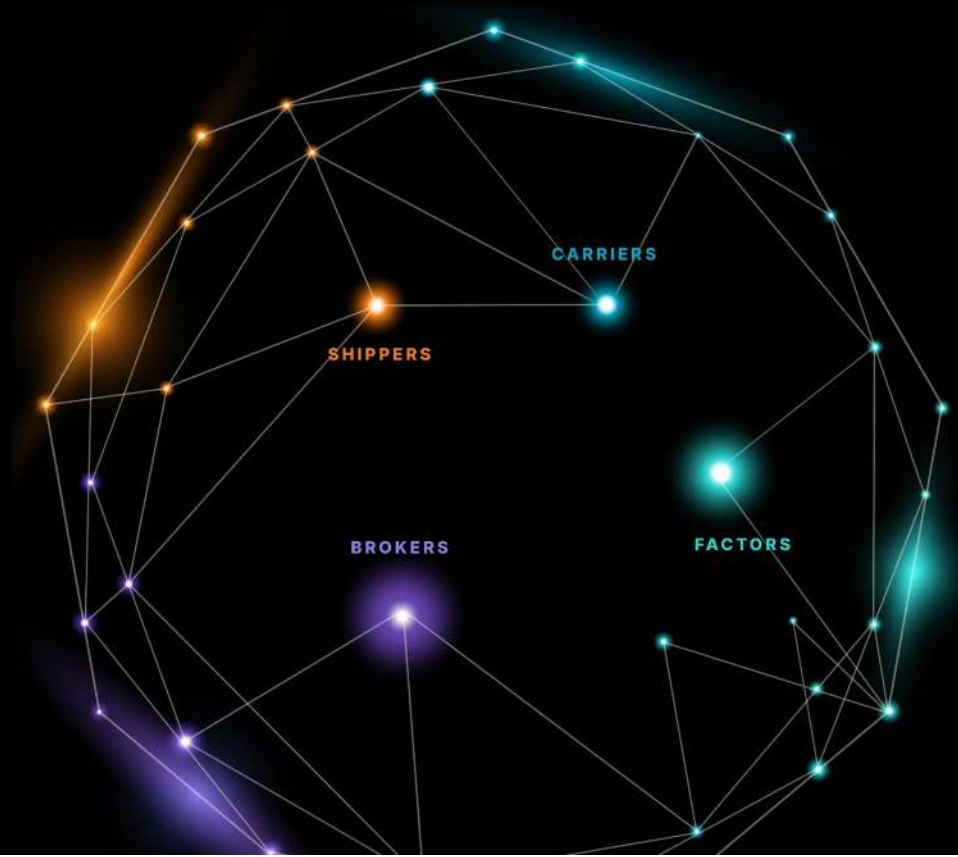
Broker Carrier Shipper Factor Solutions About Us

Log In

TRANSACT CONFIDENTLY

Get ready to experience the Triumph Network — the network modernizing and simplifying freight transactions.

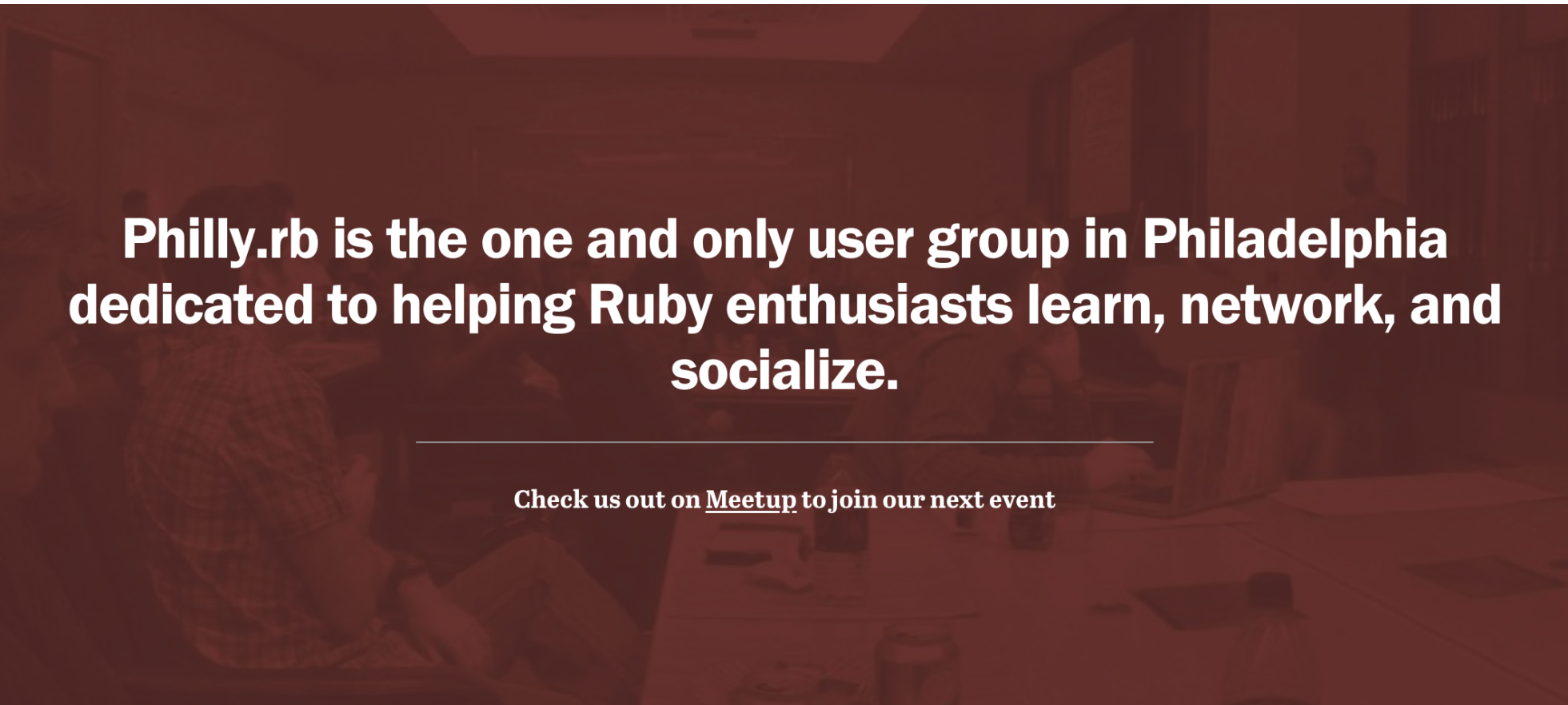
Get Started >



BLUESKY TWITTER

Philly.rb

SUBMIT TALK UPCOMING EVENTS

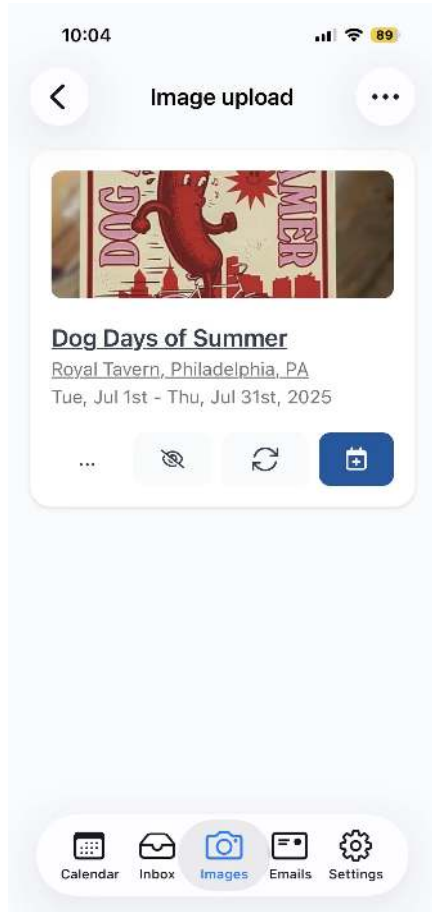


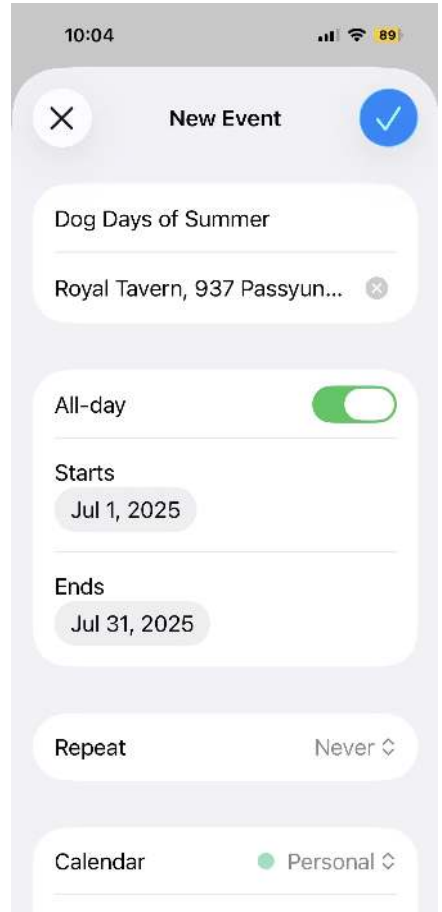
Philly.rb is the one and only user group in Philadelphia dedicated to helping Ruby enthusiasts learn, network, and socialize.

Check us out on Meetup to join our next event

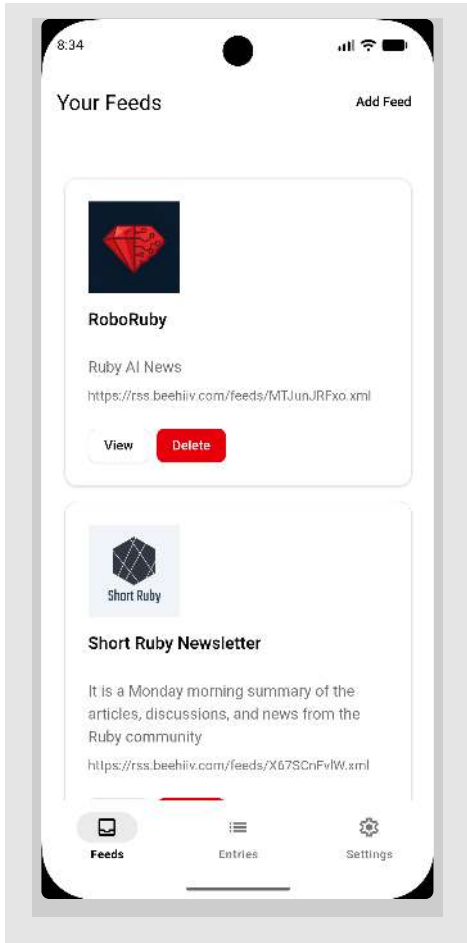
Building Calendar Vision


EV 0.0
ROYAL TAVERN
DOG DAYS OF SUMMER
NEW HOT DOGS EACH WEEK
THE ENTIRE MONTH OF JULY
20 HOT DOGS IN TOTAL
2×
VIDEO
PHOTO


10:04
89
Image upload
Dog Days of Summer
Royal Tavern, Philadelphia, PA
Tue, Jul 1st - Thu, Jul 31st, 2025
Calendar
Inbox
Images
Emails
Settings


10:04
89
New Event
Dog Days of Summer
Royal Tavern, 937 Passyun...
All-day
Starts
Jul 1, 2025
Ends
Jul 31, 2025
Repeat
Never
Calendar
Personal

RSS Reader


8:34
Your Feeds
Add Feed
RoboRuby
Ruby AI News
https://rss.beehiiv.com/feeds/MTJunJRFxo.xml
View
Delete
Short Ruby
Short Ruby Newsletter
It is a Monday morning summary of the articles, discussions, and news from the Ruby community
https://rss.beehiiv.com/feeds/X67SCnFvIW.xml
Feeds
Entries
Settings

Hotwire

Hotwire

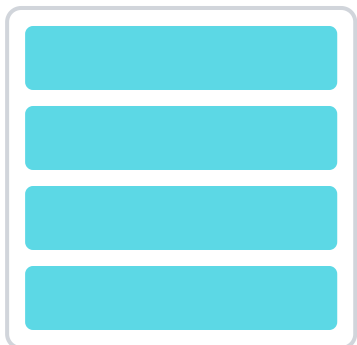
- Created by 37signals
- Default front end "framework" in Rails 7+
- HTML (not JSON) "over the wire"

"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/


TURBO
STIMULUS
NATIVE

Turbo



Turbo Drive

Swaps the whole page

Turbo Frames

Swaps one region

Turbo Streams

Swaps many regions

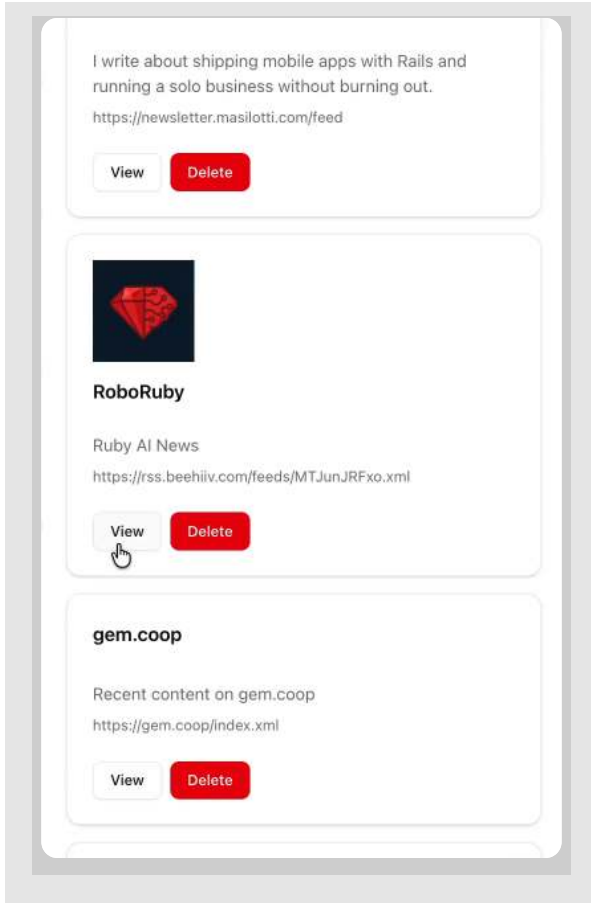
Turbo Drive

```
import "@hotwired/turbo-rails"
```

app/javascript/application.js

```
<%= link_to "View", feed_path(feed) %>
```

app/views/feeds/_card.html.erb



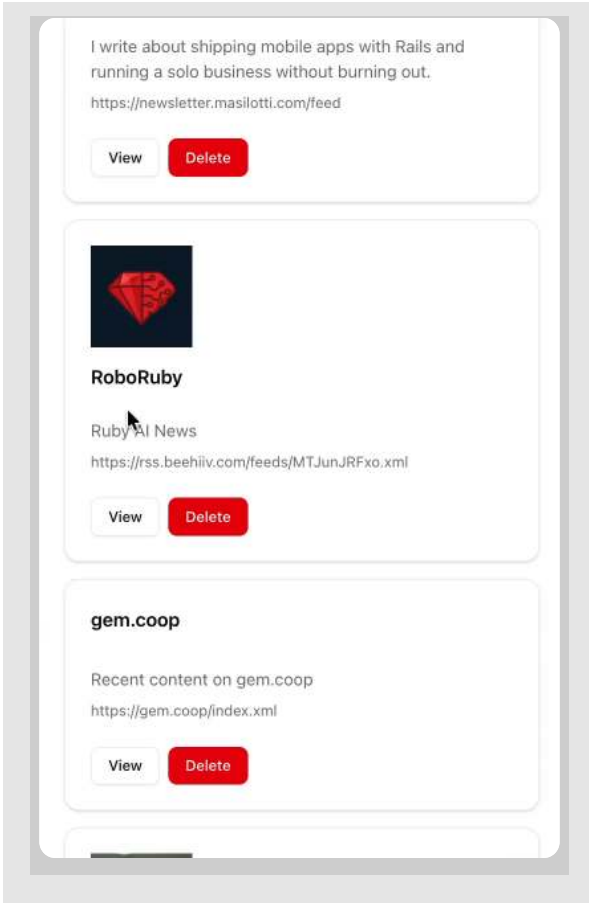
Turbo Frames

```erb
<%= turbo_frame_tag dom_id(feed, :title) do %>
  <h3>
    <%= link_to feed.title, edit_feed_title_path(feed) %>
  </h3>
<% end %>
```

app/views/feeds/_title.html.erb

```erb
<%= turbo_frame_tag dom_id(@feed, :title) do %>
  <%= form_with model: @feed,
        url: feed_title_path(@feed),
        method: :patch do |form| %>
    <%= form.text_field :title %>
    <%= form.submit "Save" %>
  <% end %>
<% end %>
```

app/views/feeds/titles/edit.html.erb



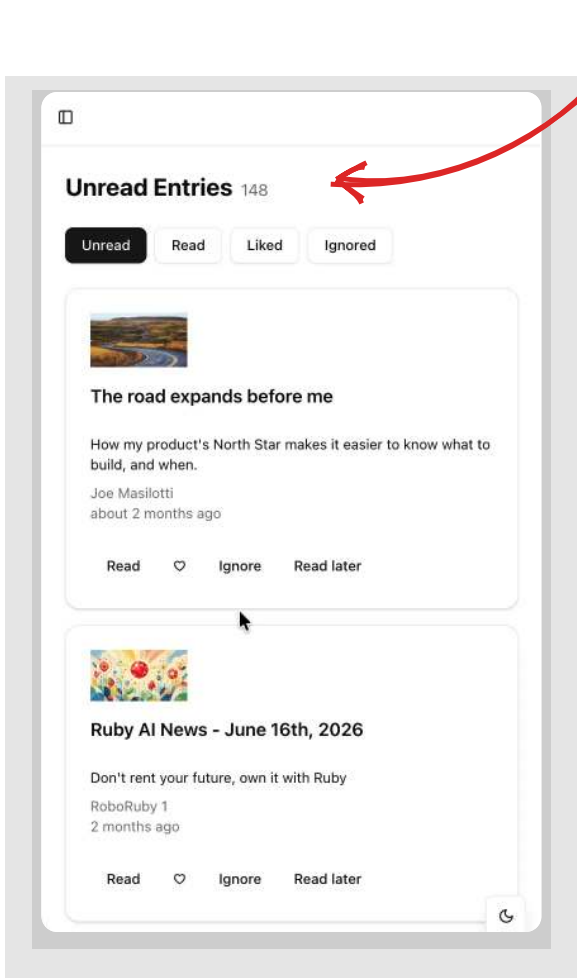
Turbo Streams

```ruby
def toggle_ignore
  @entry = current_user.entries.find(params[:id])
  @entry.update(ignored_at: Time.current)
end
```

app/controllers/entries_controller.rb

```erb
<%= turbo_stream.remove dom_id(@entry, :card) %>

<%= turbo_stream.replace "entries_count" do %>
  <%= render "entries/count", count: @entries_count %>
<% end %>
```

app/views/entries/toggle_ignore.turbo_stream.erb



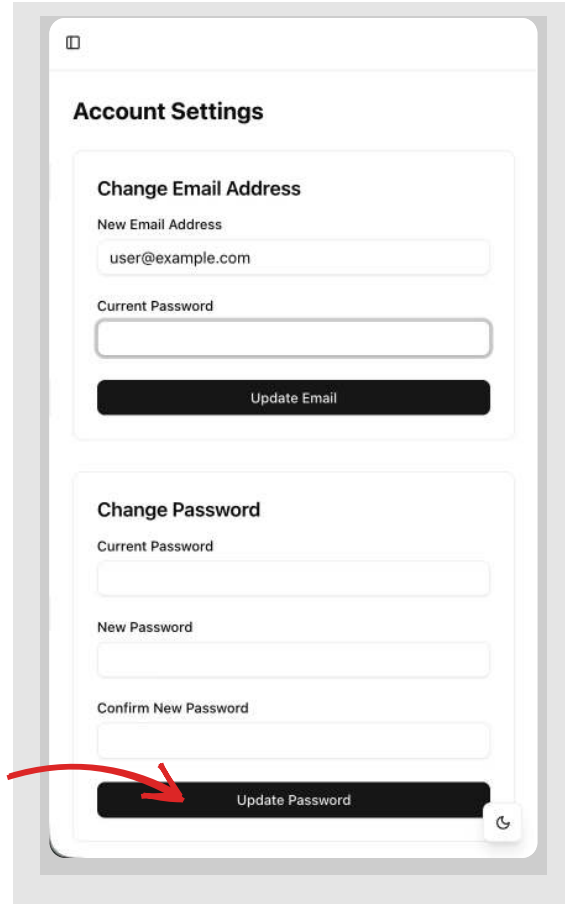
Stimulus

```
import { Controller } from "@hotwired/stimulus"

export default class extends Controller {
  connect() {
    this.timeout = setTimeout(() => {
      this.element.remove()
    }, 5000)
  }

  disconnect() {
    clearTimeout(this.timeout)
  }
}
```

app/javascript/controllers/toast_controller.js

```
<div class="toast" data-controller="toast">
  <h2><%= flash[:notice] %></h2>
</div>
```

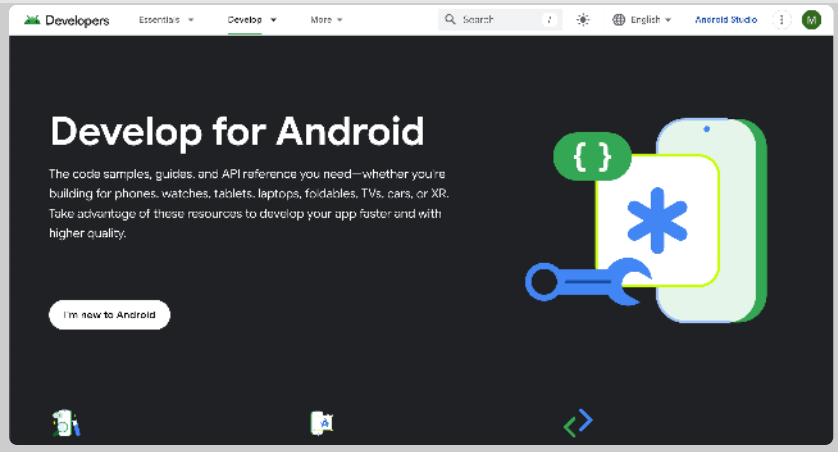
app/views/shared/_flash_toasts.html.erb



Mobile Frameworks

Native Alternatives

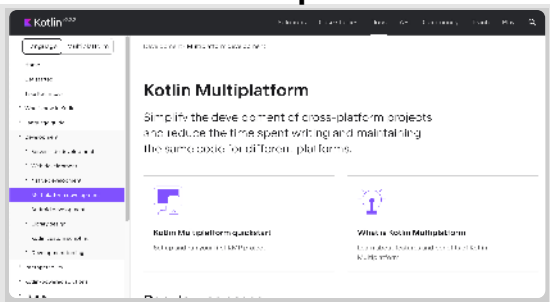
Android (Kotlin)



iOS (Swift)

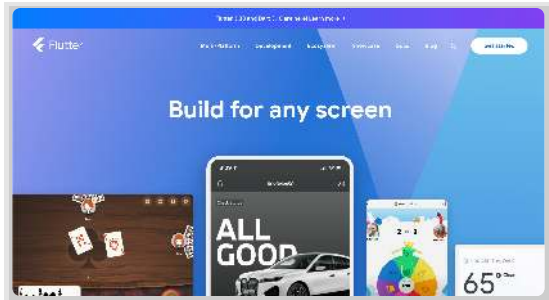
Crossplatform Alternatives

Kotlin Multiplatform



Kotlin

React Native

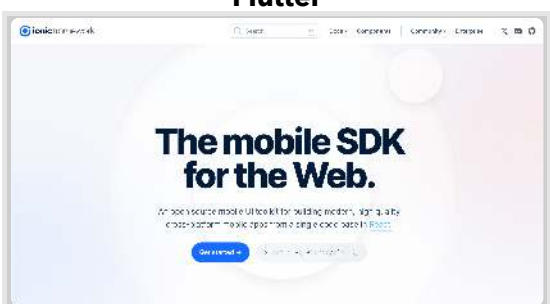


React

.NET MAUI

C#

Flutter



Dart

Ionic

Angular, React or Vue

Swift for Android

Swift

Zero native code.

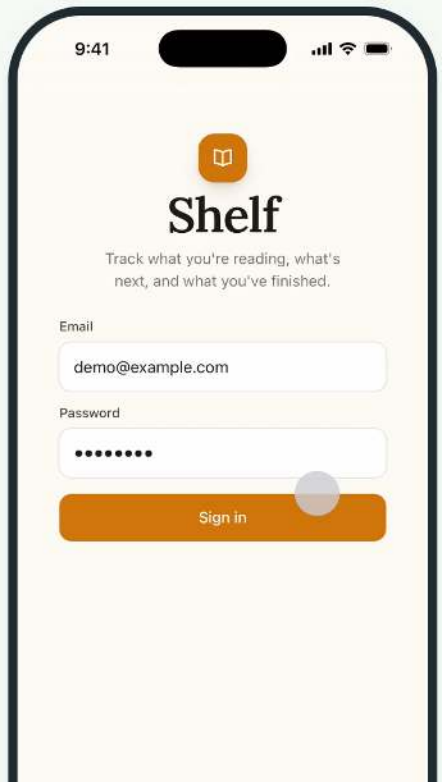
Deploy your Rails app to the app stores.

From `bundle install` to your phone in minutes.

To the App Store and Google Play without a line of native code.

Preview on your phone | Start for free

pingrb Foxance Beervana ? Your app?

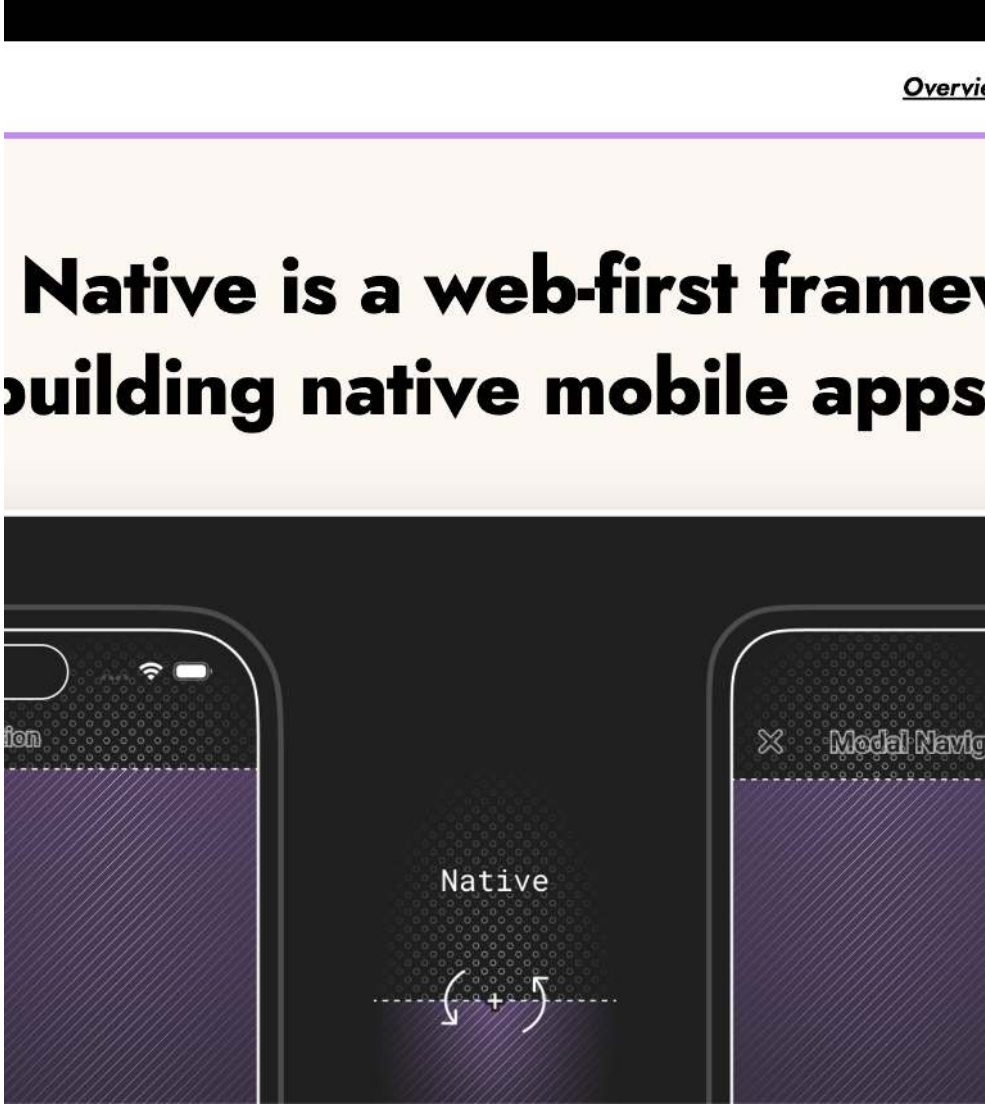


Hotwire Native is a web-first framework for building native mobile apps.

Hotwire Native

History

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



Who should use it

You need a web, iOS and Android app

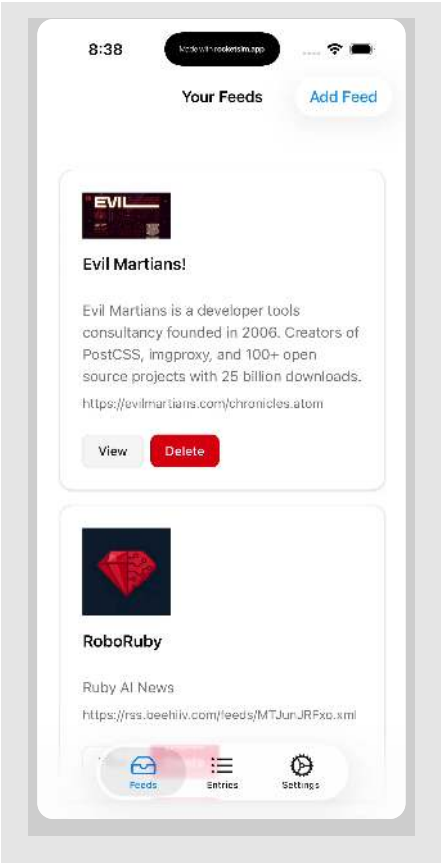
You want to build with Hotwire and Turbo

You’re comfortable being an early adopter

How does it work

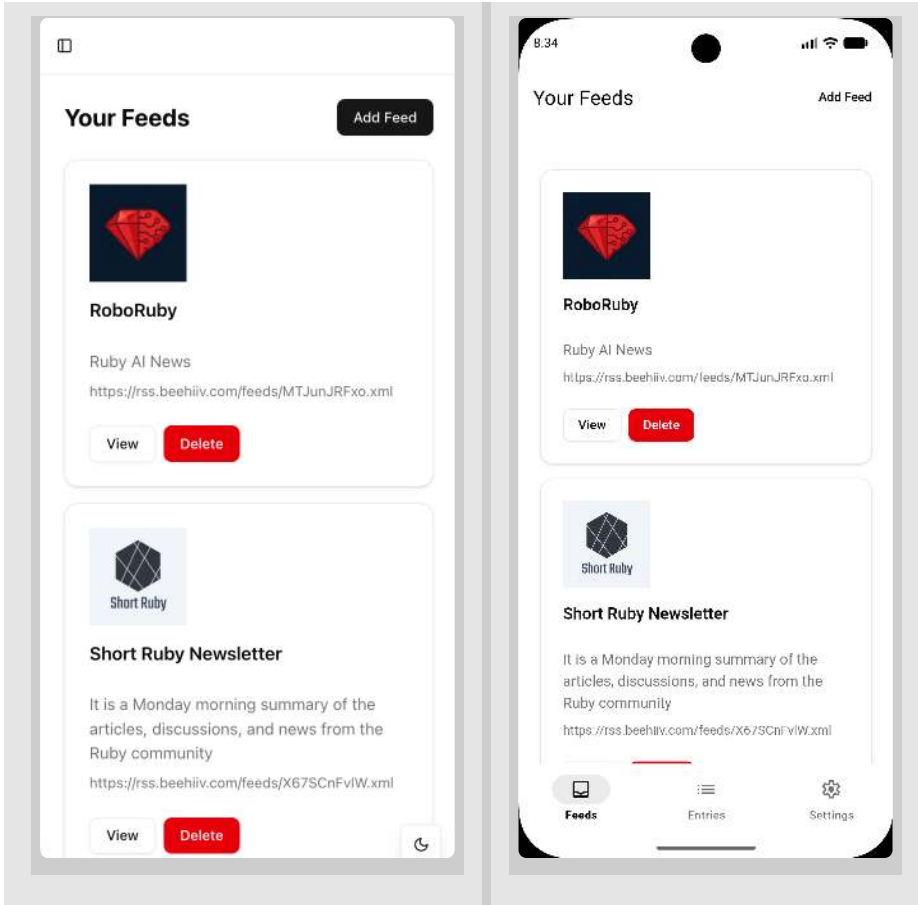
How does it work

Native navigation and animation



How does it work

Embedded web browser



Project Setup

Start with a Hotwire Rails app

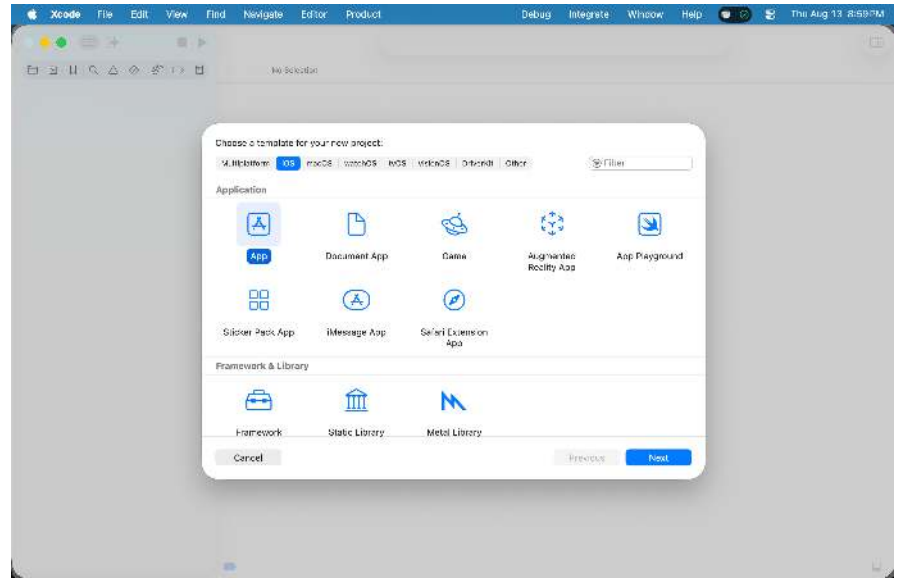
No generator like `rails new` for the mobile app

Hotwire Native apps are Native apps

Project Setup

iOS

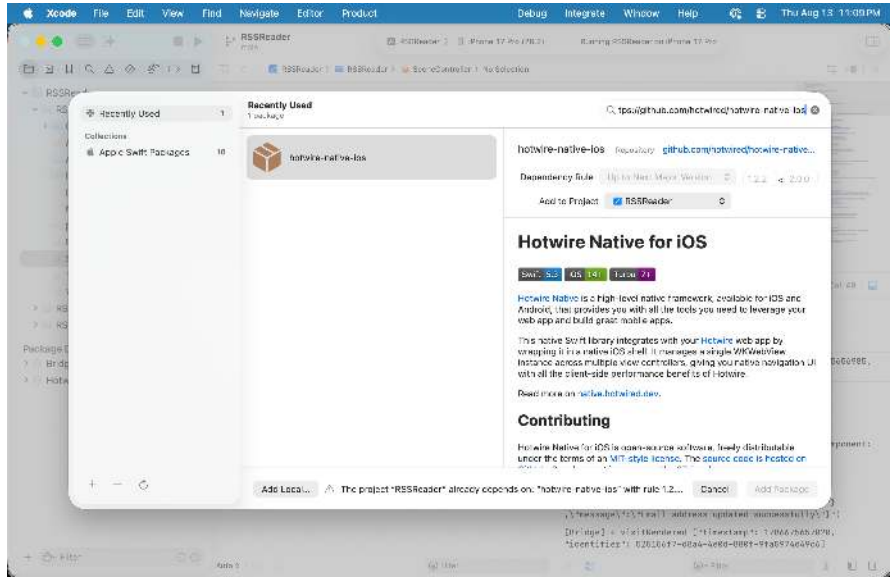
- Use the Xcode New Project Wizard



Project Setup

iOS

- Add `hotwire-native-ios` package dependency



Project Setup

iOS

- Replace `SceneDelegate.swift`

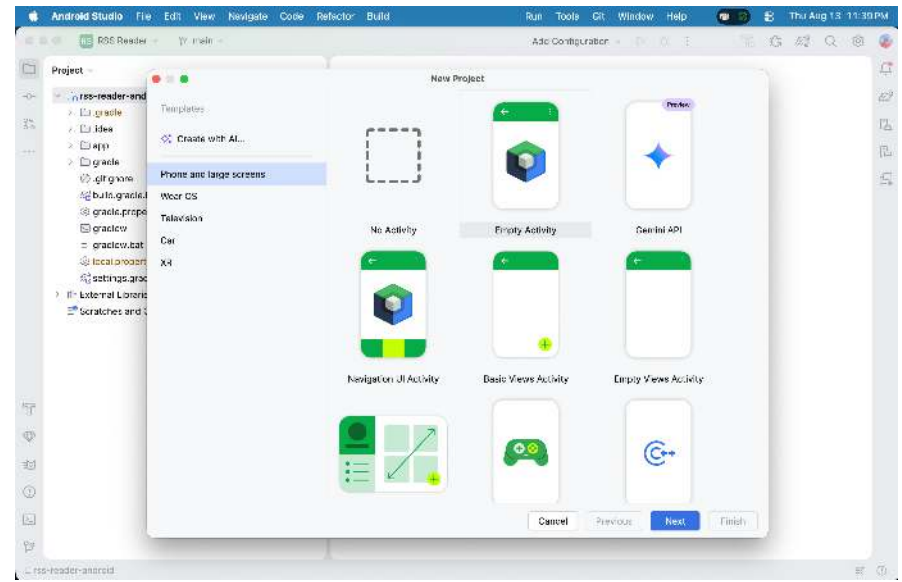
```swift
import HotwireNative
import UIKit

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

- Use the Android Studio New Project Wizard



Project Setup

Android

- Add `hotwire-native-android` dependency

```
dependencies {
    implementation("dev.hotwire:core:${libs.versions.hotwireNative}")
    implementation(
        "dev.hotwire:navigation-fragments:${libs.versions.hotwireNative}"
    )
}
```

app/build.gradle.kts

Project Setup

Android

- Enable internet access

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

app/src/main/AndroidManifest.xml

Project Setup

Android

- Replace `MainActivity.kt`

```kotlin
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

Project Setup

Android

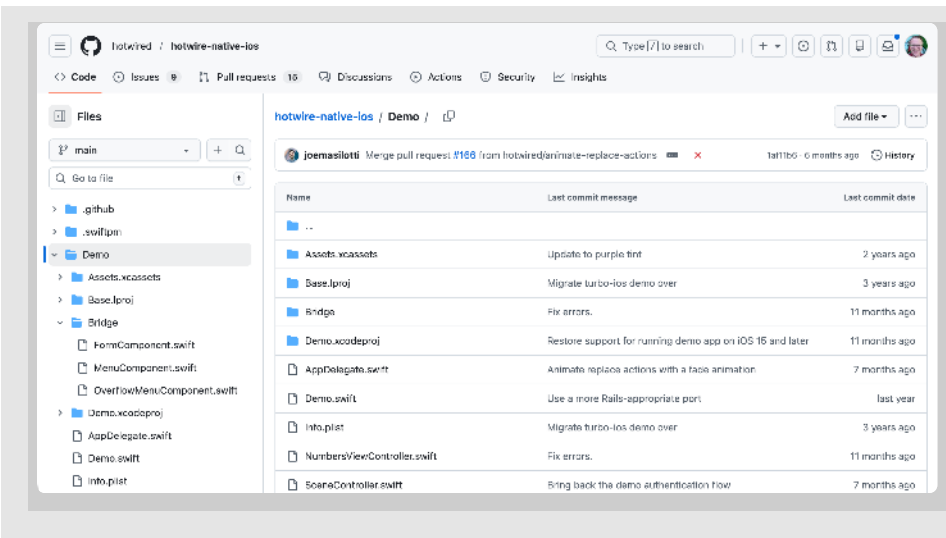
- Replace `activity_main.xml`

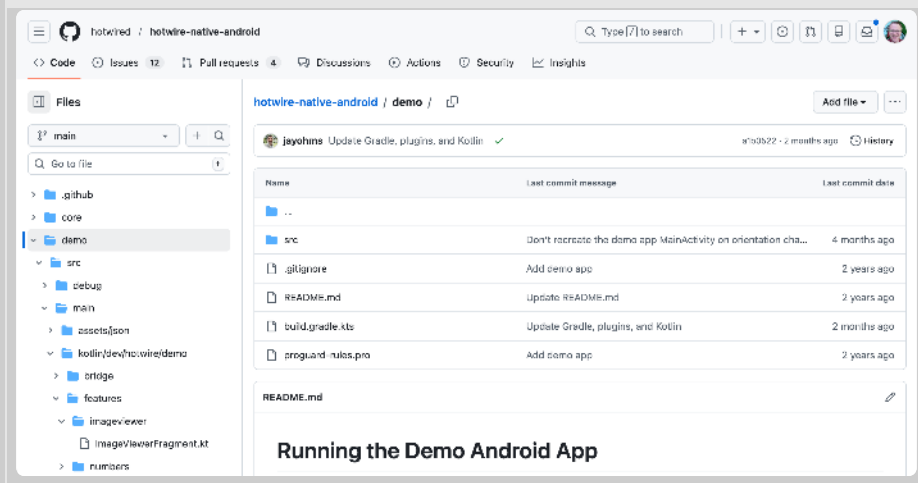
```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.fragment.app.FragmentContainerView
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/main_nav_host"
    android:name="dev.hotwire.navigation.navigator.NavigatorHost"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

activity_main.xml

Project Setup

Alternative





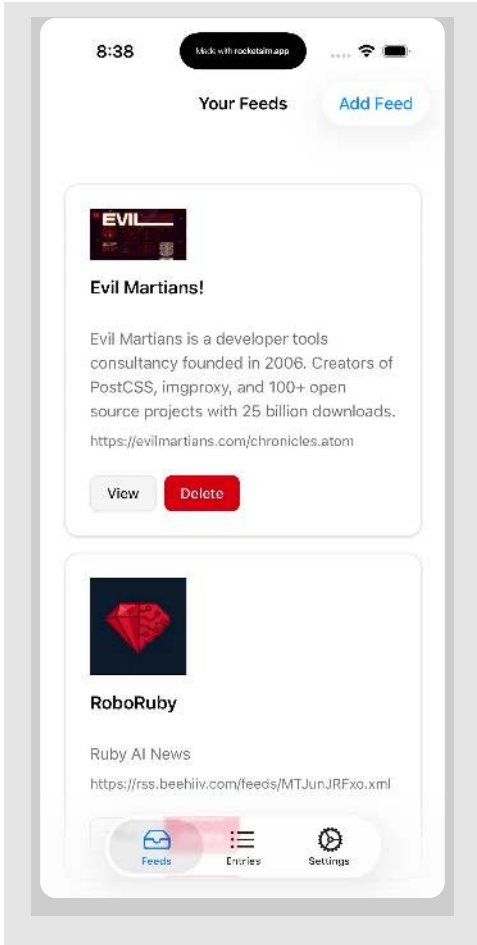
Screen Navigation

Screen Navigation

Push and Pop

```
<%= link_to "View", feed_path(feed) %>
```

app/views/feeds/_card.html.erb

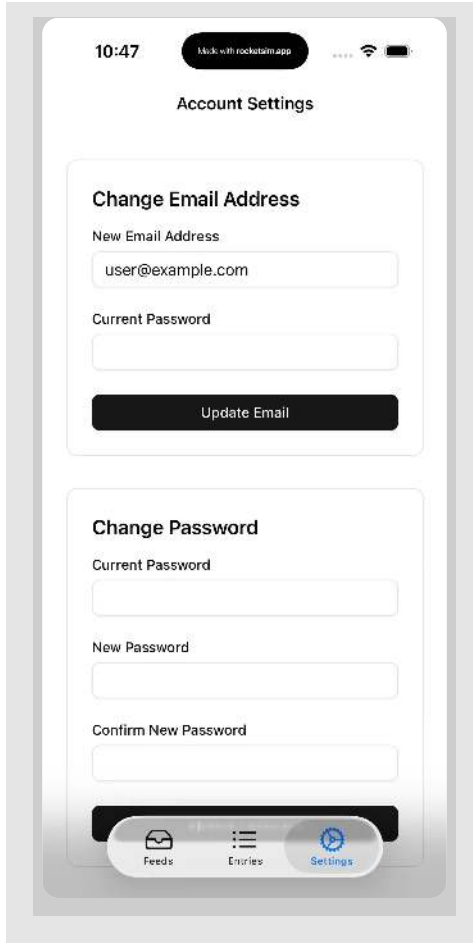


Screen Navigation

Replace

```erb
<%= form_with model: @user do |form| %>
  <div class="grid gap-2">
    <%= form.label :email_address, "New Email Address" %>
    <%= form.email_field :email_address %>
  </div>

  <div class="grid gap-2">
    <%= form.label :current_password %>
    <%= form.password_field :current_password %>
  </div>

  <%= form.submit "Update Email" %>
<% end %>
```

app/views/users/edit.html.erb

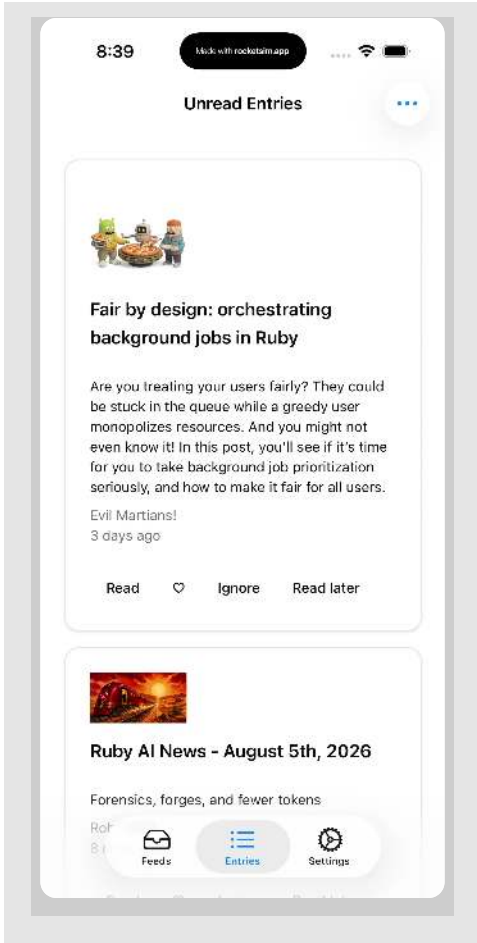


Screen Navigation

External Links

```
<%= link_to entry.title,
    entry.url,
    target: "_blank",
    rel: "noopener noreferrer" %>
```

app/views/entries/_card.html.erb

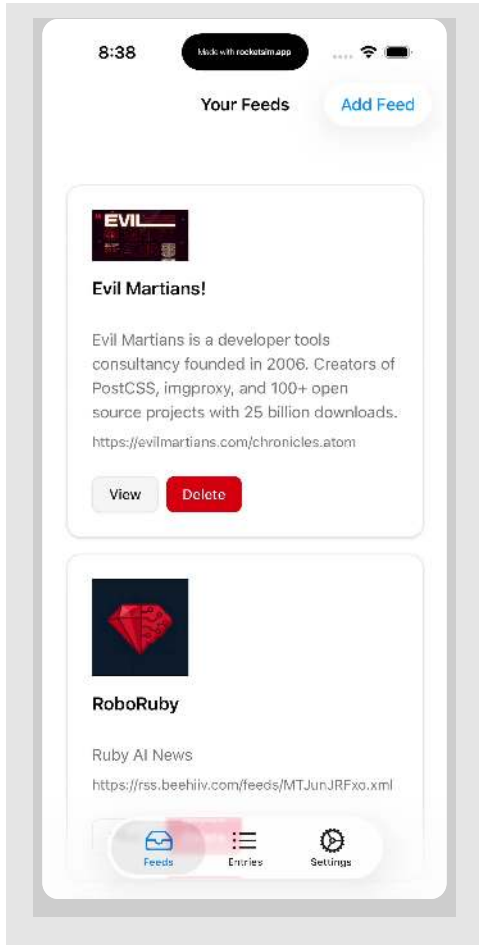


Path Configuration

Path Configuration

Basic Navigation

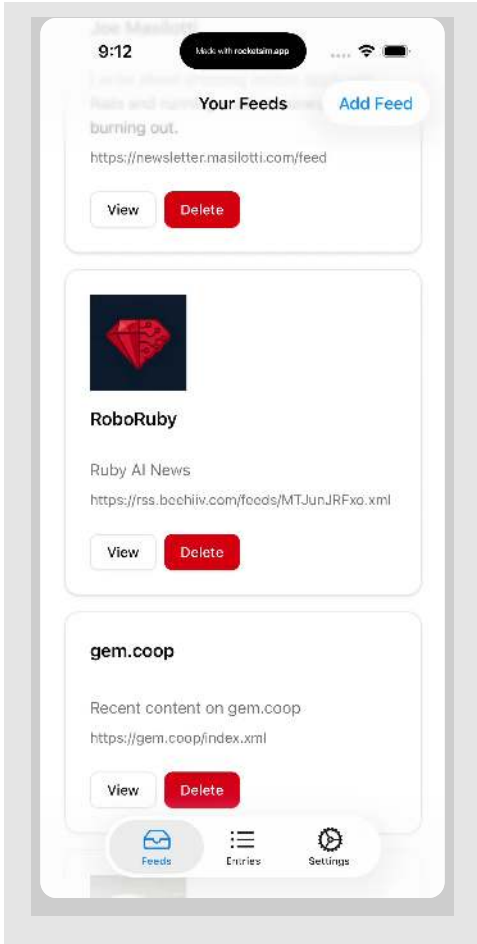
```
{
  "rules": [
    {
      "patterns": [
        ".*"
      ],
      "properties": {
        "context": "default",
        "pull_to_refresh_enabled": true
      }
    }
  ]
}
```



Path Configuration

Modal Navigation

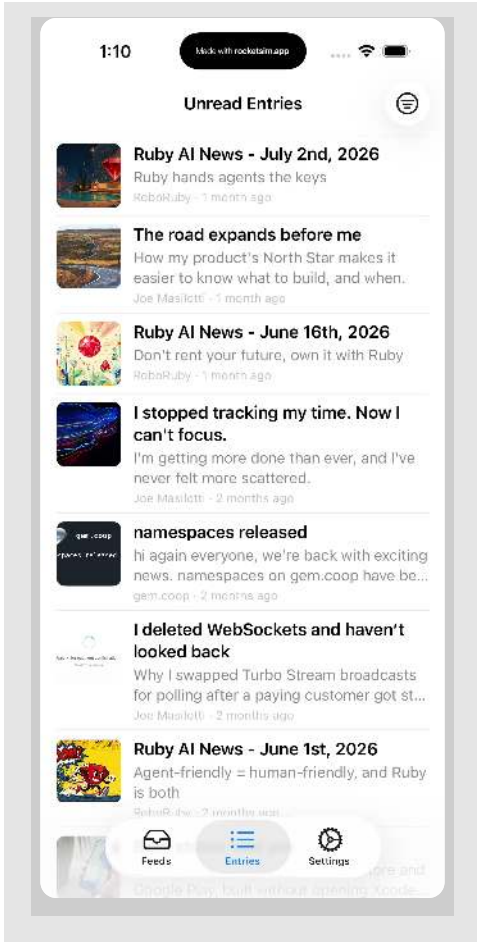
```
{
  "rules": [
    {
      "patterns": [
        "/new$",
        "/edit$"
      ],
      "properties": {
        "context": "modal",
        "pull_to_refresh_enabled": false
      }
    }
  ]
}
```



Path Configuration

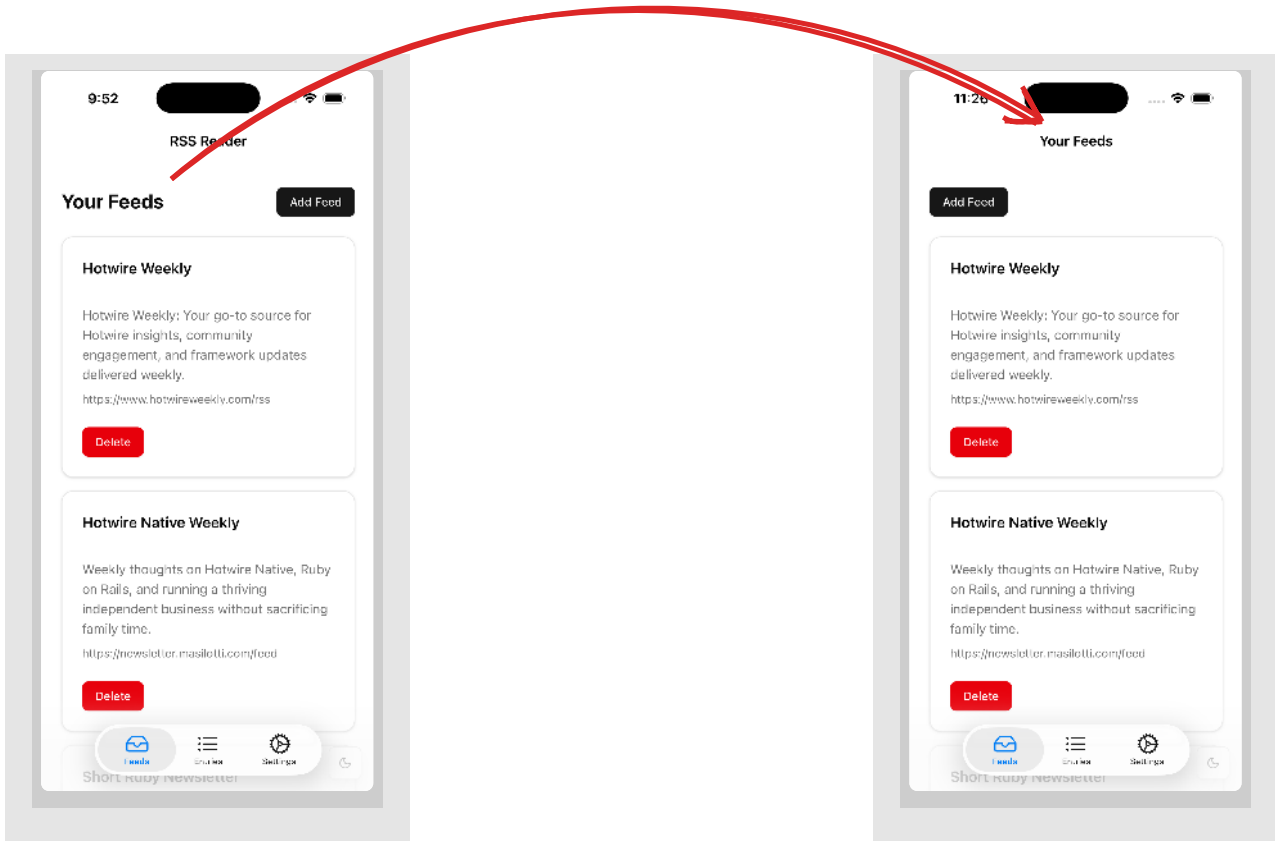
Native Screens

```
{
  rules: [
    {
      patterns: [
        "#{entries_path}$"
      ],
      properties: {
        view_controller: "entries",
        presentation: "replace_root",
        animated: false,
        pull_to_refresh_enabled: false
      }
    }
  ]
}
```



Navigation Bar Title

Navigation Bar Title

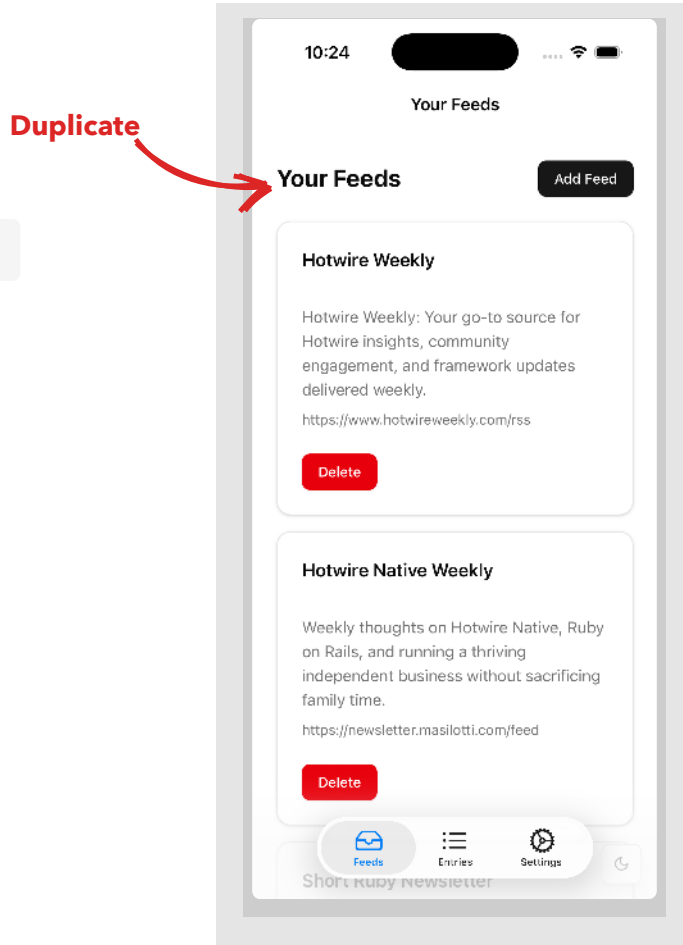


Navigation Bar Title

Add title tag

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

feeds/index.html.erb



Navigation Bar Title

Hide h1 on native

```erb
<%= tag.html(
  data: { hotwire_native: hotwire_native_app? },
) do %>
  ...
<% end %>
```

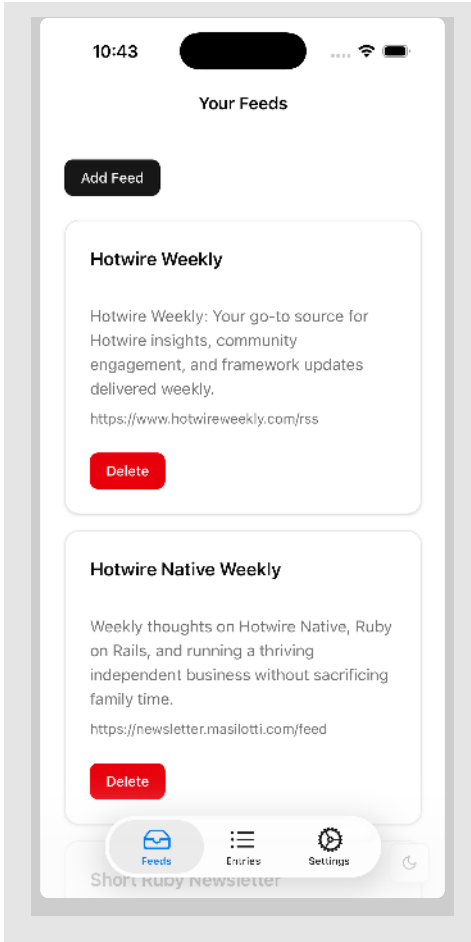
application.html.erb

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

application.css

```erb
<h1 class="hotwire-native:hidden text-2xl font-bold">
  Your Feeds
</h1>
```

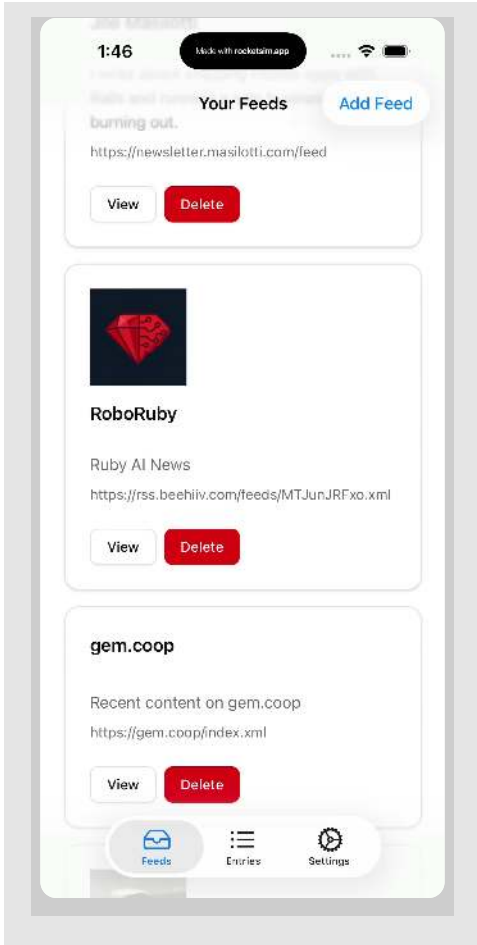
feeds/index.html.erb



Native Tab Bar

Native Tab Bar

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



Native Tab Bar

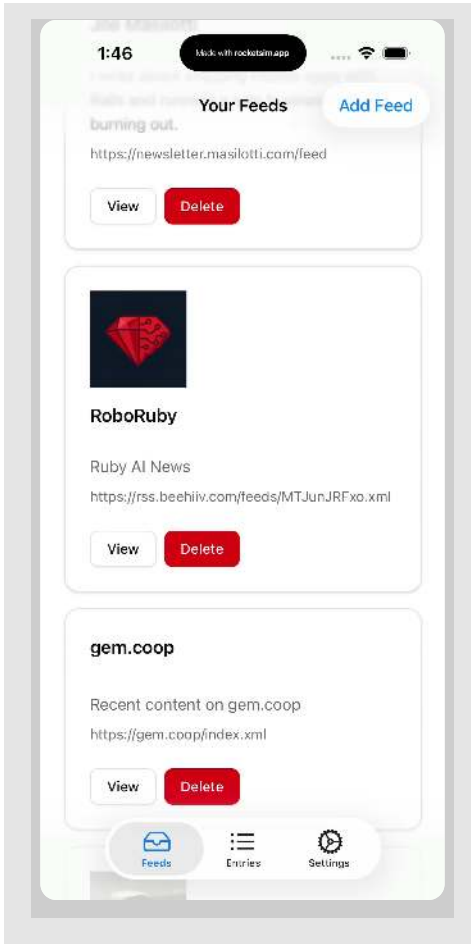
iOS

```
extension HotwireTab {
    static var all: [HotwireTab] {
        [
          HotwireTab(
              title: "Feeds",
              image: .init(systemName: "tray")!,
              url: baseUrl.appending(path: "/feeds")
          ),
          ...
        ]
    }
}
```

Tabs.swift

```
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

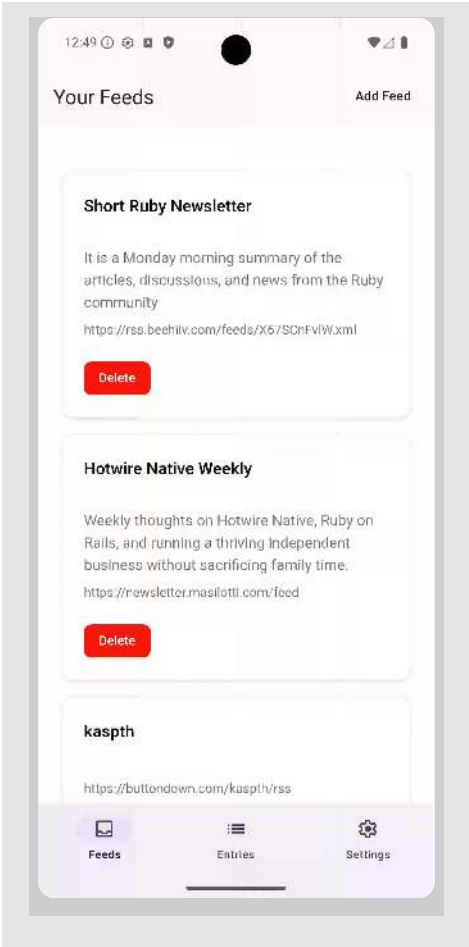


Native Tab Bar

Android

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

Tabs.kt

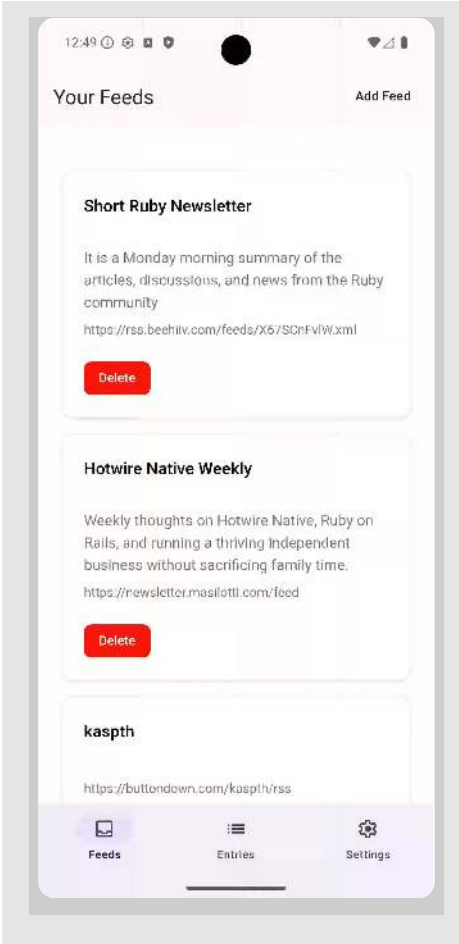


Native Tab Bar

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

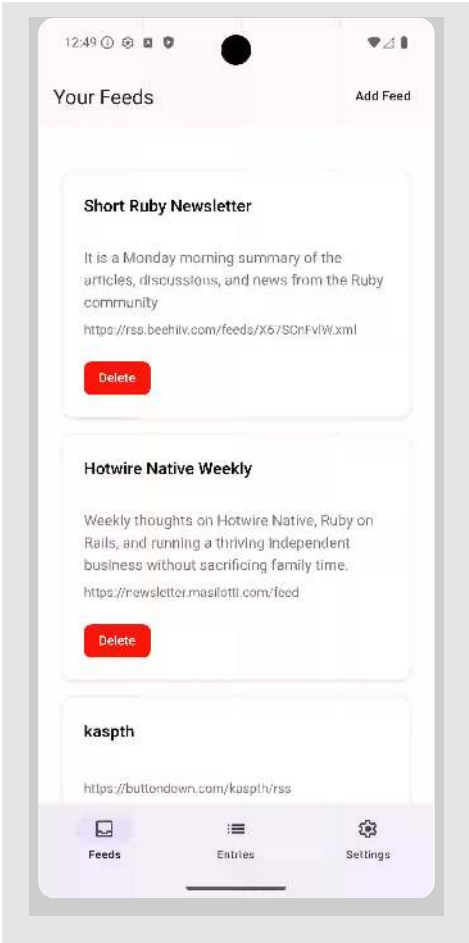


Native Tab Bar

Android

```
class MainActivity : HotwireActivity() {
    private lateinit var
      bottomNavigationController: HotwireBottomNavigationController

    override fun onCreate(savedInstanceState: Bundle?) {
        initializeBottomTabs()
    }

    private fun initializeBottomTabs() {
        val bottomNavigationView = findViewById<BottomNavigationView>(
            R.id.bottom_nav
        )
        bottomNavigationController = HotwireBottomNavigationController(
            this, bottomNavigationView
        )
        bottomNavigationController.load(mainTabs, 0)
    }
}
```

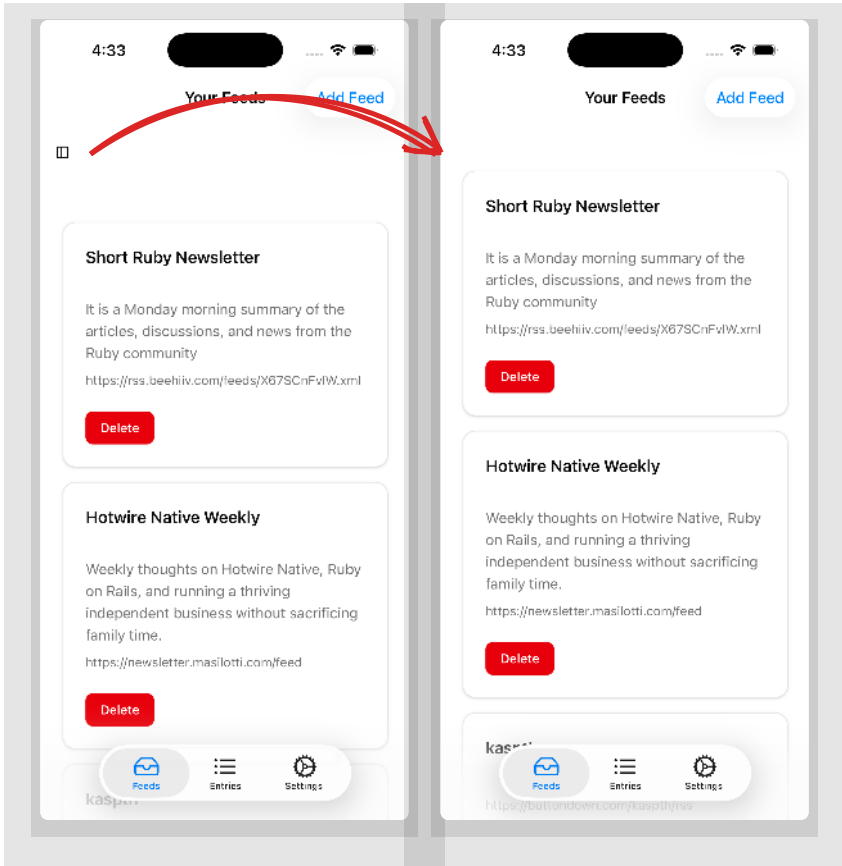
MainActivity.kt



Native Tab Bar

Hide Web Navigation

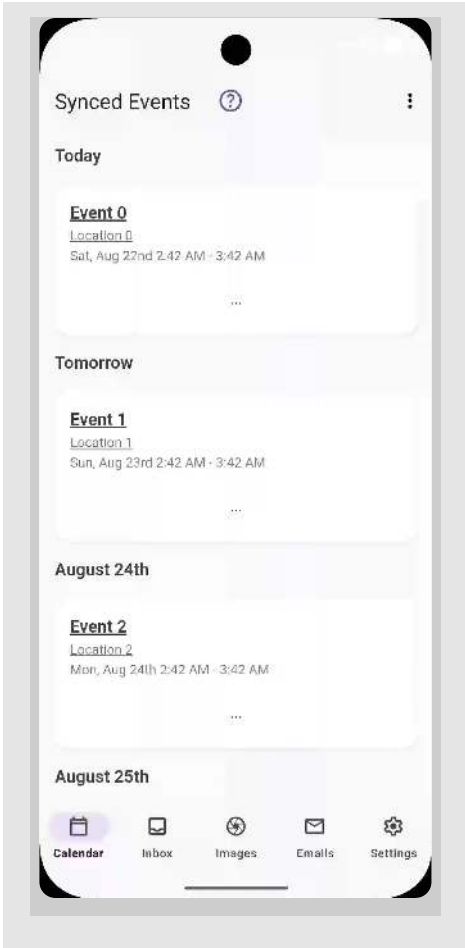
```
<body>
  <main>
    <% if authenticated? %>
      <header class="hotwire-native:hidden flex">
        <button ...>
        </button>
      </header>
    <% end %>

    <%= yield %>
  </main>
</body>
```



Calendar Switcher

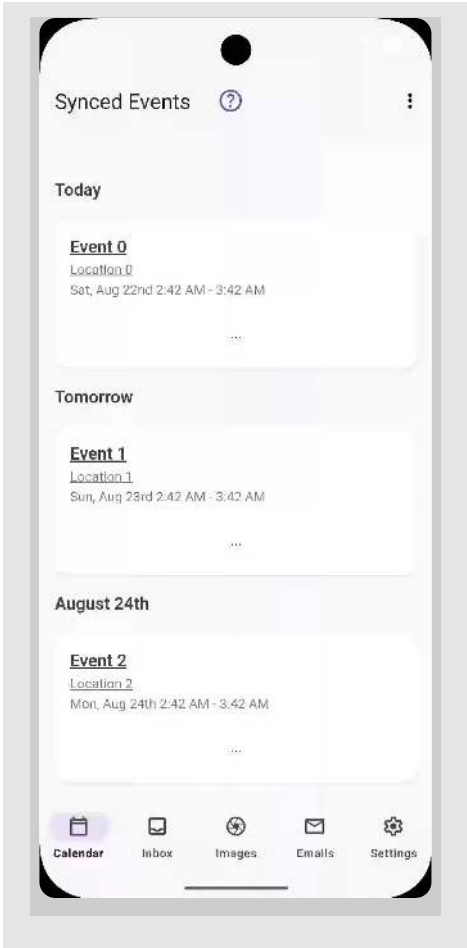
Calendar Switcher

HTML Dialog



Calendar Switcher

Native Modal



Calendar Switcher

Web Implementation

```erb
<%= link_to calendars_path, data: { turbo_stream: true } do %>
  <%= render "heroicons/arrows_up_down" %>
<% end %>
```

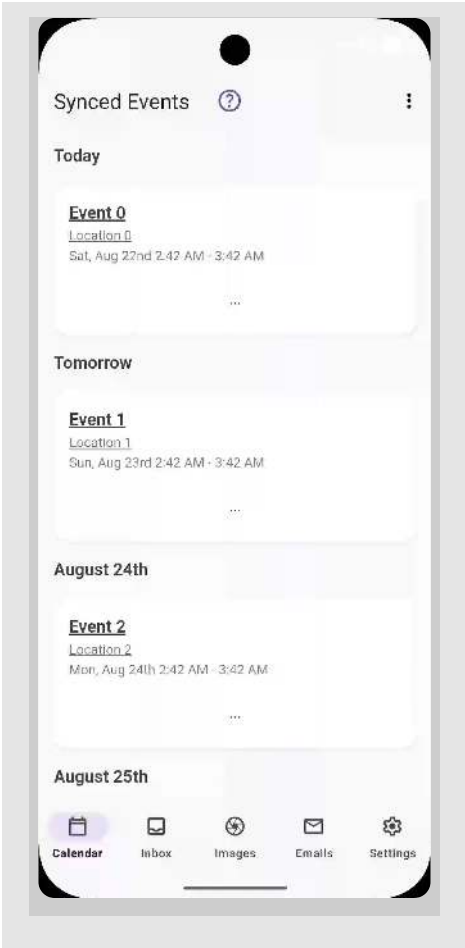
app/views/calendars/show.html.erb

```ruby
class CalendarsController < ApplicationController
  def index
    @calendars = current_user.calendars
  end
end
```

app/controllers/calendars_controller.rb

```erb
<%= turbo_stream.replace "dialog-container" do %>
  <div id="dialog-container">
    <dialog class="modal" open>
      <%= render "index", calendars: @calendars %>
    </dialog>
  </div>
<% end %>
```

app/views/calendars/index.turbo_stream.erb



Calendar Switcher

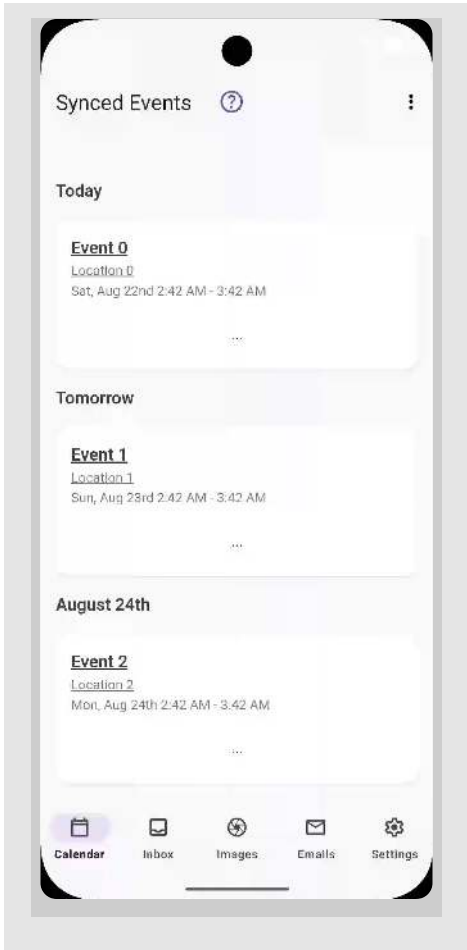
Native Implementation

```
<%= link_to calendars_path,
            data: {
              turbo_stream: !hotwire_native_app?
            } do %>
  <%= render "heroicons/arrows_up_down" %>
<% end %>
```

app/views/calendars/show.html.erb

```
<%= render "index", calendars: @calendars %>
```

app/views/calendars/index.html.erb

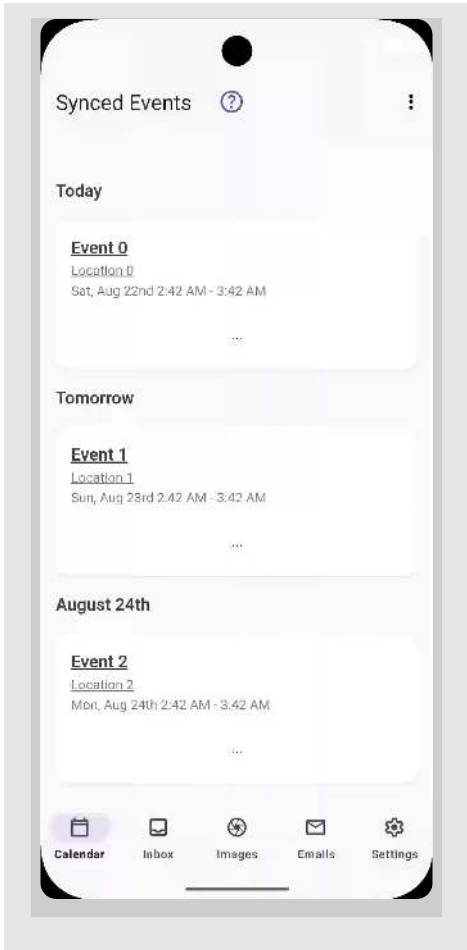


Calendar Switcher

Native Implementation

```ruby
class HotwireNative::ConfigurationsController < ApplicationController
  def android_v1
    render json: {
      rules: [
        {
          patterns: [
            "#{calendars_path}$"
          ],
          properties: {
            context: "modal",
            pull_to_refresh_enabled: false
          }
        }
      ]
    }
  end
end
```

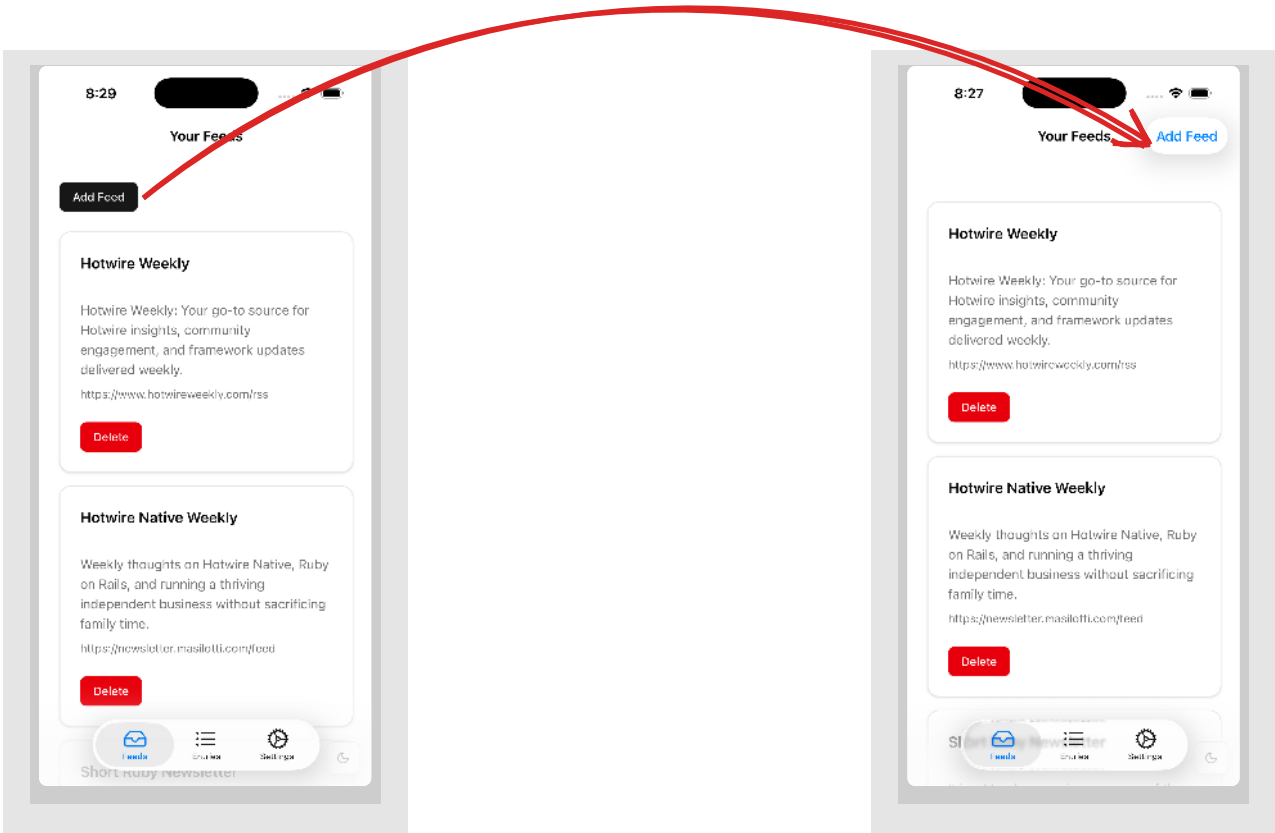
app/controllers/hotwire_native/configurations_controller.rb



Navigation Bar Button

Navigation Bar Button

Goal



Navigation Bar Button

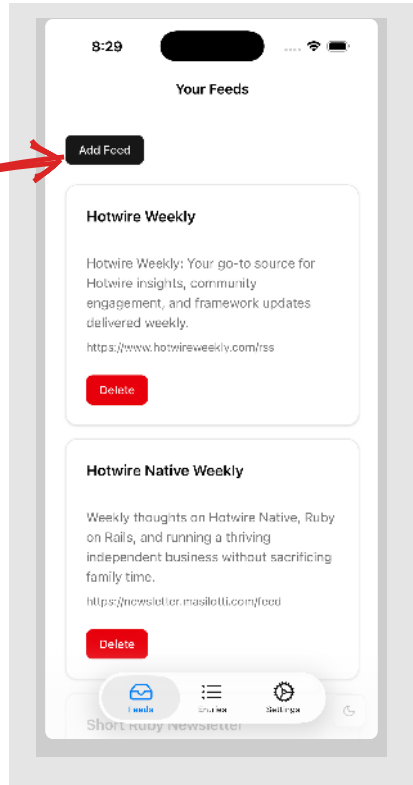
Web

```
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

```
<%= 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



Navigation Bar Button

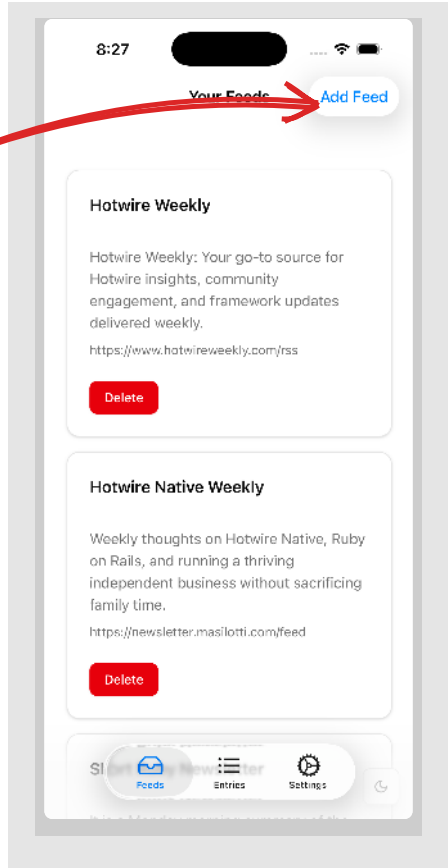
iOS

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

    override func onReceive(message: Message) {
        guard let data: MessageData = message.data(),
              let viewController = delegate?.destination as? UIViewController
        else { return }

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

    struct MessageData: Decodable {
        let title: String
    }
}
```

ButtonComponent.swift

```swift
Hotwire.registerBridgeComponents([
    ButtonComponent.self
])
```

AppDelegate.swift



Navigation Bar Button

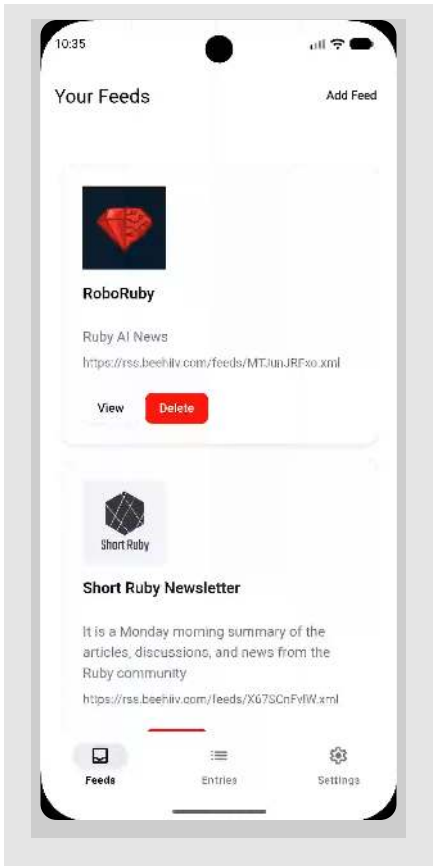
Android

```
class ButtonComponent(
    name: String,
    private val delegate: BridgeDelegate<HotwireDestination>
) : BridgeComponent<HotwireDestination>(name, delegate) {

    override fun onReceive(message: Message) {
        when (message.event) {
            "connect" -> handleConnectEvent(message)
            else -> Log.w("ButtonComponent", "Unknown event")
        }
    }

    private fun handleConnectEvent(message: Message) {
        val data = message.data<MessageData>() ?: return
        // Add a toolbar button titled data.title that
        // calls replyTo("connect") when tapped.
    }

    @Serializable
    data class MessageData(@SerialName("title") val title: String)
}
```

ButtonComponent.kt

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

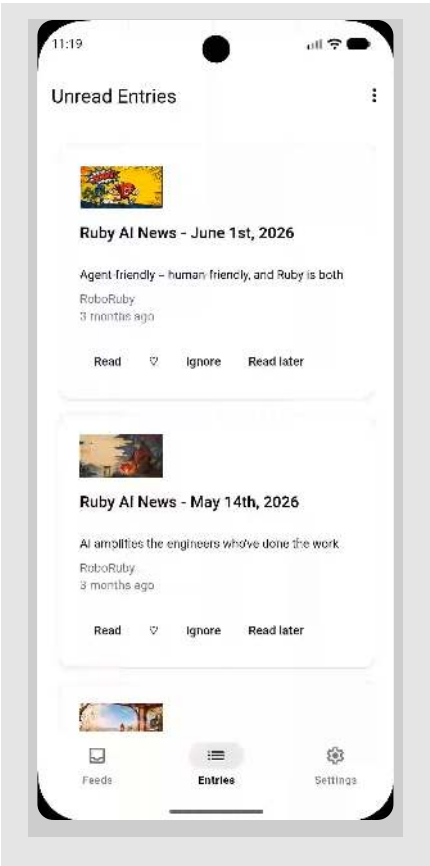
MainApplication.kt



Bridge Components

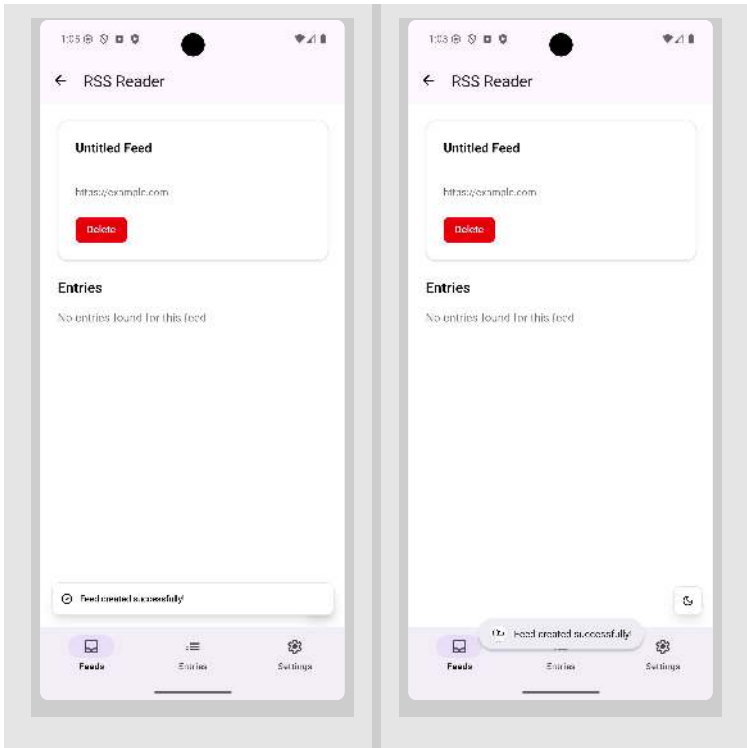
Bridge Components

Navigation Bar Menu



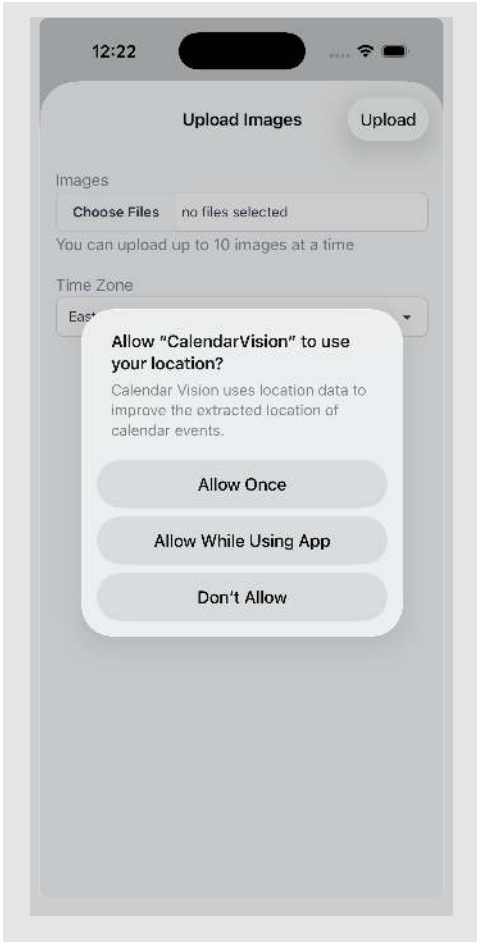
Bridge Components

Toast Messages



Bridge Components

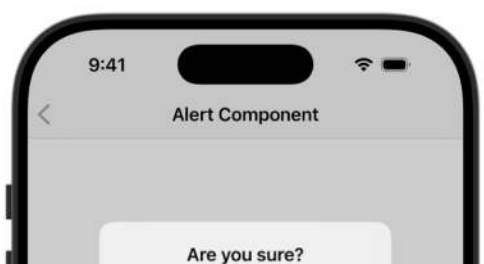
Request Permissions



Bridge Components

Native integrations for Hotwire Native apps

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



Native Components

Add to Calendar

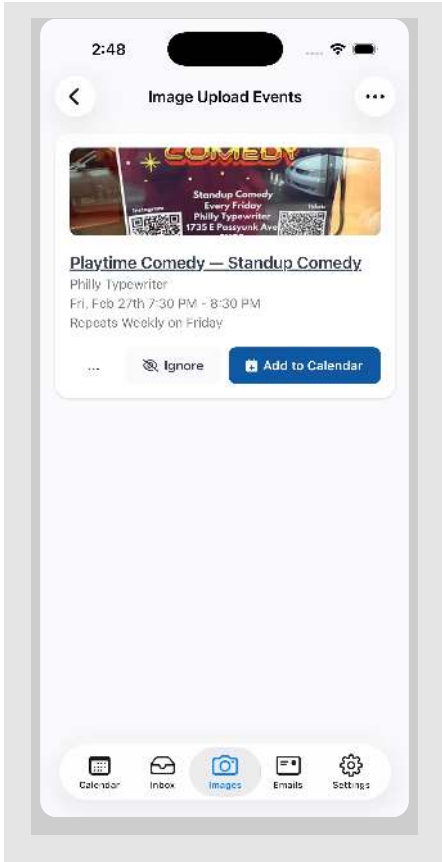
Rails

```
export default class extends BridgeComponent {
  static component = "add-to-calendar"
  static values = {
    eventPayload: Object,
  }

  add() {
    this.send("add", this.eventPayloadValue)
  }
}
```

app/javascript/controllers/bridge/add_to_calendar_controller.js

```
<% payload = calendar_event_json_payload(calendar_event) %>
<%= tag.div data: {
  controller: "bridge--add-to-calendar",
  bridge__add_to_calendar_event_payload_value: payload.to_json
} do %>
  <%= button_tag data: {
    action: "bridge--add-to-calendar#add:prevent",
  } do %>
    <i class="fa-regular fa-calendar-plus fa-lg"></i>
  <% end %>
<% end %>
```

app/views/calendar_events/_add_to_calendar.html.erb

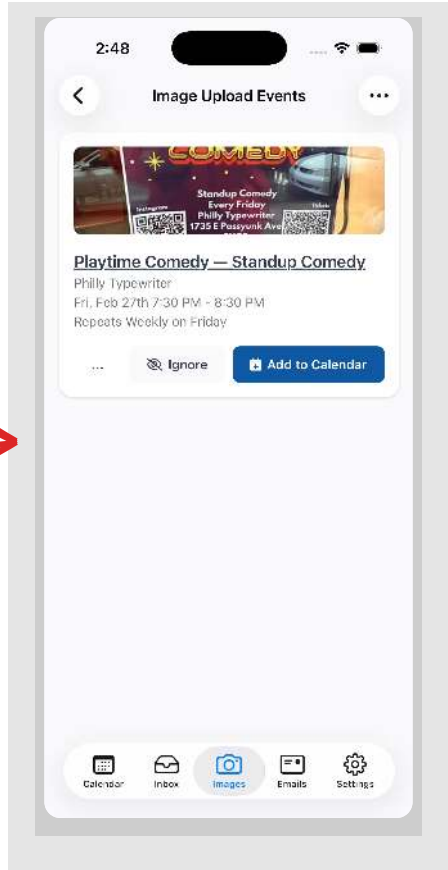


Add to Calendar

iOS

```
final class AddToCalendarComponent: BridgeComponent {
    override func onReceive(message: Message) {
        guard message.event == "add",
              let calendarEvent: CalendarEvent = message.data()
        else { return }

        Task {
            await presentEventEditor(calendarEvent: calendarEvent)
        }
    }

    private func presentEventEditor(calendarEvent: CalendarEvent) async {
        let event = EKEventBuilder.makeEvent(
            from: calendarEvent,
            eventStore: CalendarEventStore.shared
        )

        await MainActor.run {
            let editDelegate = EventEditDelegate()
            let eventViewController = EKEventEditViewController()
            eventViewController.event = event
            eventViewController.eventStore = CalendarEventStore.shared
            eventViewController.editViewDelegate = editDelegate
            viewController?.present(eventViewController, animated: true)
        }
    }
}
```

AddToCalendarComponent.swift

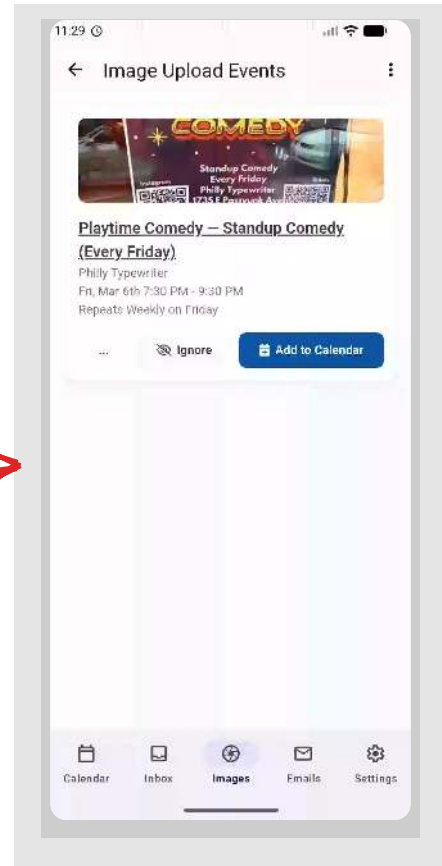


Add to Calendar

Android

```
class AddToCalendarComponent(
    ...
) : BridgeComponent<HotwireDestination>(name, bridgeDelegate) {
    override fun onReceive(message: Message) {
        when (message.event) {
            "add" → handleAddToCalendar(message)
        }
    }

    private fun handleAddToCalendar(message: Message) {
        val calendarEvent = message.data<CalendarEvent>()
        val activity = bridgeDelegate.destination
            .fragment.requireActivity()
        var intent = Intent(Intent.ACTION_INSERT)
            .setData(CalendarContract.Events.CONTENT_URI)
            .putExtra(CalendarContract.Events.TITLE, calendarEvent.name)
            .putExtra(
                CalendarContract.EXTRA_EVENT_BEGIN_TIME,
                calendarEvent.startsAtInMillisecondsSinceEpoch
            )
            .putExtra(
                CalendarContract.EXTRA_EVENT_END_TIME,
                calendarEvent.endsAtInMillisecondsSinceEpoch
            )
        activity.startActivity(intent)
    }
}
```

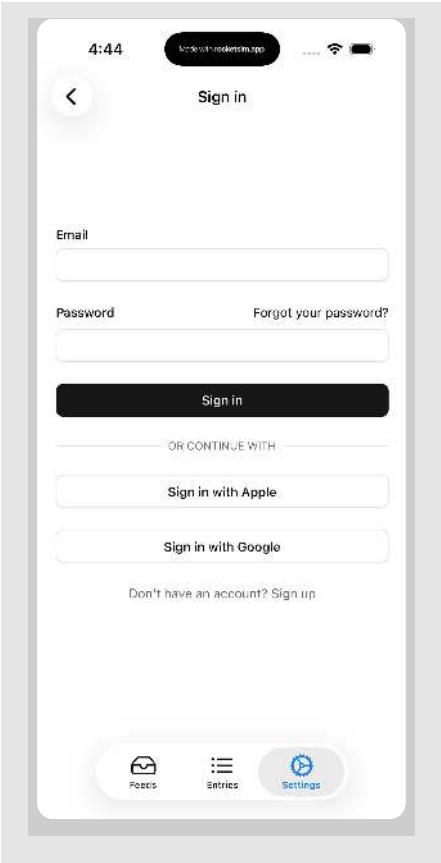
AddToCalendarComponent.kt



OAuth

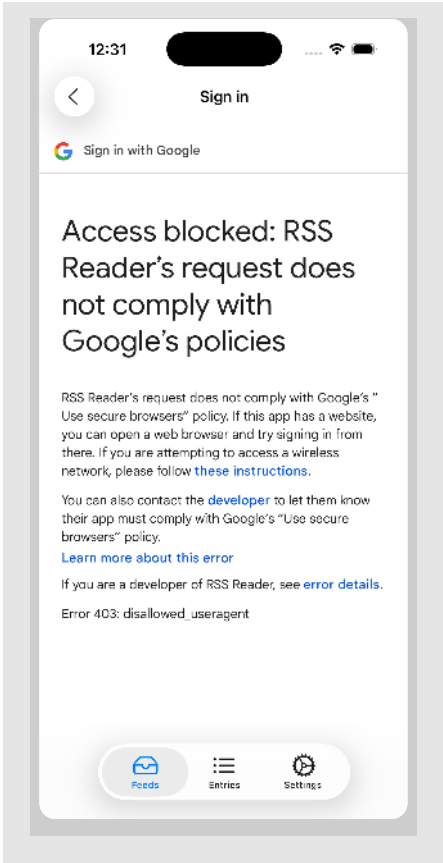
OAuth

Sign in with Apple



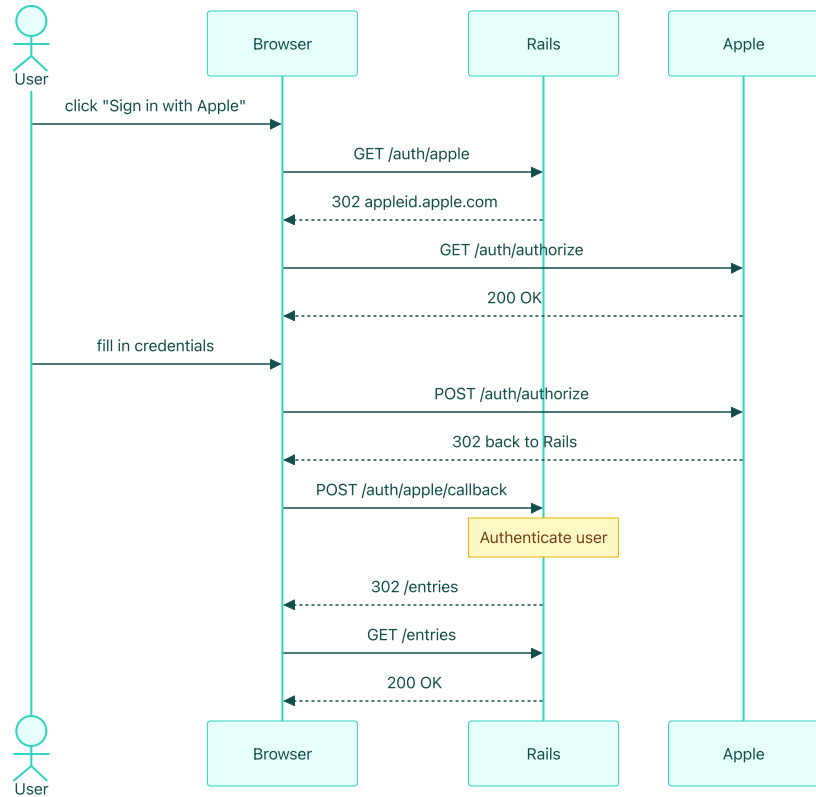
OAuth

Web view problem



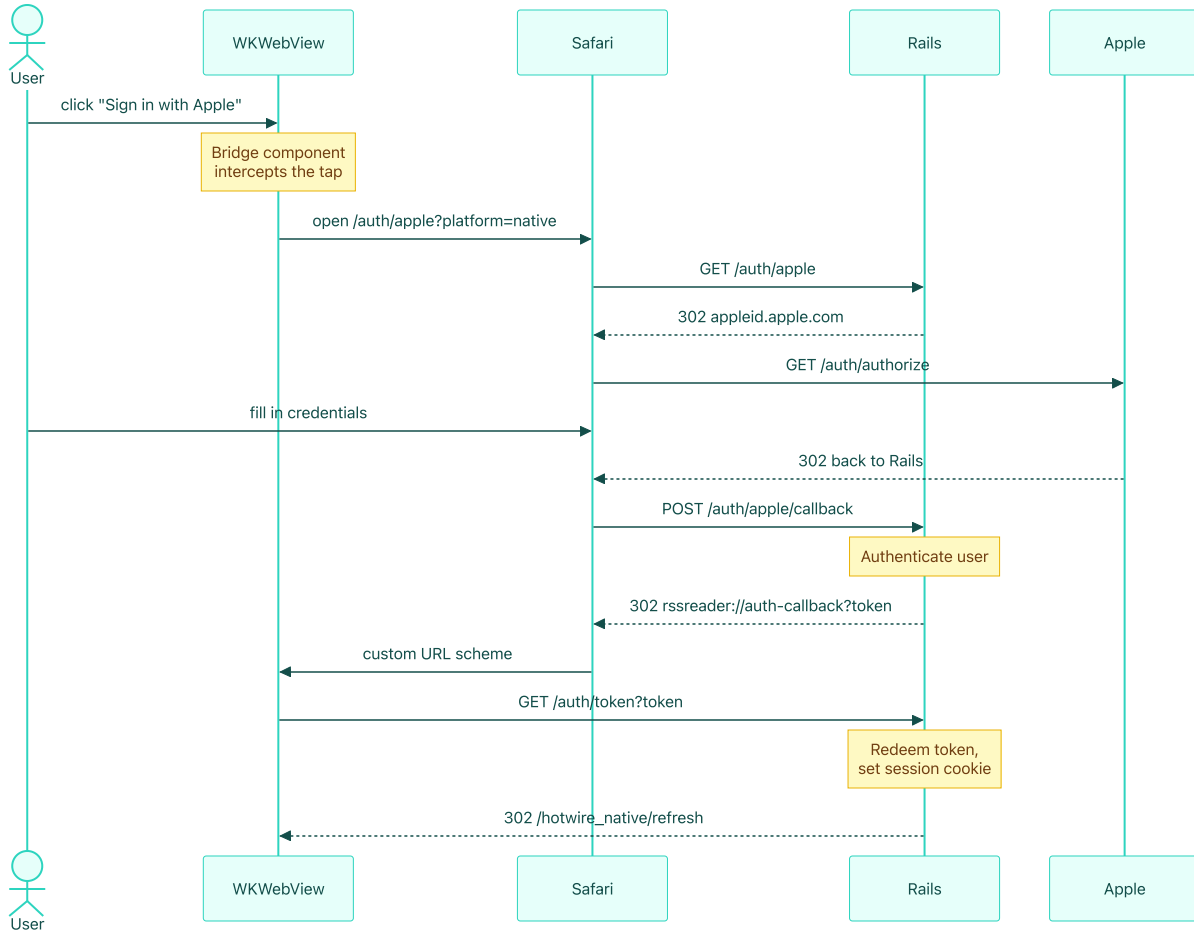
OAuth

Web


User
Browser
Rails
Apple
click "Sign in with Apple"
GET /auth/apple
302 appleid.apple.com
GET /auth/authorize
200 OK
fill in credentials
POST /auth/authorize
302 back to Rails
POST /auth/apple/callback
Authenticate user
302 /entries
GET /entries
200 OK
User
Browser
Rails
Apple

OAuth

Native

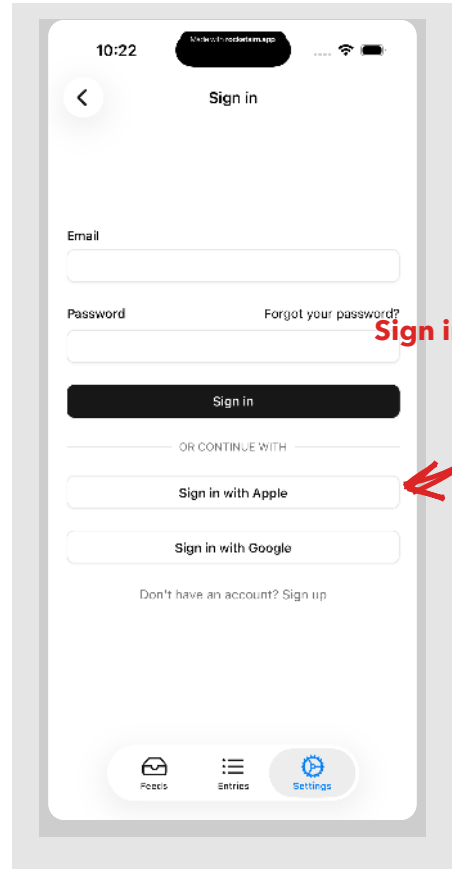


OAuth

Sign in button

```
<%= link_to auth_path(platform: "web"),
  data: {
    turbo: false,
    controller: "bridge--sign-in-with-oauth",
    action: "click->bridge--sign-in-with-oauth#interceptClick",
    bridge__sign_in_with_oauth_start_path_value:
      auth_path(platform: "native"),
    bridge__sign_in_with_oauth_token_auth_path_value:
      auth_token_path
  } do %>
  Sign in with Apple
<% end %>
```

app/views/shared/_oauth_button.html.erb

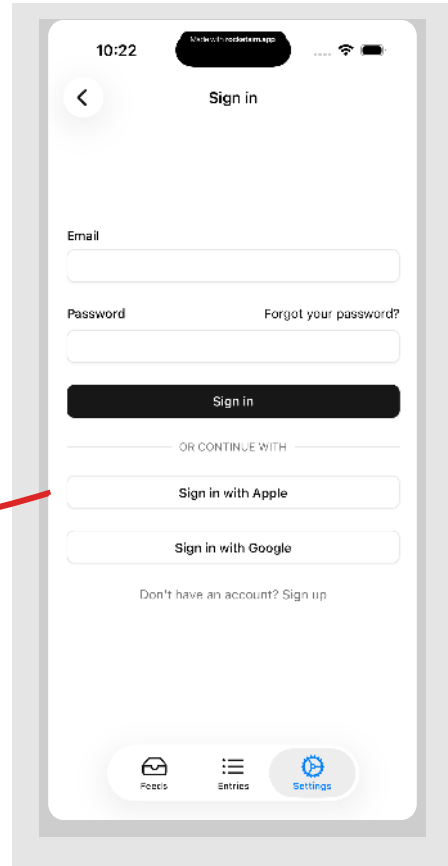


OAuth

Bridge component, Web

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

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

  interceptClick(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



OAuth

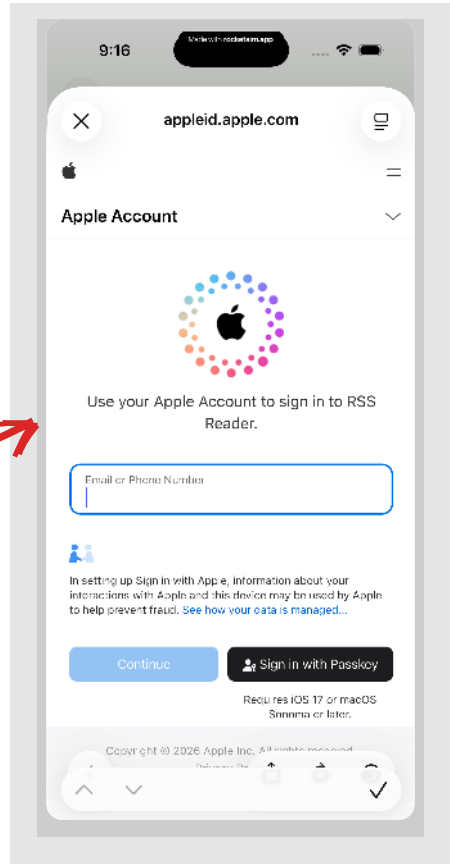
Bridge component, iOS

```
final class SignInWithOauthComponent: BridgeComponent {
    override nonisolated class var name: String {
        "sign-in-with-oauth"
    }

    override func onReceive(message: Message) {
        guard let data: MessageData = message.data() else { return }
        self.tokenAuthPath = data.tokenAuthPath

        let startUrl = URL(string: "\(baseUrl)\(data.startPath)")!
        let safariVC = SFSafariViewController(url: startUrl)
        safariVC.modalPresentationStyle = .pageSheet
        viewController?.present(safariVC, animated: true)
    }
}
```

Components/SignInWithOauthComponent.swift

```
Hotwire.registerBridgeComponents([
    SignInWithOauthComponent.self,
])
```

AppDelegate.swift

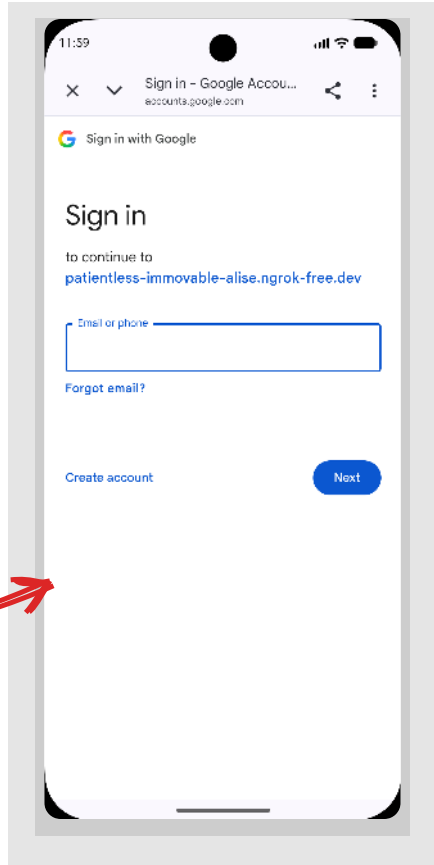


OAuth

Bridge component, Android

```kotlin
class SignInWithOauthComponent(
    name: String,
    private val delegate: BridgeDelegate<HotwireDestination>
) : BridgeComponent<HotwireDestination>(name, delegate) {
    override fun onReceive(message: Message) {
        when (message.event) {
            "click" -> handleClick(message)
        }
    }

    private fun handleClick(message: Message) {
        val data = message.data<MessageData>() ?: return
        tokenAuthPath = data.tokenAuthPath

        val startUrl = "$baseUrl${data.startPath}".toUri()
        CustomTabsIntent.Builder().setShowTitle(true).build()
            .launchUrl(fragment.requireContext(), startUrl)
    }

    @Serializable
    data class MessageData(
        @SerialName("startPath") val startPath: String,
        @SerialName("tokenAuthPath") val tokenAuthPath: String
    )
}
```

components/SignInWithOauthComponent.kt

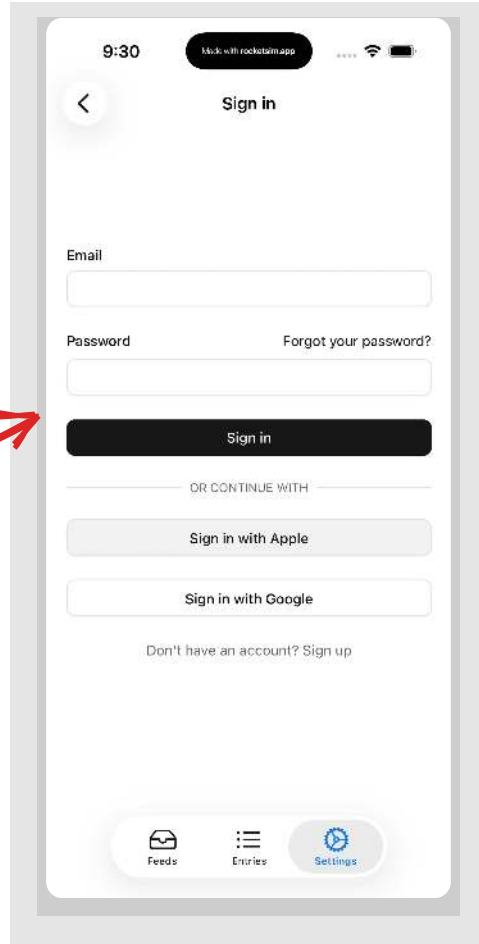


OAuth

Starting the flow

```ruby
def create
  state = OauthState.generate(platform: params[:platform])

  cookies.encrypted[:oauth_state] = {
    value: state.to_message,
    expires: 1.hour.from_now,
    same_site: request.ssl? ? :none : :lax,
    secure: request.ssl?
  }

  redirect_to apple_client.authorization_url(
    state: state.to_message,
    nonce: state.nonce,
    redirect_uri: callback_auth_url
  ), allow_other_host: true
end
```

app/controllers/oauth_sessions_controller.rb

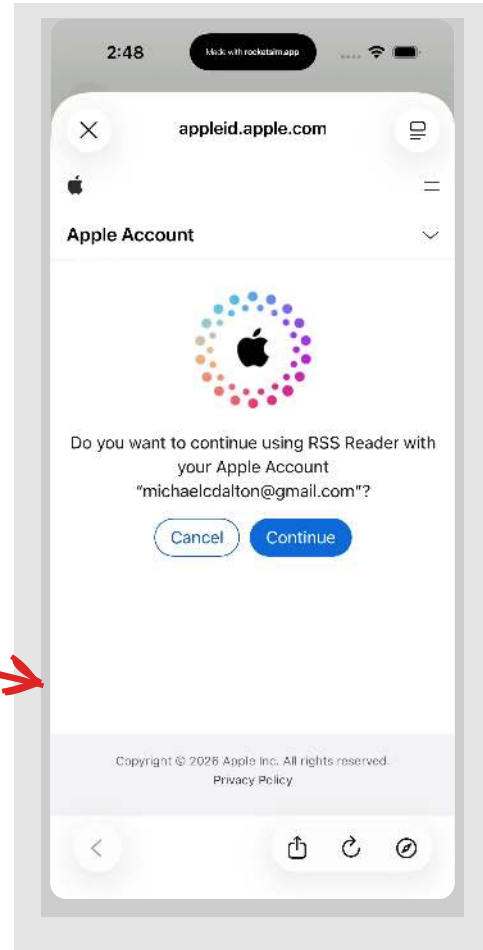


OAuth

Handling the callback

```ruby
NATIVE_CALLBACK_URL = "rssreader://auth-callback"

def callback
  state = verified_state
  return redirect_to welcome_path, alert: INVALID_REQUEST unless state

  credentials = apple_client.authenticate(
    code: params[:code],
    redirect_uri: callback_auth_url,
    nonce: state.nonce
  )

  user = User.find_or_create_from_oauth(credentials)

  if state.native?
    token = user.signed_id(purpose: :native_auth,
                           expires_in: 5.minutes)
    redirect_to "#{NATIVE_CALLBACK_URL}?token=#{token}",
                allow_other_host: true
  else
    start_new_session_for user
    redirect_to after_authentication_url
  end
end
```

app/controllers/oauth_sessions_controller.rb



OAuth

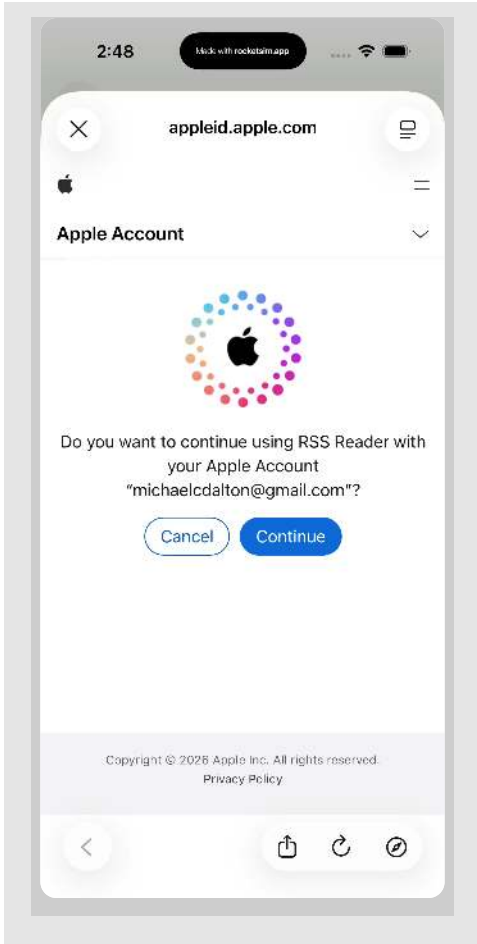
Exchanging the token

```ruby
def authenticate_by_token
  user = User.find_signed(params[:token], purpose: :native_auth)

  if user
    start_new_session_for user
    redirect_to after_authentication_url
  else
    redirect_to welcome_path, alert: UNABLE_TO_SIGN_IN
  end
end
```

app/controllers/oauth_sessions_controller.rb

```ruby
get "token", to: "oauth_sessions#authenticate_by_token"
```

config/routes.rb



OAuth

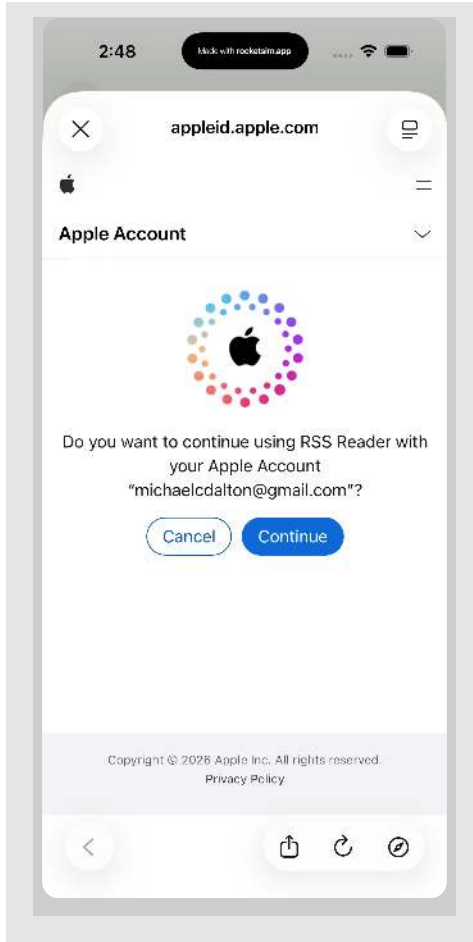
Catching the redirect, iOS

```swift
func scene(_ scene: UIScene,
           openURLContexts URLContexts: Set<UIOpenURLContext>) {
    guard let url = URLContexts.first?.url else { return }
    guard url.host == "auth-callback" else { return }

    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": token]
    )
}
```

SceneController.swift

```xml
<key>CFBundleURLSchemes</key>
<array>
  <string>rssreader</string>
</array>
```

Info.plist



OAuth

Web view sign in, iOS

```swift
@objc private func handleAuthCompletion(_ notification: Notification) {
    let token = notification.userInfo?["token"] as? String

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

private func authenticateWithToken(_ token: String) {
    let url = URL(string: "\(baseUrl)\(tokenAuthPath)?token=\(token)")!

    // Load in the web view so the cookie lands there
    webView.load(URLRequest(url: url))
}
```

Components/SignInWithOauthComponent.swift

```ruby
def authenticate_by_token
  user = User.find_signed(params[:token], purpose: :native_auth)

  if user
    start_new_session_for user
    redirect_to after_authentication_url
  else
    redirect_to welcome_path, alert: UNABLE_TO_SIGN_IN
  end
end
```

app/controllers/oauth_sessions_controller.rb

OAuth

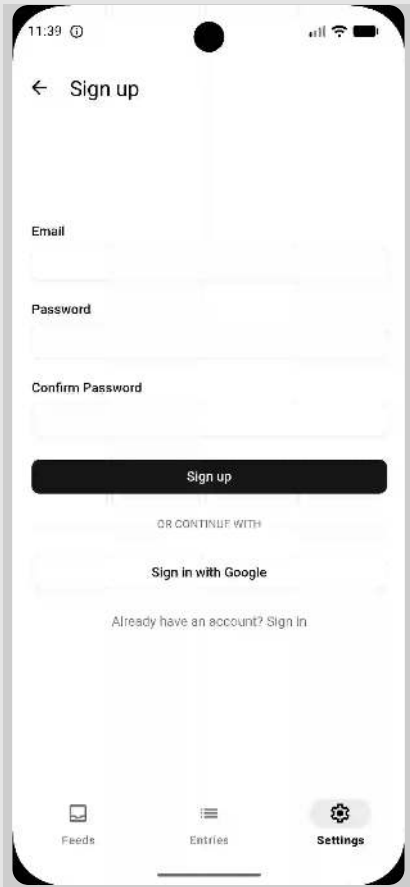
Catching the redirect, Android

```kotlin
override fun onNewIntent(intent: Intent) {
    super.onNewIntent(intent)
    setIntent(intent)
    handleIntent(intent)
}

private fun handleIntent(intent: Intent) {
    if (intent.action == Intent.ACTION_VIEW && intent.data != null) {
        handleDeepLink(intent.data!!)
    }
}

private fun handleDeepLink(uri: Uri) {
    when (uri.host) {
        "auth-callback" -> {
            val token = uri.getQueryParameter("token")
            authTokenLiveData.postValue(token)
        }
    }
}
```

MainActivity.kt

```xml
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <category android:name="android.intent.category.DEFAULT" />
  <category android:name="android.intent.category.BROWSABLE" />
  <data android:scheme="rssreader" android:host="auth-callback" />
</intent-filter>
```

AndroidManifest.xml

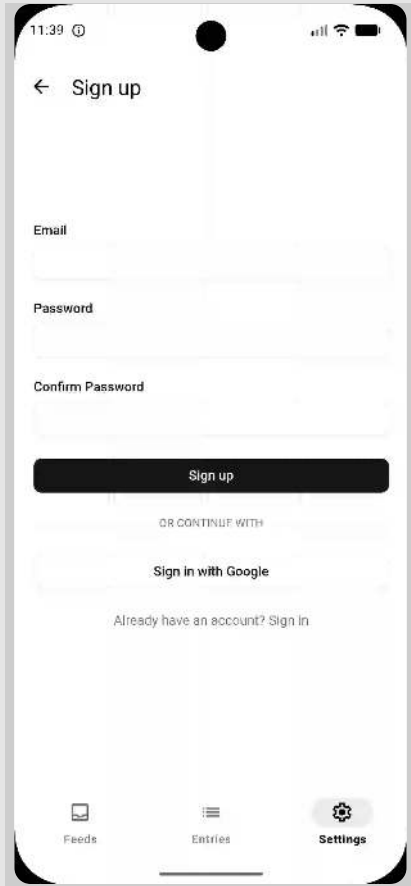


OAuth

Web view sign in, Android

```
private fun observeAuthCompletion() {
    MainActivity.authTokenLiveData.observe(fragment.viewLifecycleOwner) {
        token → authenticateWithToken(token)
    }
}

private fun authenticateWithToken(token: String?) {
    val navigator = fragment?.navigator ?: return

    // Route in the web view so the cookie lands there
    val tokenLoginUrl = "$baseUrl$tokenAuthPath".toUri()
        .buildUpon()
        .appendQueryParameter("token", token)
        .build()

    navigator.route(tokenLoginUrl.toString())
}
```

components/SignInWithOauthComponent.kt



Review

Native navigation and animation

Path configuration to change navigation behavior

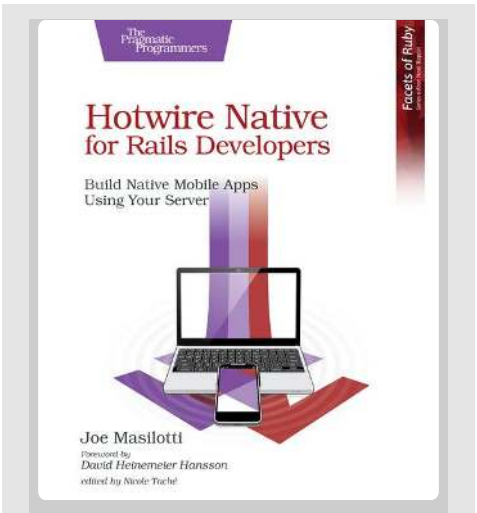
A native tab bar is an easy native component to add

Bridge components when you need a little bit of native functionality

Native components when bridge components aren’t enough

Resources

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



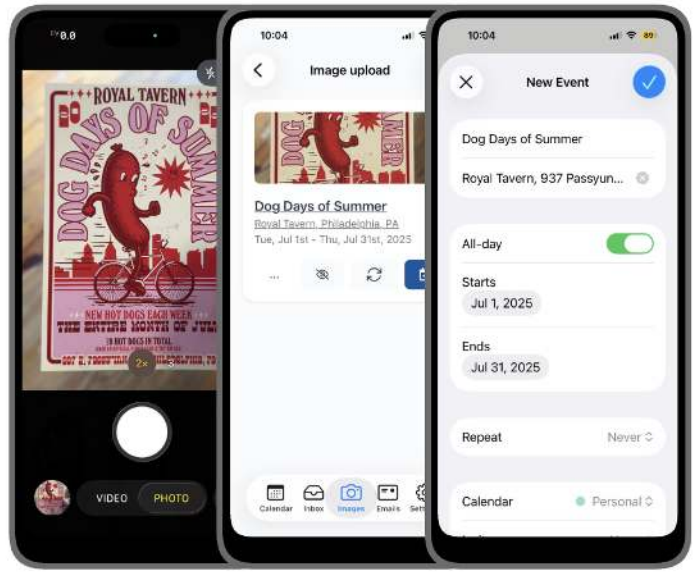
Calendar Vision

Continue as Guest | Sign in | Sign up

Turn images into events in your calendar

Take a photo of a flyer, screenshot, or email and we'll extract the event details for you and add it to your calendar.

Download on the App Store | GET IT ON Google Play



Hotwire Native DIRECTORY

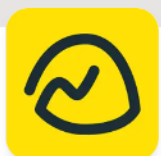
Native iOS/Android applications backed by web views

Add my app

1500cals ›
Health & Fitness

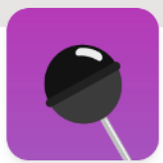
Amba - Health and Wellbeing ›
Health & Fitness

Baheth ›
Education



Basecamp ›
Project Management

BeerMenus ›
Food & Drink



BlackCandy
Music

BraveCo ›

Buzzsprout ›

Calendar Vision ›

Thanks!

Calendar Vision app · calendarvision.app

Find me online · mikedalton.co/socials

Questions? · Session Q&A