

From Swift Skeptic to Migration Lead: What React Native Taught Me About Being an iOS Engineer



**Leo
Picado**

Staff Software Engineer

Zenjob







**Swift's logo is the outline
of an *Apus apus*.**

**I carry one on my
right leg, daily.**

**Three years ago,
my company decided
to stop writing Swift.**

**Some of you came over from the Swift track.
Or the Android one.**

You know exactly what that felt like.

**The rest of you are going to be
standing next to someone who does.**

ACT I

The identity

6 years of iOS, before Germany.



Some of the people I learned from are on the Swift track this week.

2019

**I was in the keynote room
when SwiftUI was announced.**

SOUNDCLOUD, 2021

UIKit → SwiftUI + Combine

Brand new. Not just to the team. To the industry.

*"I ended up more of a multiplier
than a single decision-maker."*

Zenjob is a staffing platform for students.

Short and mid-term work.

2

native apps

Swift and Kotlin.
Both born in 2016.

1:5

the odds

Native engineers, against
backend and web.

Actually, four.

Swift. Kotlin.

React, in web views inside both apps,
for the flows nobody wanted to build twice.

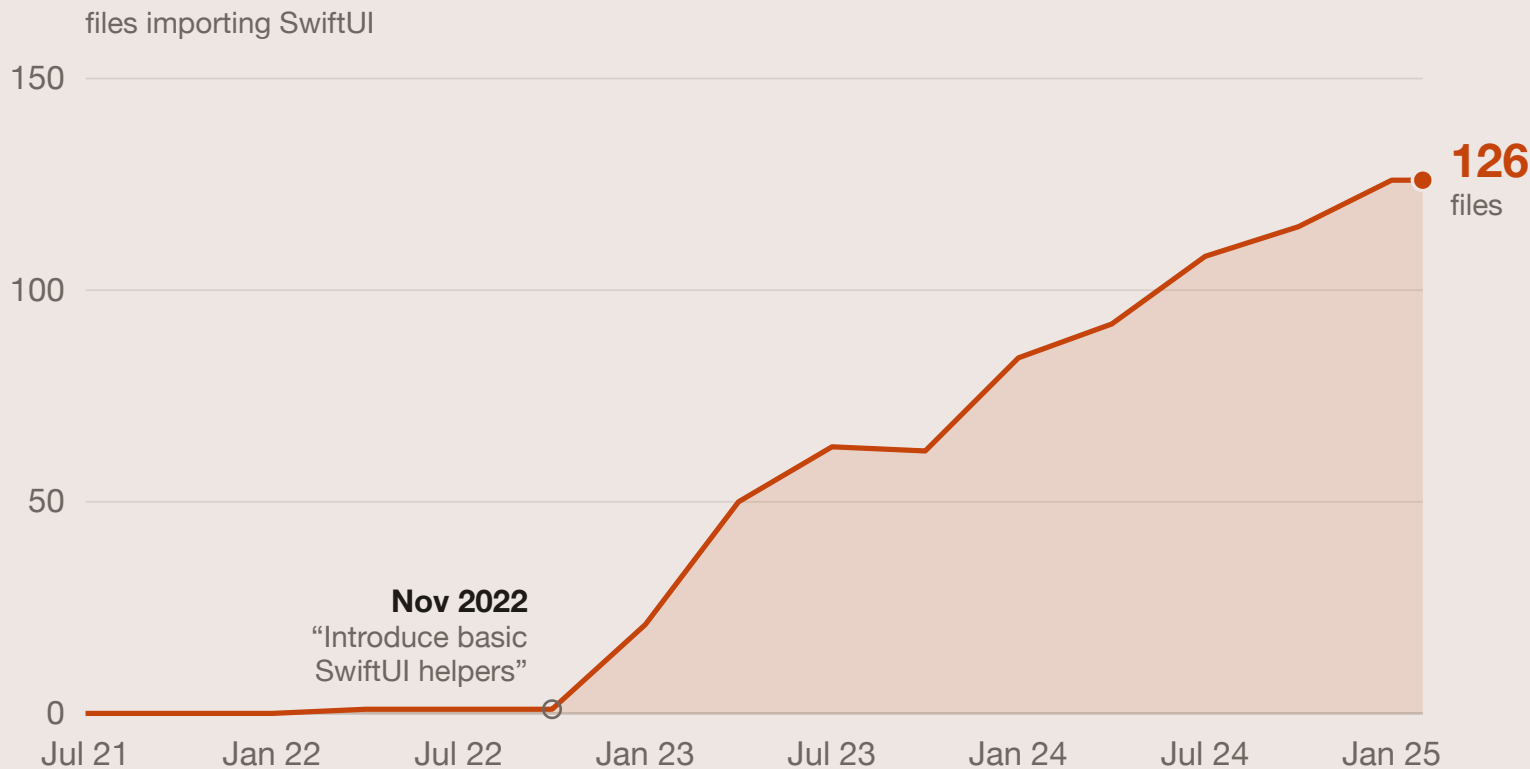
And a fourth app, for a second market.

22 NOVEMBER 2022

"Introduce basic SwiftUI helpers"

Author: me. Nobody assigned it.

Then this happened



zenjob-ios, origin/main — files containing `import SwiftUI`.

MAY 2023

I killed my manager's cross-platform proposal.

It was based on web views.

JUNE 2023 · ONE MONTH LATER

I started learning Kotlin.

Nobody asked me to.

**I've never been sure
those two things were unrelated.**

FEBRUARY 2024 · A 1:1

**My manager had been
playing with Expo on his own.**

**Nearly half the people
touching mobile were writing
every feature twice.**

19 of the 43 who touched mobile.

In the year before the rewrite, committing to both
the iOS and the Android codebases.

**My reputation for defending
my territory**

**made my manager hesitant
to bring me a good idea.**

*"I'll keep an open mind
and try to make sure it works."*

I heard the rumblings.

I mostly shut them down.

From the reading list of the proposal to replace our stack:

***“Is there a drop in native iOS
and Android hiring at startups?”***

Gergely Orosz · The Pragmatic Engineer

**Until the call came
from inside the house.**

This makes me junior again.

This makes the tattoo a joke.

I was literally hired because of this.

**I moved from Costa Rica
to Germany to do iOS.**

I left my life behind for it.

**If Swift is what I'm good at,
and we stop writing Swift,
what am I?**

ACT II

The choice

I had four objections.

OBJECTION 1

**"Airbnb tried this
and quit."**

6

years old by the
time of the pitch

2017

the React Native
it described

63%

of the Airbnb engineers
who used it would
choose it again

**I wasn't citing a study.
I was citing a headline that agreed with me.**

OBJECTION 2

**"I used React Native
at university.
It was a disaster."**

OBJECTION 3

**"This threatens
our expertise."**

OBJECTION 4

**"Our vendor integrations
won't work."**

**Three of my four objections
were about me.**

The fourth one was my job.

My choices

I could leave.

**I could stay and wait to say
*I told you so.***

Or I could find out.

What we actually compared

Flutter

A whole new language, Dart. For the entire org, not just mobile.

Kotlin Multiplatform

The one I wanted to win. But the UI still had to be written twice.

React Native + Expo

Most of the company already wrote React. Some of the app already was React, via web views.

**Being the native guy made me
the best person to evaluate this.**

Not the worst.

I argued hardest against it.

So they asked me to lead it.

ACT III

The work, and what it cost

**The risk was never
the business logic.**

2 Expo Modules

Written in Swift and Kotlin.

Today the app ships **5**.

**This year I shipped OCR
that runs on the device.**

Both platforms. Written once.
Something the native apps never had.

9

months, first commit
to first release

34

people committed
to the new app

1,200+

commits

Then we were ready to roll out.

THE ROLLOUT

Nine days, phased.

0 → 100% of users in under two weeks.

One weekend on call.

Nobody got logged out.

**We replaced the entire binary.
Native to React Native,
and back again if we had to.**

Sessions survived it, in both directions, by design.

Reversible. In both directions.

You cannot roll an app store backwards.

**So the way back was forwards:
the old codebase, shipped as a newer version.**

Submitted the day React Native went live, on an expedited review.
Approved in hours and held there. A force update would have done the rest.

**We cut two features
the usage data said
almost nobody needed.**

**Infrequent is not the same
as unimportant.**

One of them buried our support team.

**No usage data. No product owner.
Nobody in the building to ask.**

**Both were back in production
within a week.**

Over the air. No release train, no store queue.

**That was the week
it paid for itself.**

548 releases. 9 years

That's about **60 a year.**

About **5 a month.**

A little over **1 a week.**

Which is a perfectly healthy cadence.

That's both apps.

One release, per app, every other week.

548 releases

9 years

the two native apps,
Swift and Kotlin

11 months

the React Native app

It's at 923 now.

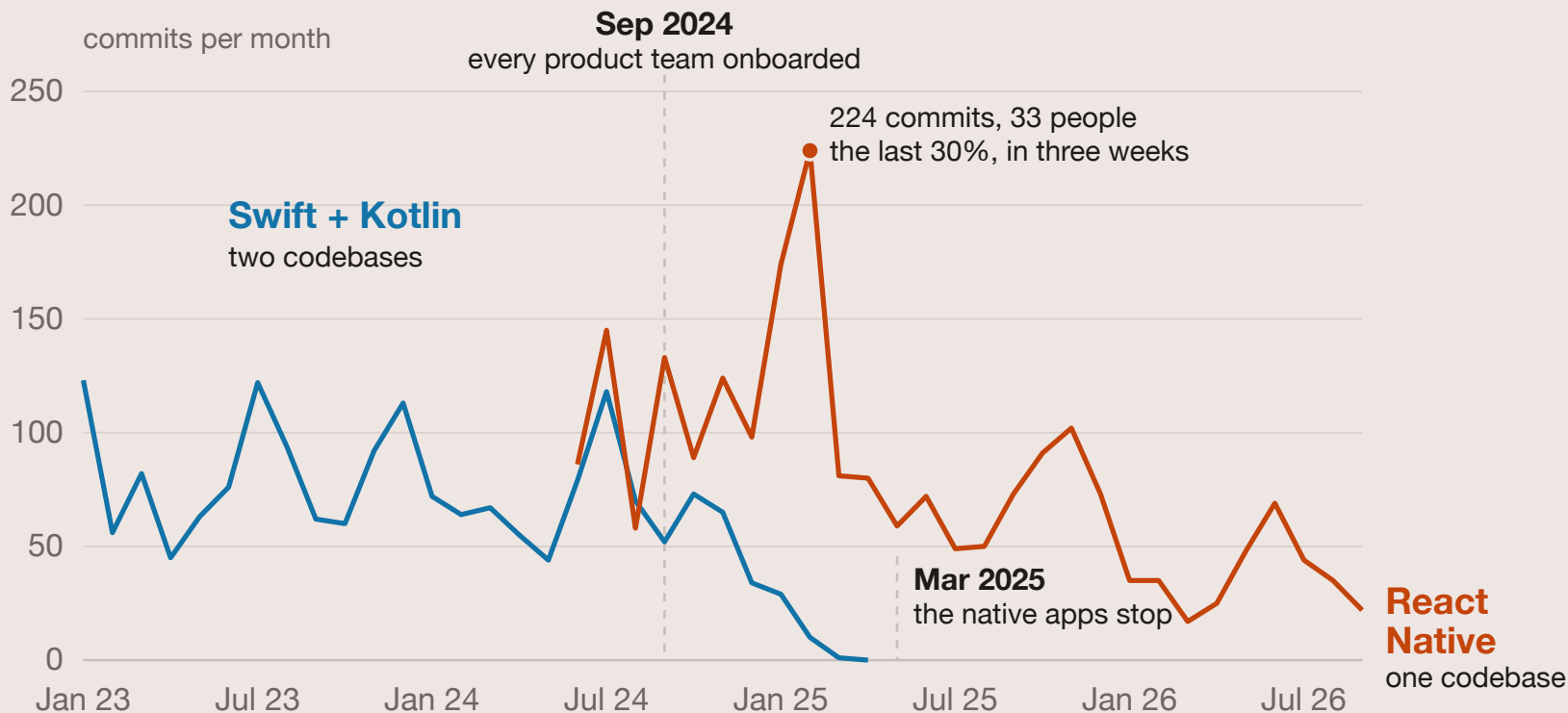
What it bought

4 weeks

to get a copy change to a user

7 min

from merge to a user's phone



Mainline commits, merges and bots excluded. Merge conventions differ between these repos, so read this as the shape of a handover — not a productivity comparison.

**And then the rest
of the company followed.**

A year later, the backend chose TypeScript.

“The mobile team had recently completed a migration to React Native with TypeScript. The backend was the last part of the organization still on the JVM.”

SEPTEMBER 2026

**Shopify went back
to native.**

"Coding models have gotten dramatically better, and for our apps and our team, building the same feature in Swift and Kotlin no longer carries the cost it used to."

Shopify Engineering, September 2026

"...unlocking the ability for a single mobile contributor to be productive in both iOS and Android codebases simultaneously."

Coinbase · Senior Staff Software Engineer, Native Mobile

Job posting, 29 September 2026

**If they're right,
everything I just told you
is a period piece.**

**They're right about the part
they're talking about.**

**Writing the same feature twice was the tax.
Nearly half our people were paying it.**

If a model writes the Kotlin while you write the Swift,
that tax gets a lot smaller.

It made writing cheaper.

It didn't make shipping faster.

Or cheaper to test.

Two codebases still means two test suites.

What it bought

4 weeks

to get a copy change to a user

7 min

from merge to a user's phone

**No model shortens App Store review.
Or Play Store review.**

Two apps. Two queues. Every time.

Approval isn't arrival.

**A binary arrives
when people update.**

Days. Often weeks.

**Ours arrives
on next launch.**

Over the air.
7 minutes from merge.

What a better model doesn't touch

One pipeline. Not two.

A real dev loop. Previews aren't the running app.

Over-the-air updates. The stores won't let you.

RUNWAY, 2025 · 300 MOBILE ENGINEERS

**77% of teams need a hotfix
every three to five releases.**

**The question was never whether
you'll ship outside the train.**

It's how long the fix takes to arrive.

And look at what it took them to go back.

Helix: an in-house system that reviews every migrated slice,
with tests and visual verification required and an adversarial reviewer.

An architecture rewritten so agents could run the app headlessly.

A platform organisation to build both.

Twelve weeks to rebuild Shop.

Shopify's answer is available to Shopify.

8 SEPTEMBER 2026

Expo Modules 2.0

A native module is now an annotated Swift or Kotlin class.

No DSL, no boilerplate, around five times faster.

**Shopify left to get closer to the platform.
The platform is coming closer anyway.**

**Somewhere in Shopify there's an engineer
with a React Native sticker on their laptop,
who found out a month ago.**

**You are one blog post away
from being me.**

So what did it cost?

People ask if I miss Swift.

I do not.

I miss being *that guy*.

**I was "that guy" partly because
everyone else was frightened of the app.**

**Then I helped demolish it.
On purpose.**

THEN

*"You should use an early
return here: guard let,
not nested if let."*

*"You're not leveraging
extensions; this
could be far terser."*

NOW

*"We've got this entire initiative
that we need to ship in a
week. What do you think?"*

**In the two-codebase world,
nobody would have asked me that.**

"Can we ship an entire initiative in a week?"
wasn't even a consideration

when a copy change took 4 weeks.

529

commits in Swift

21

in Kotlin

234

in TypeScript

SOUNDCLOUD

**I helped move an app
to Apple's
newest stack.**

ZENJOB

**I helped move an app
away from it.**

SOUNDCLOUD, 2021

Ran bug bashes

PR review across squads

Paired with everyone

Over ten engineers

ZENJOB, 2024–25

Ran an in-house bug bash

Paired every team with a mobile engineer

Made other people capable

Every product team in the company

Same job. Twice.

I was never the syntax.

I was the judgment.

Knowing which risks were real.

Knowing where it breaks.

The syntax was just how I'd been proving it.

So am I still *that guy*?

Yes. In more rooms than before.

Swift just isn't one of them.

**You are not less of a problem solver
because you stopped typing
the language you love.**

**The native engineer next to you
is the reason your migration
will work, or won't.**

**Don't ask them to be less native.
Ask them where the bodies are buried.**



**The tattoo
is still there.**

**It's a story now,
not a job title.**

Thank you

Leo Picado

leopicado.com/talks/swift-skeptic

The Zenjob posts, the Expo post, and these slides.

