Modern mobile development
Native vs cross-platform apps
Sebastiano Poggi
Scope

‣ Goal: help you choose

‣ Agenda
  › Preconditions for success
  › Understanding mobile
  › Native or cross-platform
  › Pick a cross-platform stack
“It depends”

- Everything in this talk may or may not apply to you
- Apply common sense
Terminology

Native app
- Uses native build tools
  - Android: Kotlin/Java/C++
  - iOS: ObjC/Swift

Cross-platform app
- Non-native build tools
- Potentially uses web tech
- Same tech across OSes

Neither runs in the browser
Company dynamics
Why are you here?

- You want a mobile app
- Greenfield vs The Big Rewrite™
  - Tech debt
  - Pre-existing team
  - Recovering from failure
Teams

- No full-stack engineers in mobile
  - Mobile devs dislike backend work
  - ...and vice versa
- Information/knowledge silos
- Misalignment and misunderstandings
Product vs Tech

- Different chains:
  - Mobile reports to CPO
  - Web and backend report to CTO
- Mobile as nobody’s child
  - Management doesn’t “get” it
  - Tech not built for it
Not all tech is created equal

- Web is almost always ahead
  - Mobile comes later
- Web is straightforward
- Mobile can exacerbate org issues
When things go wrong
Nobody likes failure

- Failure causes management frustration
- Blaming games
- Tech stack as way to shift responsibility
- Wrong choices for the wrong reasons
Bad apps exist...

- Bad choices → bad apps
  - Don’t force choices, evaluate assumptions
- Tech stacks don’t always work 1:1 on mobile
- Reach outside comfort zone
  - Ensure higher-ups’ buy-in
Users don’t care about the tech
Users don’t care about the tech

They just want to get stuff done
Users don’t care about the tech

They just want to get stuff done

Help them, help your business
(Re)starting
Before you (re)start

› Ask the tough questions
Before you (re)start

› Ask the tough questions

Do your users want, or need, a mobile app?
Before you (re)start

▶ Ask the tough questions

Can you satisfy your users with a high quality, responsive website?
Before you (re)start

▪ Ask the tough questions

Does your competition have an app?
Before you (re)start

- Ask the tough questions

*Do their users use it?*
Before you (re)start

› Ask the tough questions

How good is it?
Before you (re)start

- Ask the tough questions
- Use data to drive decisions
  - Focus groups, user studies, etc
- Trust the data
  - Even when you don’t like it
Scoping and responsibilities

‣ Who owns mobile?
  ▶ Align with rest of tech if possible
‣ Think about your users
  ▶ What do they want to do?
‣ Define app scope and what’s not in it
Capability vs capacity

▷ What can your existing teams do?
  ▷ Any native mobile devs?
  ▷ Do they want to do cross-platform?
▷ Ensure platform-native capability
  ▷ You’ll need it
Capability vs capacity

‣ Most apps require OS interactions
  ‣ If your app doesn’t, consider a website
  ‣ “Website apps” waste resources
‣ E.g.: ReactJS dev team doing mobile?
  ‣ Somewhat different tech and tools
  ‣ Native knowledge required
Team participation

‣ Involve your devs in the choice
 ‣ Listen to their fears
 ‣ Provide safety
‣ Avoid chasing tech fads
‣ Spikes are good
 ‣ ...but can deceive
Commitment

‣ In for the long run
  ‣ Big investment
‣ Huge switching costs
  ‣ Tech and skill lock-in
‣ Change of tech means rewrite
Native or cross-platform?
The native advantage

‣ Native is always “better”
  ‣ Better performance
  ‣ Better integration and support
  ‣ More consistent with the OS
  ‣ More APIs/features
‣ Tooling is constantly improving
Not all is rosy

- Native is more expensive
  - Dedicated team per OS
- Infrastructure & processes
  - Different CI setups
  - Different deploy and publishing
The cross-platform pragmatism

- Native may not be the best for you
- Cross-platform may be “enough”
  - Vastly improved over the years
  - Some dev experience advantages
- Prefer strong, non-native design language
A fictional app case study

- Wearables company
  - Do they need an app?
A fictional app case study

- Wearables company
  - Do they need an app? ✅
A fictional app case study

› Wearables company
  › Do they need an app? ✓
  › Do they need a native app?
A fictional app case study

‣ Wearables company
  ‣ Do they need an app? ✓
  ‣ Do they need a native app?
    ‣ Using the OS APIs heavily?
A fictional app case study

‣ Wearables company
  ‣ Do they need an app? ✓
  ‣ Do they need a native app?
    ‣ Using the OS APIs heavily? ✓
A fictional app case study

- Wearables company
  - Do they need an app? ✓
  - Do they need a native app?
    - Using the OS APIs heavily? ✓
  - Can users achieve their goals?
A fictional app case study

‣ Wearables company
  › Do they need an app? ✓
  › Do they need a native app?
    ➔ Using the OS APIs heavily? ✓
    ➔ Can users achieve their goals? ✓
A fictional app case study

- Wearables company
  - Do they need an app? ✓
  - Do they need a native app? ✓
    - Using the OS APIs heavily? ✓
    - Can users achieve their goals? ✓
The main choices

React Native

Xamarin

Flutter

Web-based

Cordova

Ionic

PhoneGap

Kotlin MP
The main choices

- React Native
- Xamarin
- Flutter
The main choices

- React Native
  - From Facebook
  - Derived from ReactJS
    - Share skills/code with web team
    - Built on JavaScript and npm
    - 3rd party supports desktop/wearables/tv/...
The main choices

- You can make B2C apps with it
  - Plenty of “big” RN apps
- Performance has some limitations
- Custom UI needs per-platform implementations
- Famous cases of apps abandoning it
The main choices

- **React Native**
  - From Microsoft
    - Used to be paid, now it’s free and OSS

- **Xamarin**
  - Uses C# tools and NuGet, “full stack”
  - Unique UI approach
    - Xamarin.Forms or native views

- **Flutter**
The main choices

- React Native
  - Wraps and exposes platform-native APIs
  - Limited support and tools
    - Best for internal and unsophisticated apps
    - Very enterprise-oriented
  - Unsuitable for B2C apps?

- Xamarin

- Flutter
The main choices

- React Native
- Xamarin
- Flutter

- From Google
  - Quickly rising in popularity
  - Lots of investments & marketing
  - Great 1st party integrations (Firebase)
- Uses Dart and Pub
The main choices

- Mobile, desktop, web, embedded
  - No WatchOS and tvOS
  - Full-stack: backends in Dart
- Best-in-class testing capabilities
- Dev audience skewed to Android
The main choices

- React Native
- Xamarin
- Flutter
A sense of scale

- Cordova: 17% (iOS), 20% (Android)
- React Native: 12% (iOS), 15% (Android)
- Flutter: 6% (iOS), 14% (Android)
- Ionic: 8% (iOS), 11% (Android)
- Xamarin: 4% (iOS), 5% (Android)
- Titanium/Appcelerator: 1% (iOS), 1% (Android)

Source: AppFigures.com
Numbers can deceive

~1/3rd of all mobile developers uses cross-platform tech

Source: Statista.com
Numbers can deceive

Source: Statista.com
Numbers can deceive

Source: Statista.com
Numbers can deceive

Source: Statista.com
Numbers can deceive

Source: Statista.com
Another fictional case study

- Investment (fintech) company
  - Do they need an app?
Another fictional case study

- Investment (fintech) company
  - Do they need an app?  

✓
Another fictional case study

- Investment (fintech) company
  - Do they need an app? ✓
  - Do they need a native app?
Another fictional case study

- Investment (fintech) company
- Do they need an app? ✓
- Do they need a native app?
  - Using the OS APIs heavily?
Another fictional case study

- Investment (fintech) company
  - Do they need an app? ✔️
  - Do they need a native app?
    - Using the OS APIs heavily? ❌
Another fictional case study

- Investment (fintech) company
  - Do they need an app? ✓
  - Do they need a native app?
    - Using the OS APIs heavily? ✗
    - Can users achieve their goals?
Another fictional case study

- Investment (fintech) company
  - Do they need an app? ✓
  - Do they need a native app?
    - Using the OS APIs heavily? ✗
    - Can users achieve their goals? ✓
Another fictional case study

- Investment (fintech) company
  - Do they need an app? ✅
  - Do they need a native app? ❌
    - Using the OS APIs heavily? ❌
  - Can users achieve their goals? ✅
Another fictional case study

‣ Which cross-platform framework?
Another fictional case study

‣ Which cross-platform framework?

Strong in-house ReactJS team
Another fictional case study

› Which cross-platform framework?

[X] Strong in-house ReactJS team
Another fictional case study

‣ Which cross-platform framework?

❌ Strong in-house ReactJS team

Strong in-house .Net team
Another fictional case study

› Which cross-platform framework?

❌ Strong in-house ReactJS team
❌ Strong in-house .Net team
Another fictional case study

› Which cross-platform framework?

- Strong in-house ReactJS team
- Strong in-house .Net team
  Using Firebase services
Another fictional case study

- Which cross-platform framework?
  - ❌ Strong in-house ReactJS team
  - ❌ Strong in-house .Net team
  - ✅ Using Firebase services
Another fictional case study

- Which cross-platform framework?
  - ❌ Strong in-house ReactJS team
  - ❌ Strong in-house .Net team
  - ✔ Using Firebase services
  - Lots of custom UI
Another fictional case study

- Which cross-platform framework?

  - Strong in-house ReactJS team ✗
  - Strong in-house .Net team ✗
  - Using Firebase services ✓
  - Lots of custom UI ✓
Another fictional case study

Which cross-platform framework?

- Strong in-house ReactJS team
- Strong in-house .Net team
- Using Firebase services
- Lots of custom UI

Flutter
Testing on mobile

- Unit testing
  - More or less a solved problem
- UI testing
  - Easier in Flutter (widget tests)
  - Native, React Native use per-platform tools
  - Xamarin too, but custom
Testing mobile UI on CI

- Instrumented tests
  - Slow, runs on virtual/physical devices
  - Cloud services exist, but expensive
  - Complex to set up and maintain
- Workarounds: more unit testing
- Specialised mobile CI solutions
Review decisions

- Bad choices will not be immediately clear
  - RN apps abandoning RN after years
  - Flutter may turn out the same

- Keep an eye on the progress
  - Failure will be expensive...
  - ...but stopping early will help
Takeaways
1. Understand if mobile can work for you

Make data-driven decisions
1. Understand if mobile can work for you
   Make data-driven decisions

2. Create the right environment
   Sort out organisation and teams
1. Understand if mobile can work for you
   Make data-driven decisions

2. Create the right environment
   Sort out organisation and teams

3. Assess the compromises
   There is no silver bullet
1. Understand if mobile can work for you
   Make data-driven decisions

2. Create the right environment
   Sort out organisation and teams

3. Assess the compromises
   There is no silver bullet

4. Make the right choice and GO!
   Hopefully this talk helped you
That’s all, folks!

Questions?
Sebastiano Poggi

twitter.com/seebrock3r
go.sebastiano.dev/qcon-2022