

Don't do native if you want VC 

and why Google frameworks are very
startup friendly  

Agenda

- Disclaimers, speaker, blah blah blah
- Startups 101 recap
- A mental model for CTOs in the making
- A good generalization: **avoid native**
- The risk of technical debt
- Q&A

Me

@giansegato practically everywhere

- founder & ceo @ **Uniwhere**: we use data to make students' journey from college to the job market less stressful; raised 600K+; top 100 Startups in Berlin
- Economics background, fluent in Python, crush for Flutter
- Started freelancing as full-stack dev in 2007
- I blog about startups & stuff giansegato.com

Disclaimers

- Cross-talk between Finance & Tech
- Not much code will be shown
- Google technology will come at the end...
as deus ex machina
- Still in time to leave, bros & sisters



Nerd presentation stack

1. Markdown + Deckset
2. Yes, this is the *Solarized* color scheme

```
make students' journey from college to the job
market less stressful; raised 600K+; top 100
Startups in Berlin
- Economics background, fluent in Python, crus
for Flutter
- Started freelancing as full-stack dev in 200
- I blog about startups & stuff [giansegato.co
(https://giansegato.com/)

---

# Disclaimers

! [right filtered] (https://images.pexels.com/
photos/2170649/pexels-photo-2170649.jpeg?
auto=compress&cs=tinysrgb&dpr=2&w=500)

- Cross-talk between Finance & Tech
- Not much code will be shown
- Google technology will come at the end... as
deus ex machina
- Still in time to leave, bros & sisters

---

# Nerd presentation stack
```

Let's say that you have an idea...

(it always starts like this)

- A friend comes and asks for help
- You do actually have a great idea
- You get some money or time from your company
- You start a petty project that scales like crazy

Congrats. You're now the CTO!

— and very good luck

Startups 101

What on earth is the CEO thinking?!

1. Startups: 1B \$ companies in the making
2. Ultra-scale fast propelled by other people's money (spoiler alert: agencies are not startups)
3. These people are called VC
4. They want to see proof before going for the bank transfer

Proof VCs want

- Hint that there's the so called product-market fit
- In pragmatic terms: **prove traction!**
- Issue: you have to iterate a lot because you have no fucking clue
- Double issue: you have to move fast

Why fast? Why iterate?

I know what I want to build and I
know how much time it will take.

— you, now.

«As a startup you're either trying to do something no one knows how to do or competing with 50 other companies»

— @austen

«You're doing both.»

— me.

MVPs: minimum viable products

try everything by changing everything, fast

- small set of features
- complete enough to show that there's something
- keeping its complexity as low as possible, as you can't know in advance when and how this fit will happen

Reframing the problem

MVP = constrained minimization problem.

- minimize costs
- with the constraint of still being able to verify whether your biz idea works or not

Constraint

- Constraint = the product specs, the definition of the features that you'll have to defend during an investor meeting
- responsibility of the product team (CEO/CPO)

Minimization

- Minimization = Under the control of the tech team (CTO), which gets the specs and has to find a way to make them happen

«We don't care about specs. We're nerds.»

— us, today.

Mental model for CTOs

who wants to minimize whatever comes from the specs

Mental model for first-time CTOs

1. Minimize the number of code repos
2. Minimize code that is not visible
 - 2.b Corollary: outsource non-critical complexity
3. You (almost) only need numbers

1. Minimize code repos

AKA: centralise your tech stack

Minimize the number of code repos

- multiple code repos = different languages, different specialized developers and different dependencies
- which means:
 - coordination costs
 - management boilerplate
 - sunk costs
- growth + retention can't offset cost increase

Which means...

1. Go cross-platform
2. Avoid a backend altogether, if possible

2. Minimize code that is not visible

AKA: prioritize fast deploys over your ability
to scale

Minimize technology invisible to the users

- scale-related things should be significantly de-prioritized
- even documentation...
- especially management :-)

2b. Corollary: outsource non-critical
complexity

AKA: use off-the-shelf tools as much as you
can; nobody will remember your brand
anyway

3. You (almost) only need numbers

AKA: let's make an exception for analytics!

Wrapping up: Fast > scale

- No backend
- Single client codebase
- Heavy outsourcing
- Remove management boilerplate

Google is totally in line with this approach

- No backend: Firebase Cloud Functions
- Fast deploy: Flutter
- Cross-platform: Flutter
- Outsource: Firebase Auth
- Remove management: Material Design

Y u so angry? I love native tho...

— you, native developer.

Native development costs. **A lot.**

- coordinate different devs doing the same thing
- adapt the very same feature to different environments (tech infrastructure & UI)
- track and then consolidate different metrics sources and KPIs
- very different performance marketing campaigns
- different marketing assets for each platform

cost(scale_opportunity) > benefit(proving_scalability)

tl;dr: kicking off a startup?

Flutter & Firebase FTW

But wait... tech debt!


- Usually... you don't raise only once
- As CTO you have to speak w/ the CEO to include refactoring costs in the next funding round
- Money to rewrote everything from scratch again...

A photograph of a man and a young child sitting in a red go-kart on an indoor track. The man is smiling and looking at the child, who is also smiling and holding the steering wheel. The track is surrounded by other colorful go-karts and has large windows in the background. The image has a dark blue overlay.

MVP: "this might work..."

Seed: "this can scale!"



A photograph of a server room with rows of server racks. The floor is tiled with a checkered pattern. The image is overlaid with a dark blue tint.

As CTOs, you have to make sure
that the tech strategy follows the
fundraising one

Thanks.

Q&A