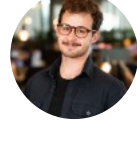


# How we misunderstood microservices



Afonso Delgado

Follow

Apr 2 · 4 min read

★

🐦

🌐

📘

📌

👤

SumUp Software Engineer Afonso Delgado shares some eye-opening insights from last year’s O’Reilly Software Architecture Conference.



O'Reilly Software Architecture Conference in Berlin, November 16, 2019.

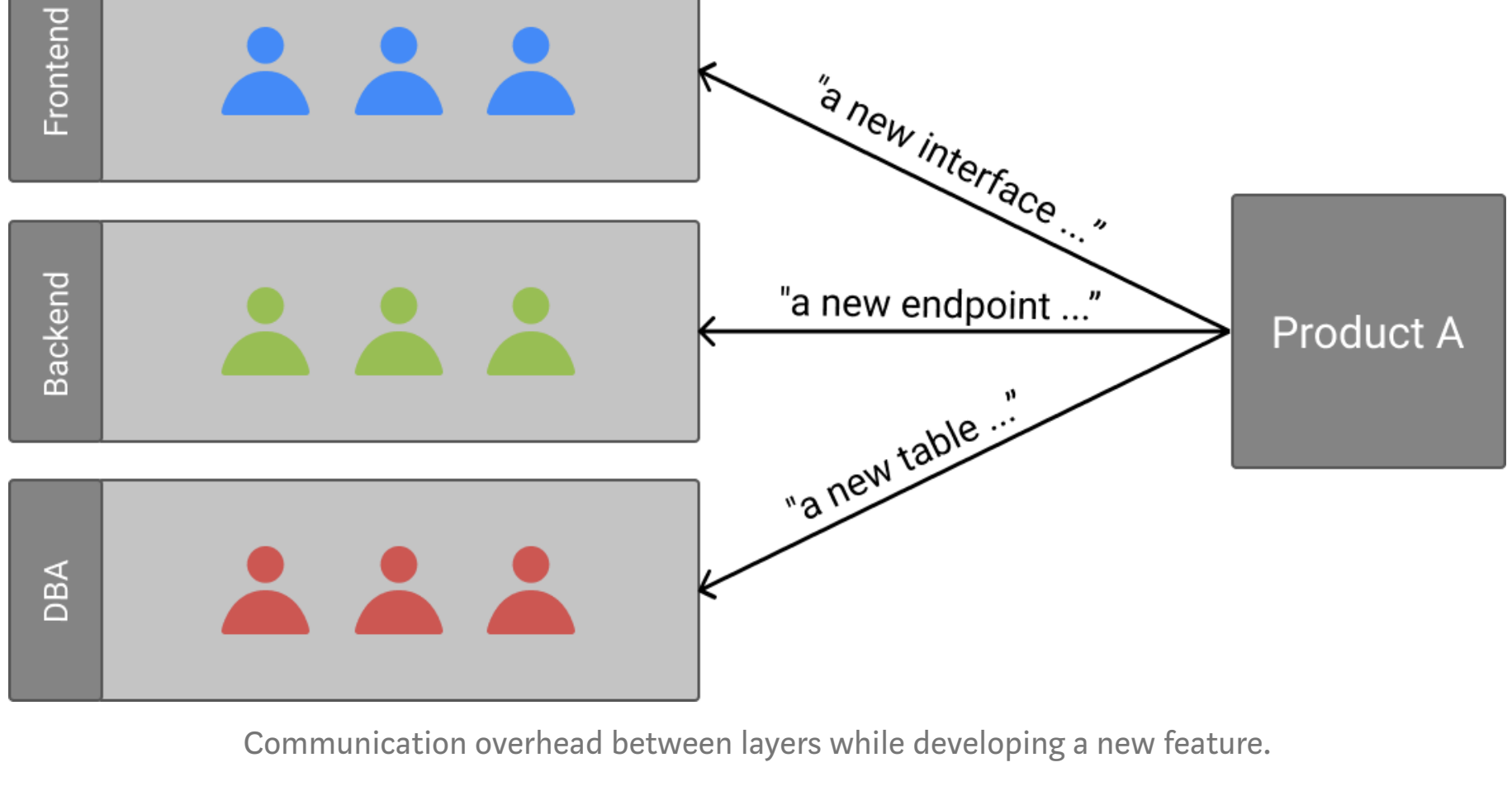
In November 2019, I had the opportunity to attend the O’Reilly [Software Architecture Conference](#) with SumUp and although there were many amazing talks, there was one that really caught my attention.

[Erik Dörnenburg](#) was scheduled to speak about Patterns for Microfrontends (you can check the full presentation [here](#)). I was pleasantly surprised to see a frontend architecture-related talk at such a big software architecture conference, and excited for a deep dive on the topic.

During the talk, the audience was presented with patterns that can be used to apply a micro frontend approach and start the decoupling of a frontend monolith. But what really resonated with me was the introduction, during which Erik explained what microservices should be from the beginning and, at some level, how we’ve misunderstood microservices.

Organisations which design systems are constrained to produce designs which are copies of the communication structures of these organisations — **Melvin Conway**

If we take a look at how we used to form our teams (and in some cases still do), the effort put into delivering a product was divided into layers, and if you’re familiar to [Conway’s Law](#) you’ll know that we would be constrained to design systems that copy the communication structures of our organisation.

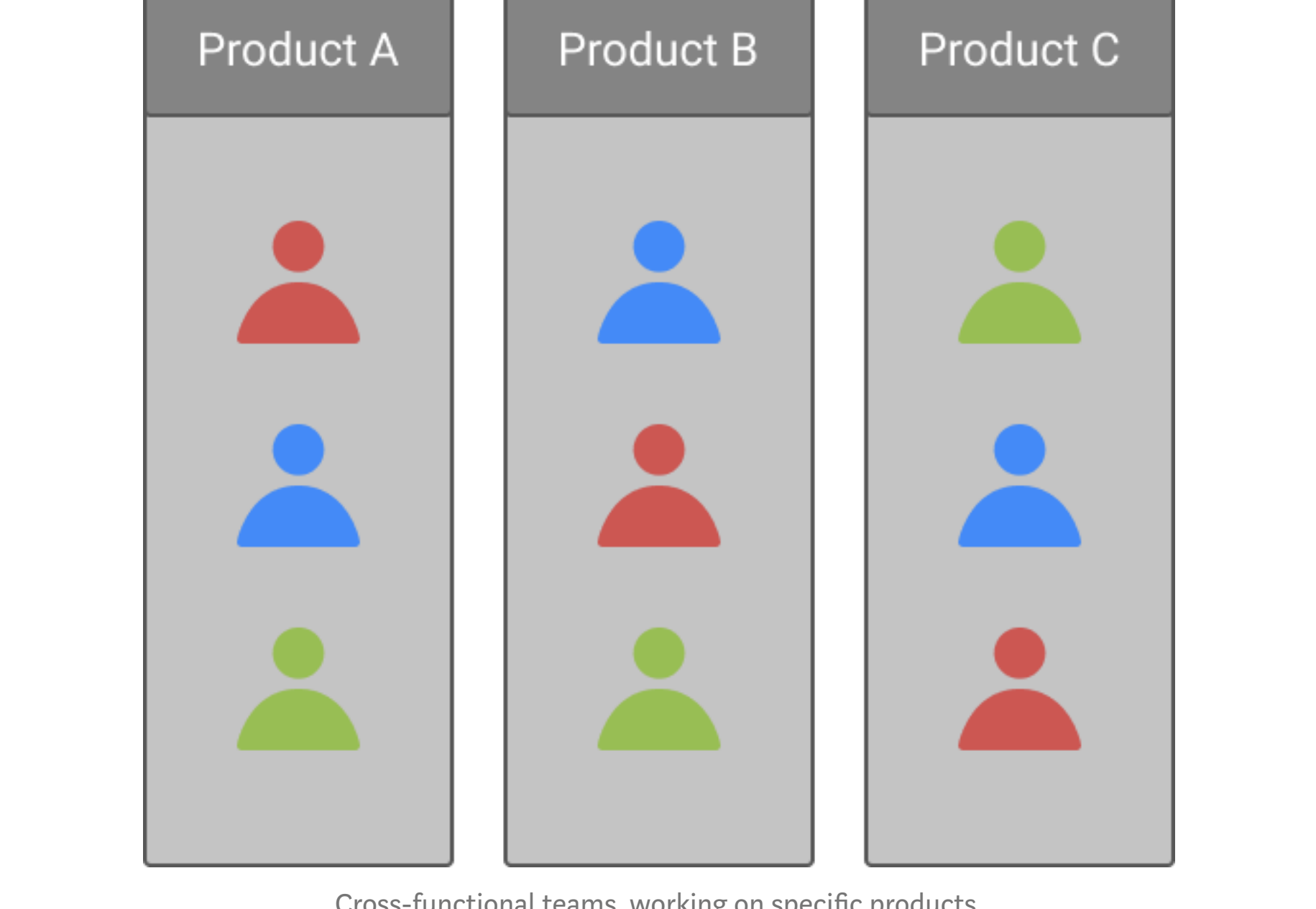


What organisations discovered after years working with this architecture and seeing how much friction is caused by communicating the need for a feature between all those (and many other) layers is that they were missing a metric that is extremely important when delivering a product to market, which is cycle time — the time it takes from when the teams start working on a feature until the delivery.

How could teams reduce external dependencies and synchronisation needs while developing a new product or a new feature?

The answer is to create empowered and autonomous teams, which own the product end-to-end and keep the feedback loop with users tight. The table was turned. With teams containing all of the necessary capabilities to work end-to-end on a product, they should be autonomous to deliver and work on their vision without external dependencies.

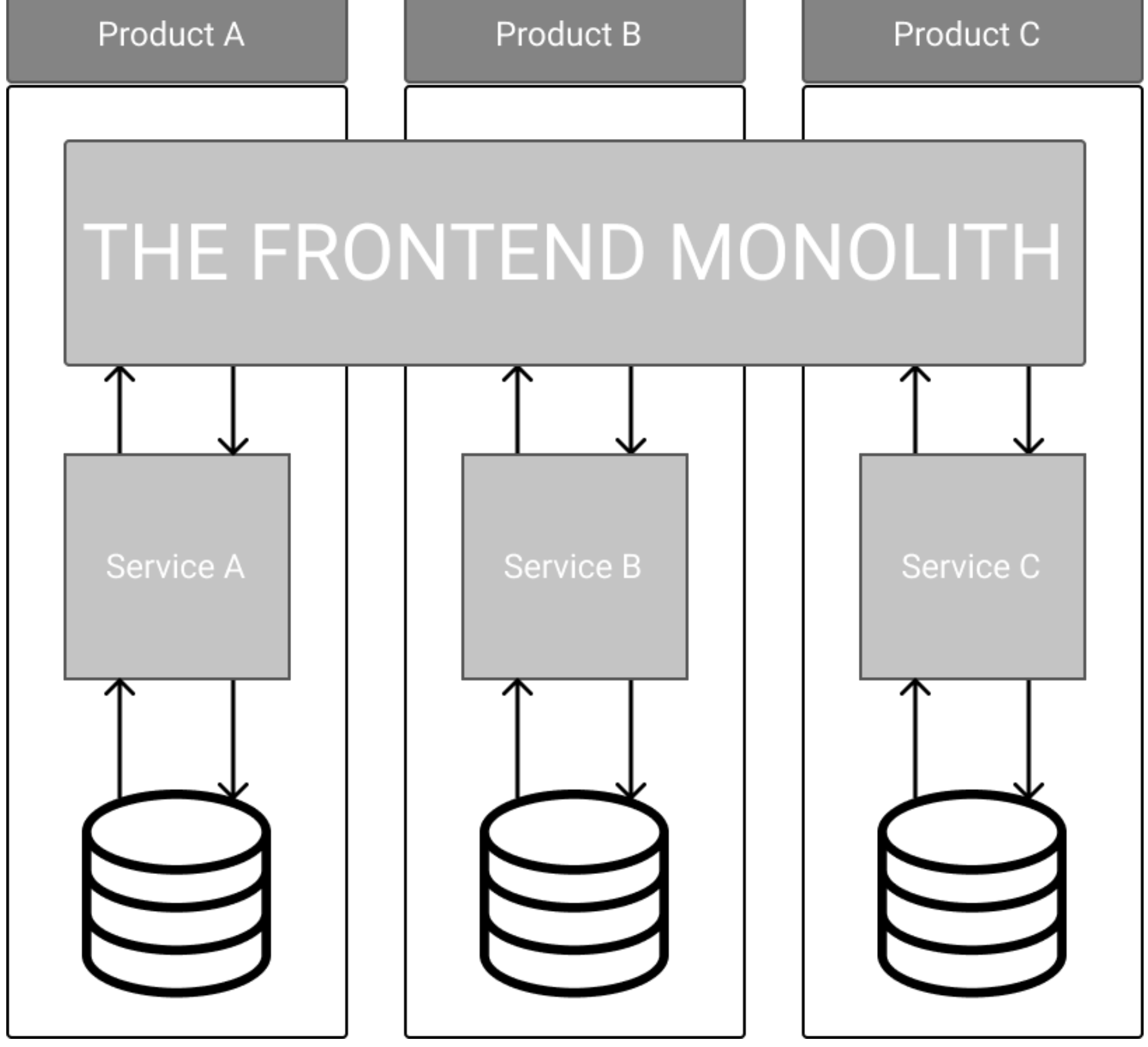
The organisation’s architecture should follow the new communication structure.



The ThoughtWorks team added [microservices](#) for the first time on their **Adopt** column and soon, teams started to work on their **independent evolvability**, decoupling monoliths used by many teams into microservices (the name does not mean a service that is as small as possible, but small enough to fit in one’s hand) decoupling backend services and databases. We got it.

But did we?

An important part of our systems was missing in this decoupling, and it was creating a monster. Teams ended up with well-decoupled backend services, with clear rules and interfaces between them, asynchronous communication channels, well-designed APIs, decoupled databases and CI/CD pipelines, but they kept and grew the **Frontend Monolith**, ending up with the following architecture.



Let’s take a moment to look at this. This is the worst architecture you can have for your organisation. Keeping the frontend monolith, your teams will put all the overheads and all the extra work they need to go through while implementing a microservices-oriented architecture, but end up with none of the benefits. You still have synchronisation points between teams and your teams won’t be able to evolve their systems independently.

What were they — the people that came up with microservices — thinking?

The core idea [about microservices] is that you own everything from the user interface to the data storage so that those teams that own the service can work independently of each other — **Erik Dörnenbug**

A microservice should include all the parts necessary to enable a team to deliver software whenever they feel ready to deliver value to their users. Those teams should also own their interfaces, not just back-end services and databases.

**In short, we really misunderstood microservices.**

*I’m Afonso, a Software Engineer at SumUp. If you’re interested in helping small merchants do what they love, check out our [current engineering vacancies](#).*

Engineering

Microservices

Software Architecture

Software Development

40 claps

🐦

🌐

📘

📌

👤

WRITTEN BY

Afonso Delgado

Software Engineer @ SumUp

Follow

Inside SumUp

Behind the scenes at a global tech company on a mission to empower small businesses.

Follow

Write the first response

## More From Medium

- More on Engineering from Inside SumUp

Cross-functional collaboration in times of uncertainty

Maximilian Stella in Inside SumUp

Apr 30 · 6 min read

21
- More on Engineering from Inside SumUp

International Women’s Day 2020: women in tech leadership at SumUp

Callum Conway in Inside SumUp

Mar 27 · 4 min read

77
- More on Engineering from Inside SumUp

International Women’s Day 2020: women in tech at SumUp

Paul O’Callaghan in Inside SumUp

Mar 9 · 5 min read

41

- Discover Medium

Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. [Watch](#)
- Make Medium yours

Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. [Explore](#)
- Explore your membership

Thank you for being a member of Medium. You get unlimited access to insightful stories from amazing thinkers and storytellers. [Browse](#)