# The Not-So-Straightforward Road From Microservices to Serverless

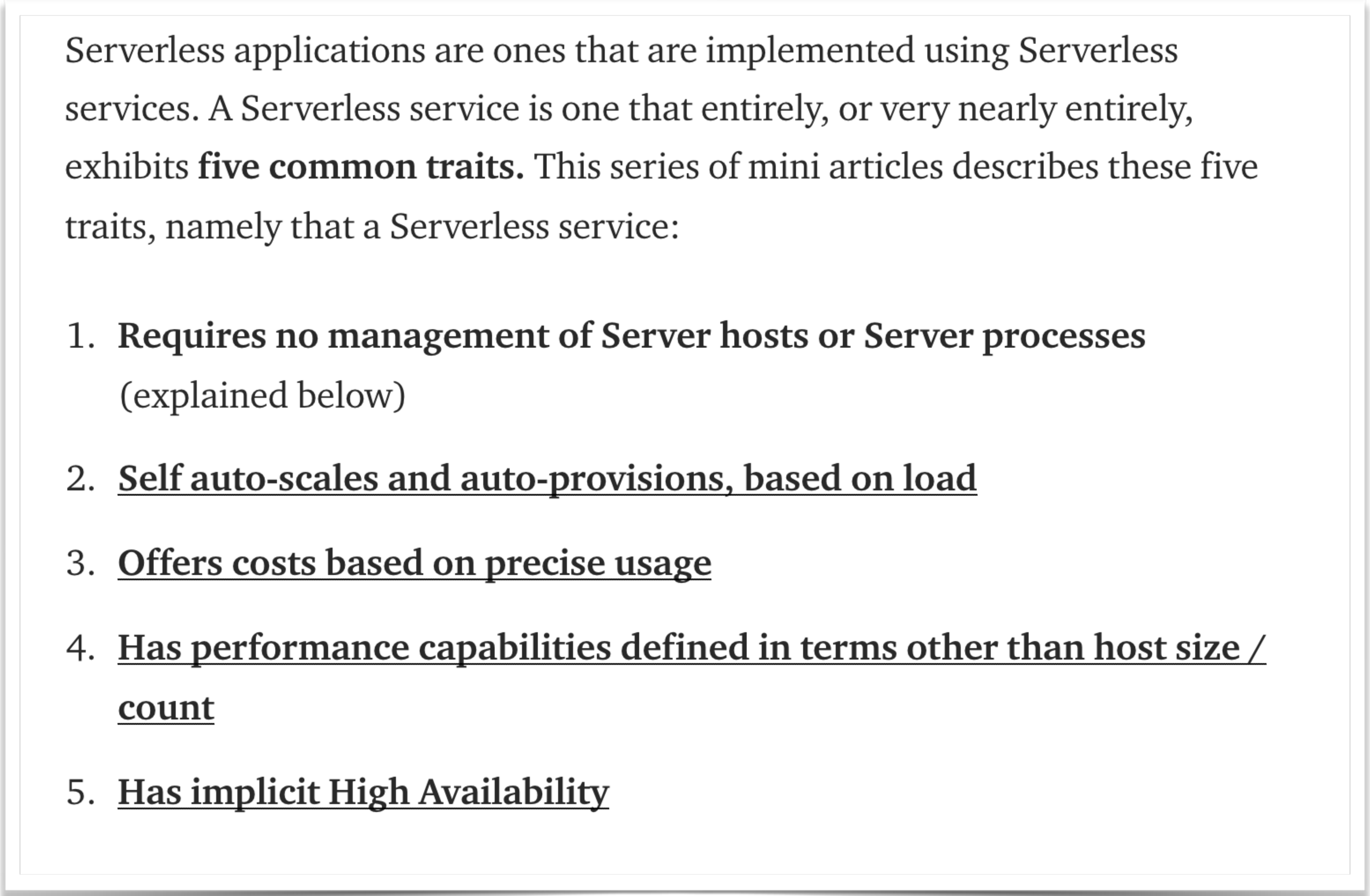
Phil Calçado - @pcalcado - philcalcado.com

# What are microservices?

Microservices seem to be highly-distributed application architectures.

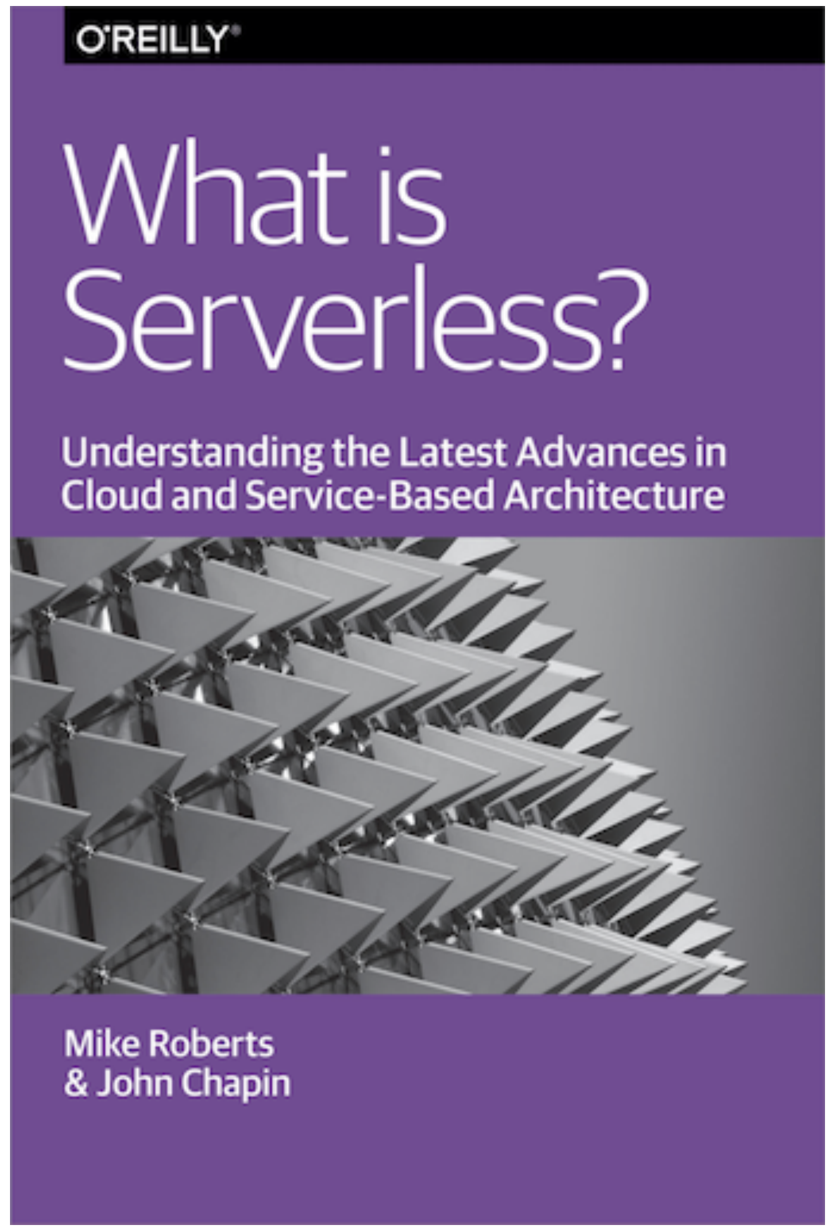
# What is Serverless?

# Who knows? 🤷🏻‍♀️



Serverless applications are ones that are implemented using Serverless services. A Serverless service is one that entirely, or very nearly entirely, exhibits **five common traits.** This series of mini articles describes these five traits, namely that a Serverless service:

1. **Requires no management of Server hosts or Server processes** (explained below)
2. **Self auto-scales and auto-provisions, based on load**
3. **Offers costs based on precise usage**
4. **Has performance capabilities defined in terms other than host size / count**
5. **Has implicit High Availability**



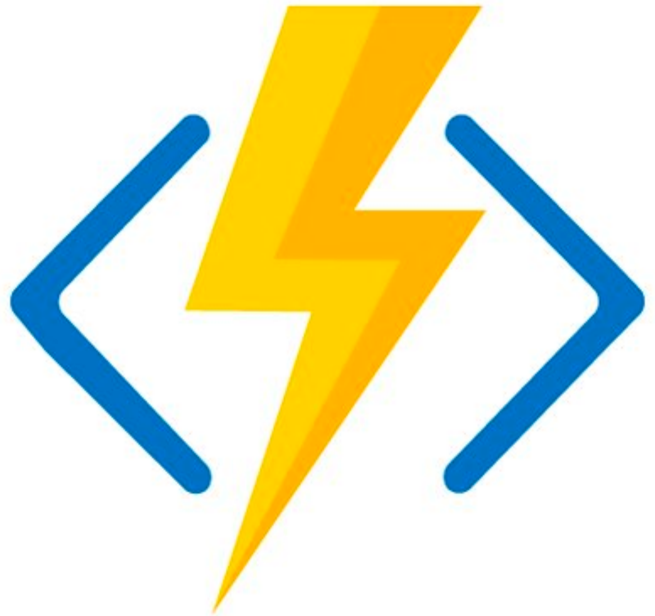
O'REILLY

What is Serverless?

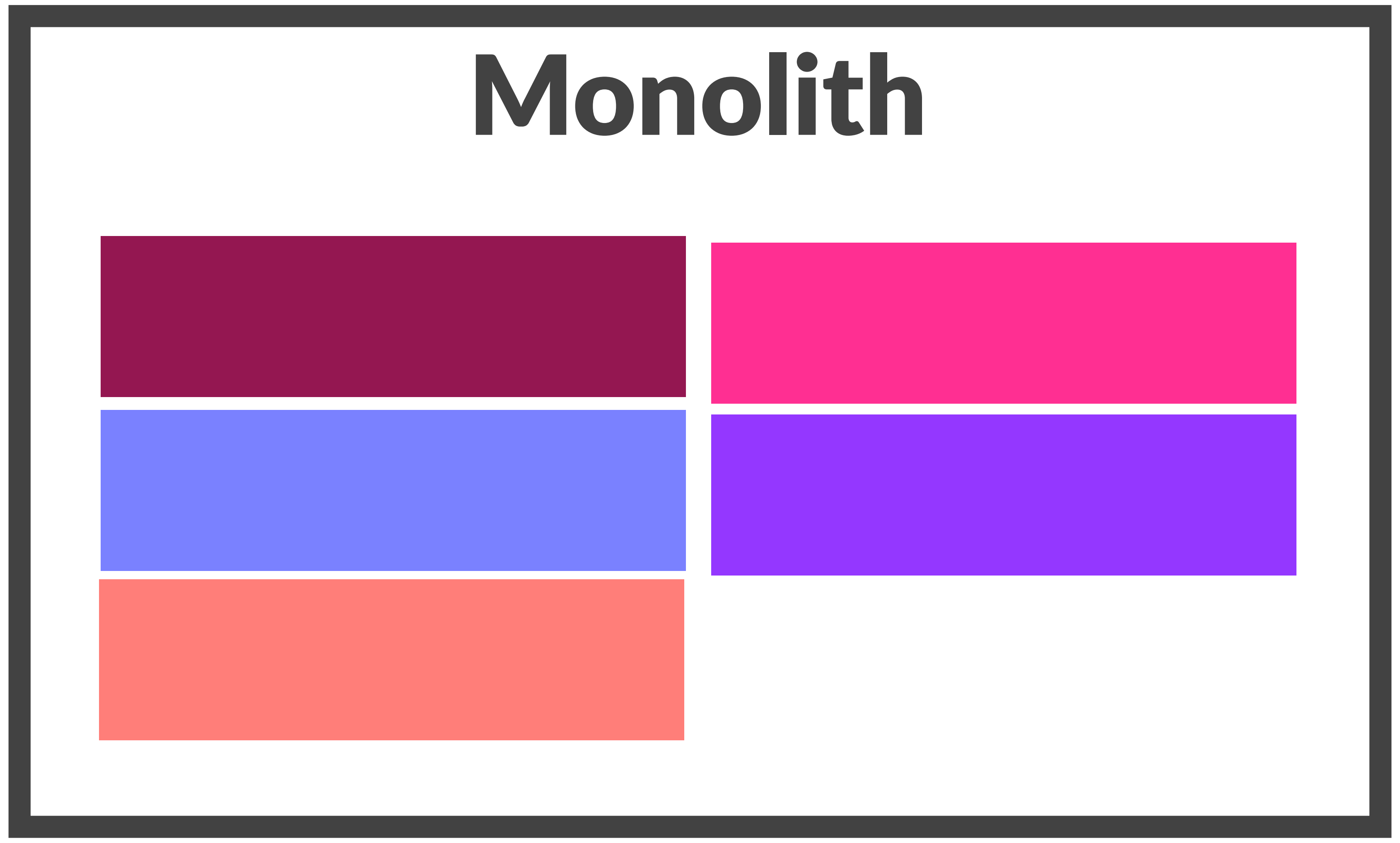
Understanding the Latest Advances in Cloud and Service-Based Architecture

Mike Roberts & John Chapin

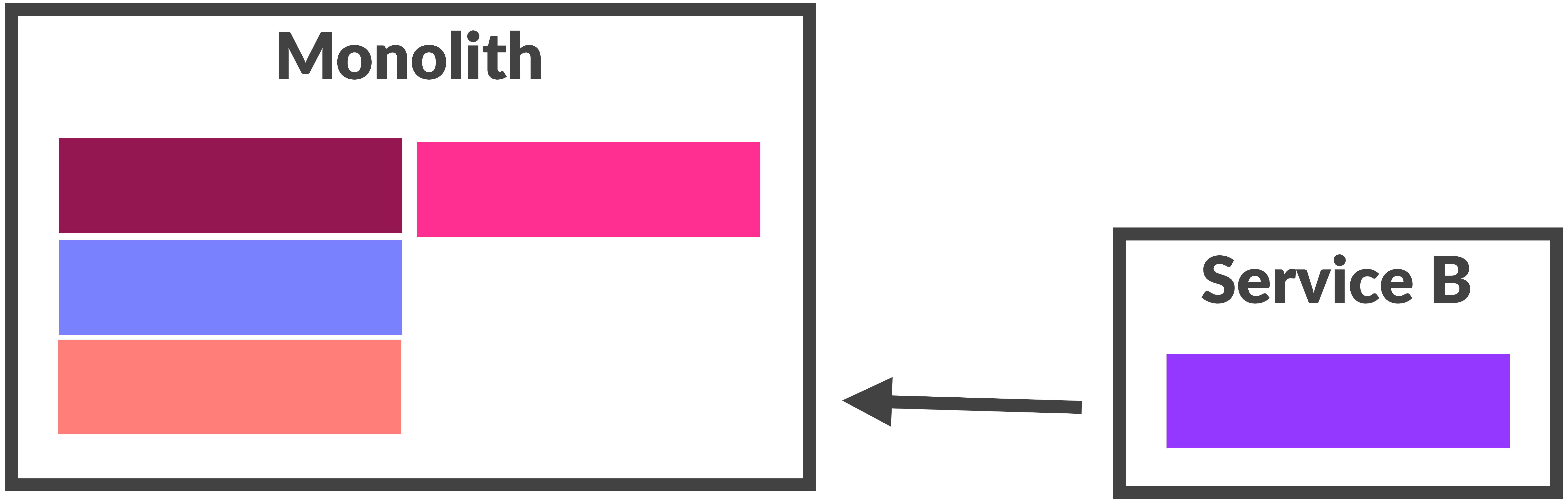
# e.g.

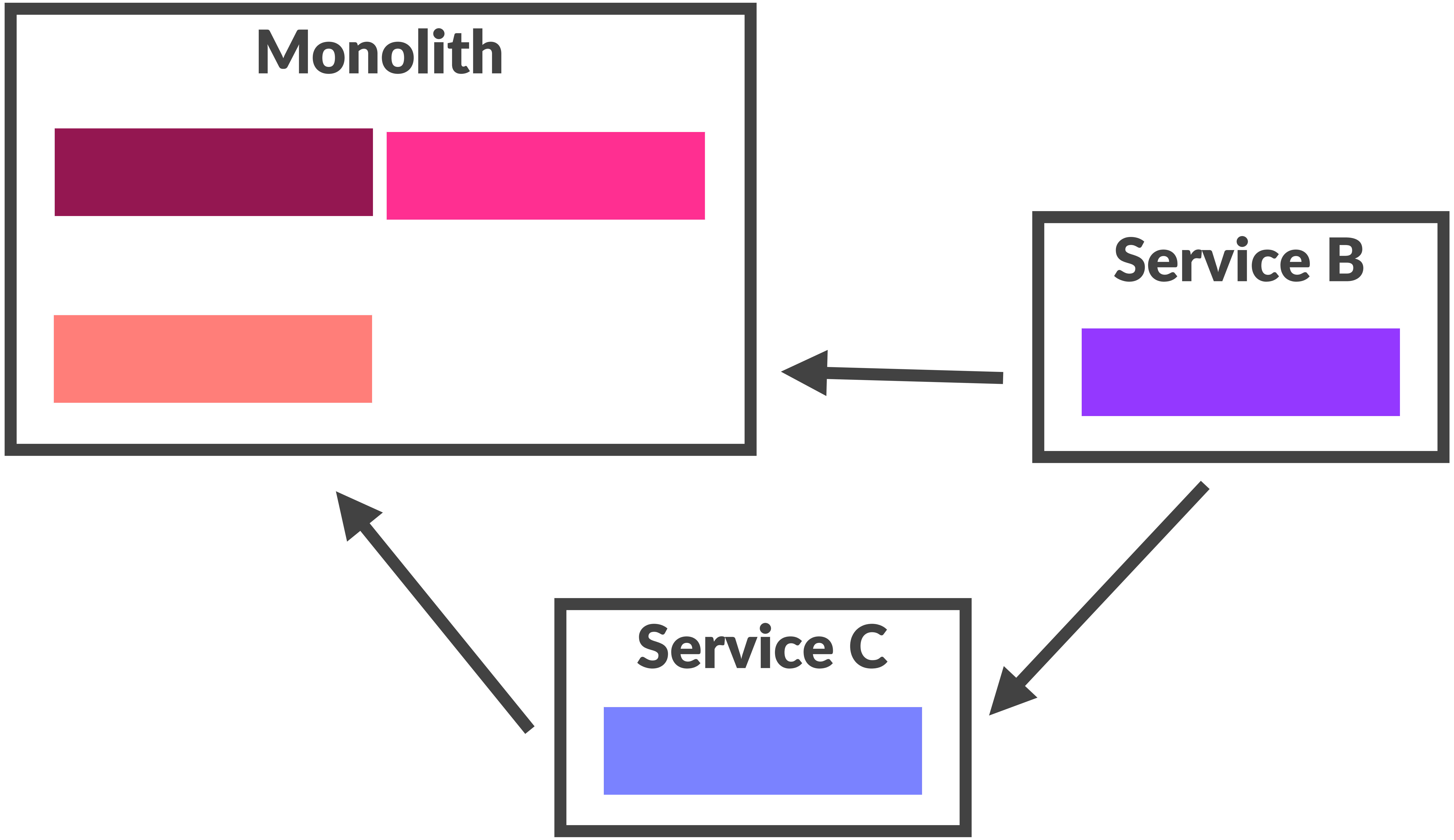


# Keep your monolith for as long as possible



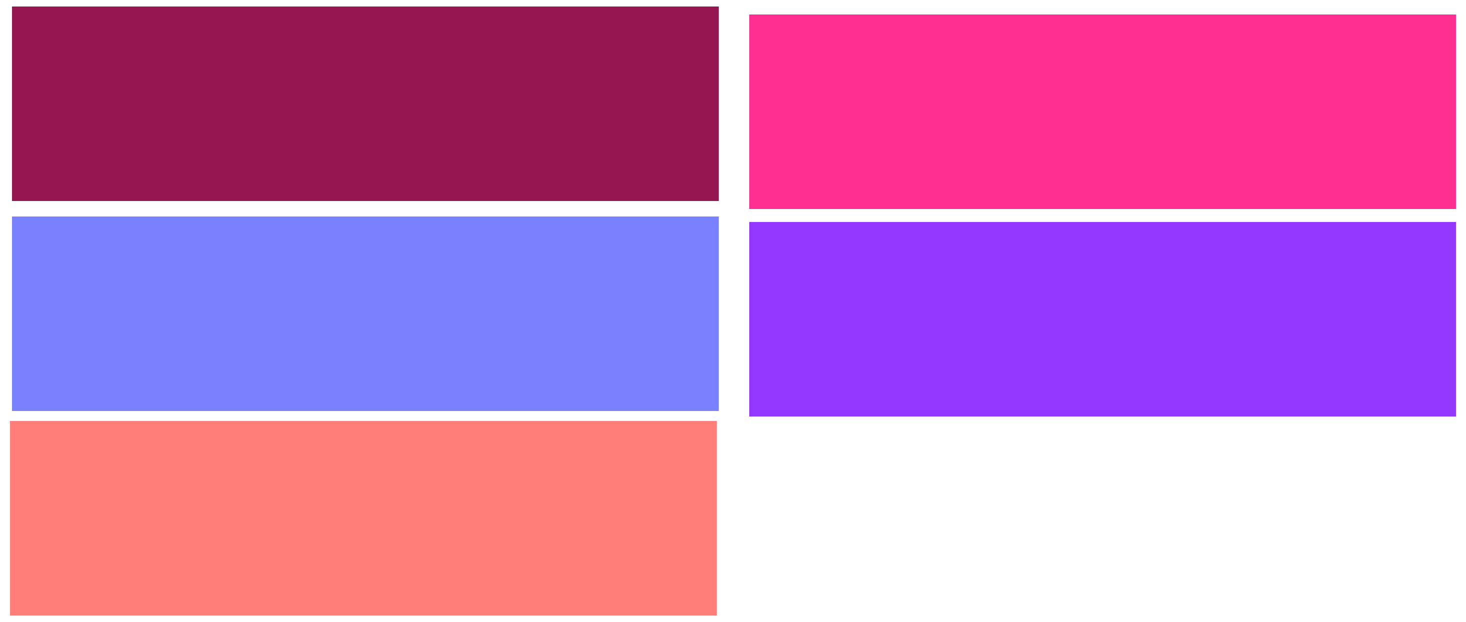
Monolith

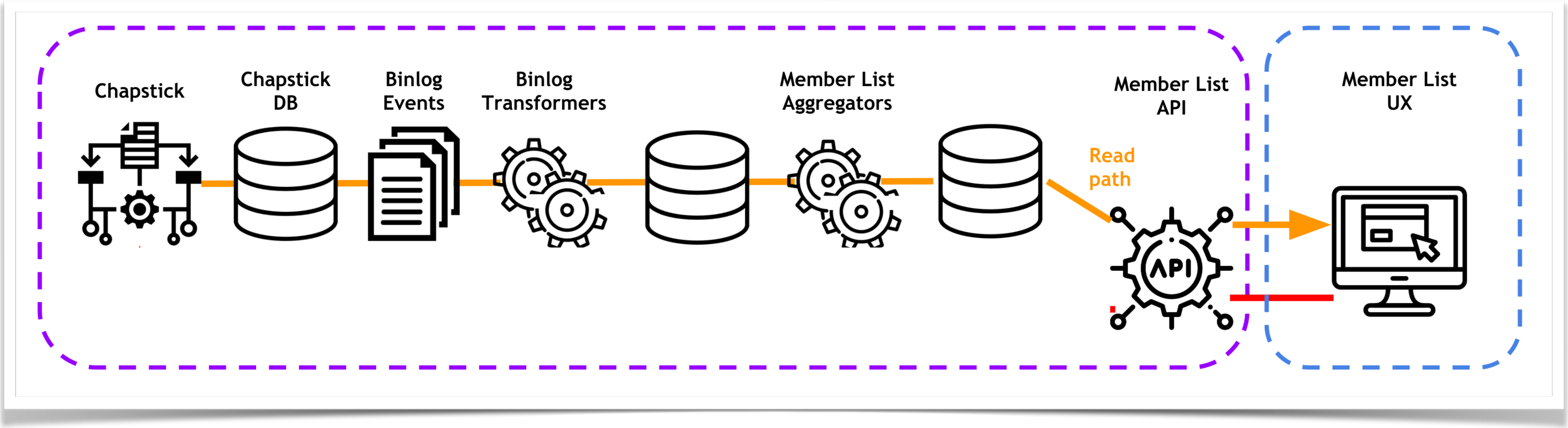

Monolith
Service B


Monolith
Service B
Service C

But what if you waited so long ~~microservices aren’t hype anymore~~ there are other alternatives?

# “Chapstick"



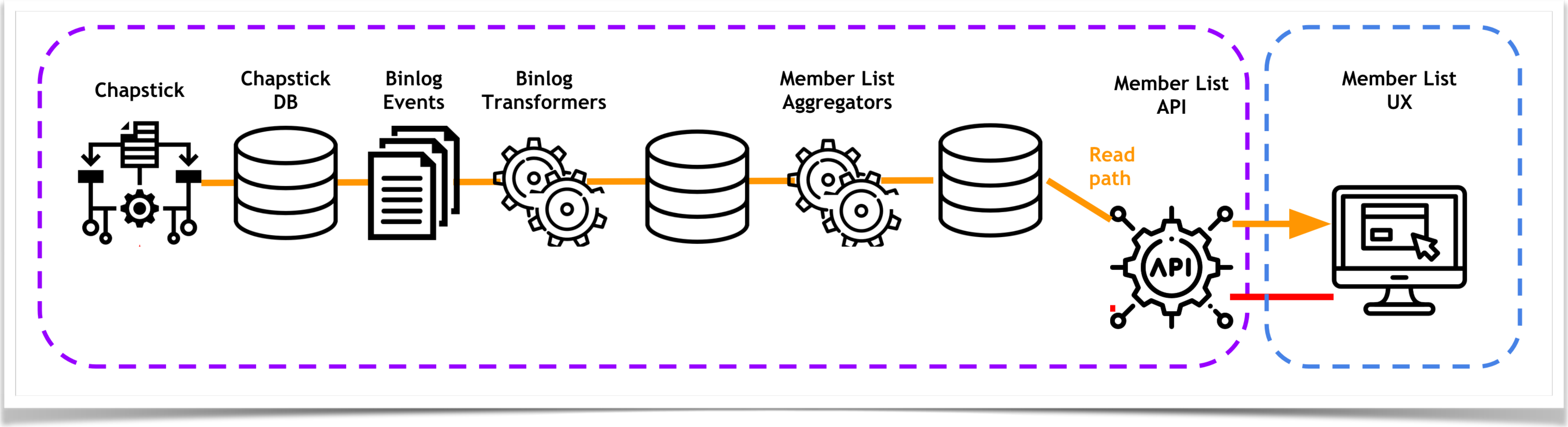

Chapstick
Chapstick DB
Binlog Events
Binlog Transformers
Member List Aggregators
Member List API
Read path
API
Member List UX

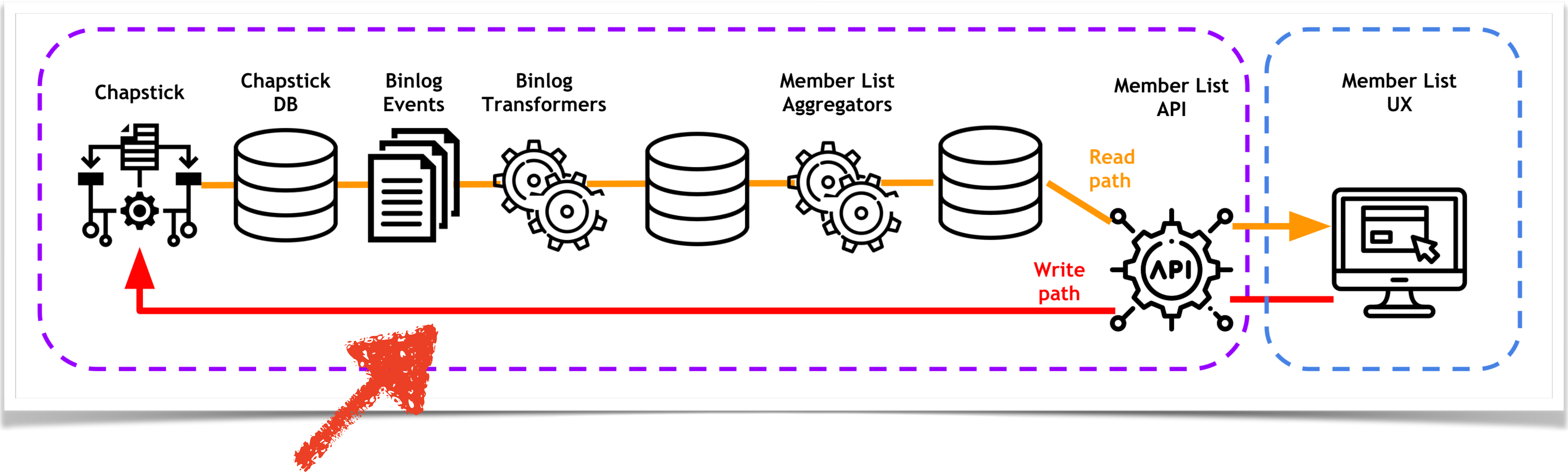
This project wasn’t particularly successful

# Some challenges we faced

Solving a bug means re-processing all the data

# “The write path issue”




Chapstick
Chapstick DB
Binlog Events
Binlog Transformers
Member List Aggregators
Member List API
Member List UX
Read path
Write path

Write path

# Cold start for JVM Lambdas

# DynamoDB Costs

Who owns what? What calls what? Is this Lambda supposed to be in production?

# Some of the challenges we DID NOT face

# Getting new engineers productive on Lambda/ DynamoDB

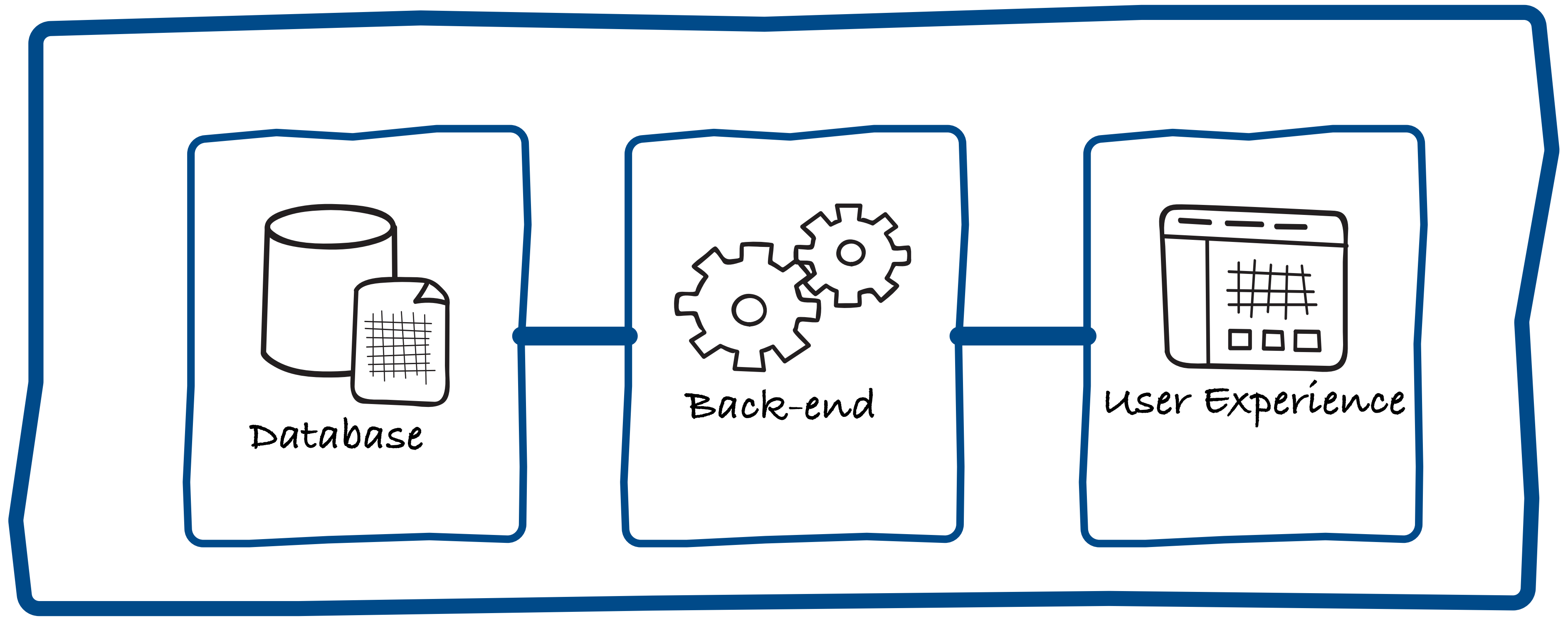
# Cold start for Node.js Lambdas

# Operations allergy

# "Local" development

# Developer happiness

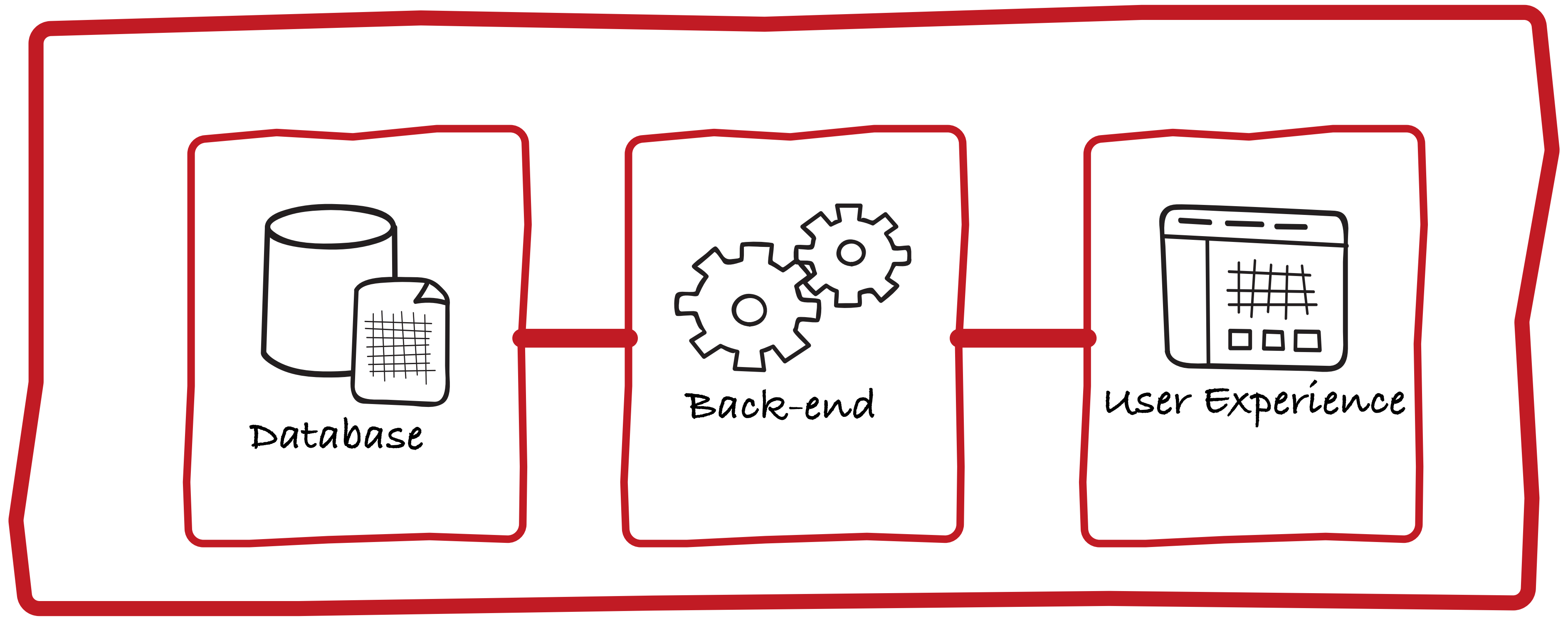
# How can we keep the good and get rid of the bad?


Database

Back-end

User Experience

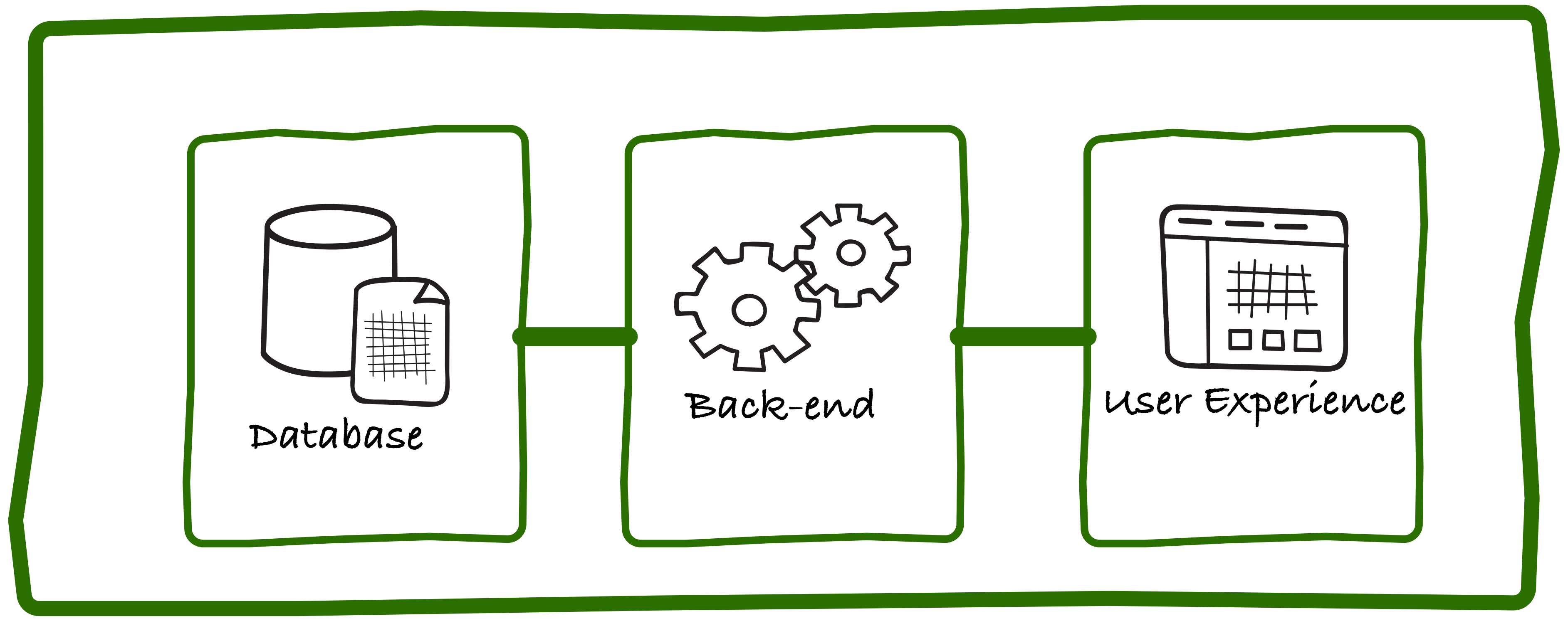
Financial Reporting


Database

Back-end

User Experience

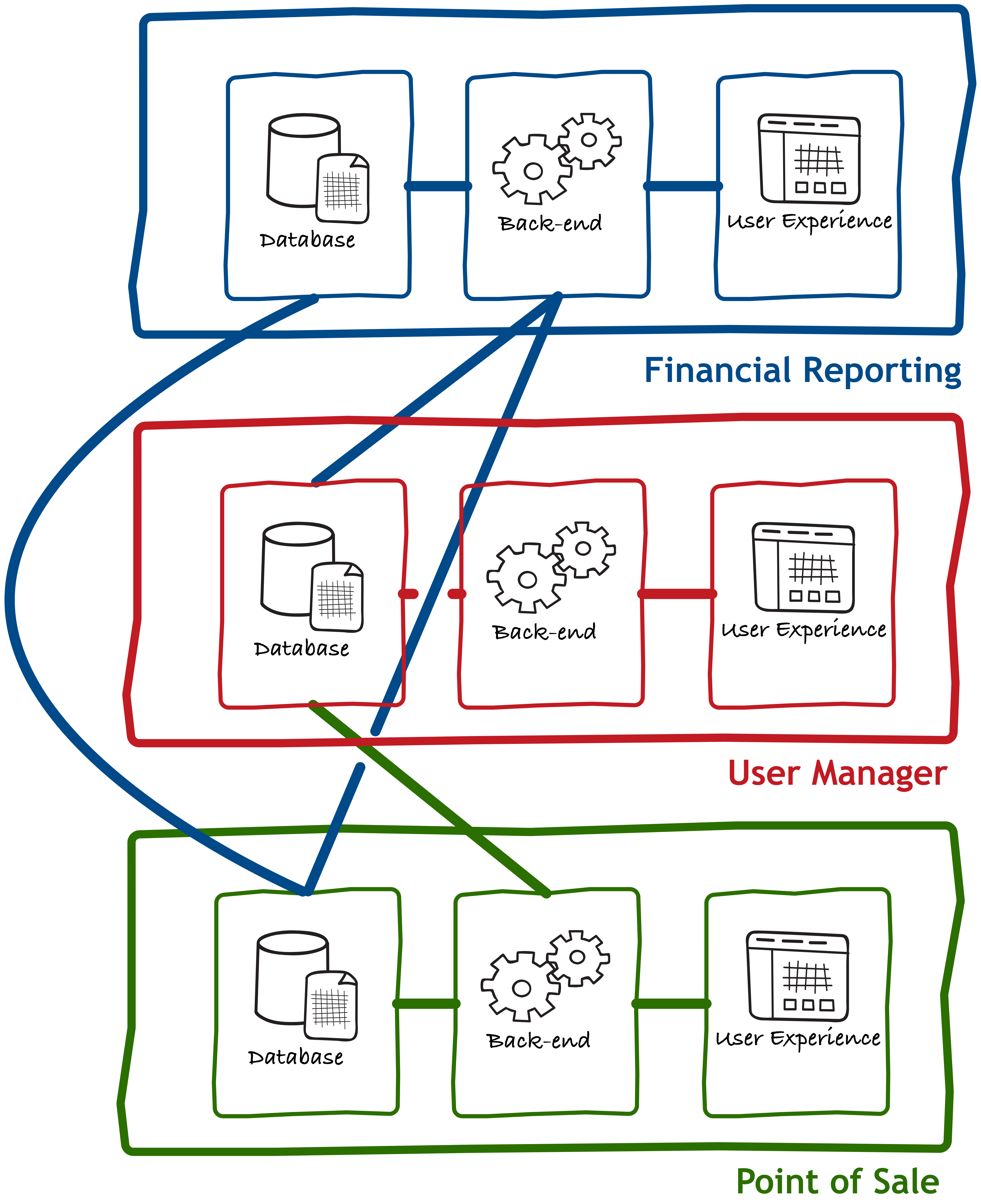
User Manager


Database

Back-end

User Experience

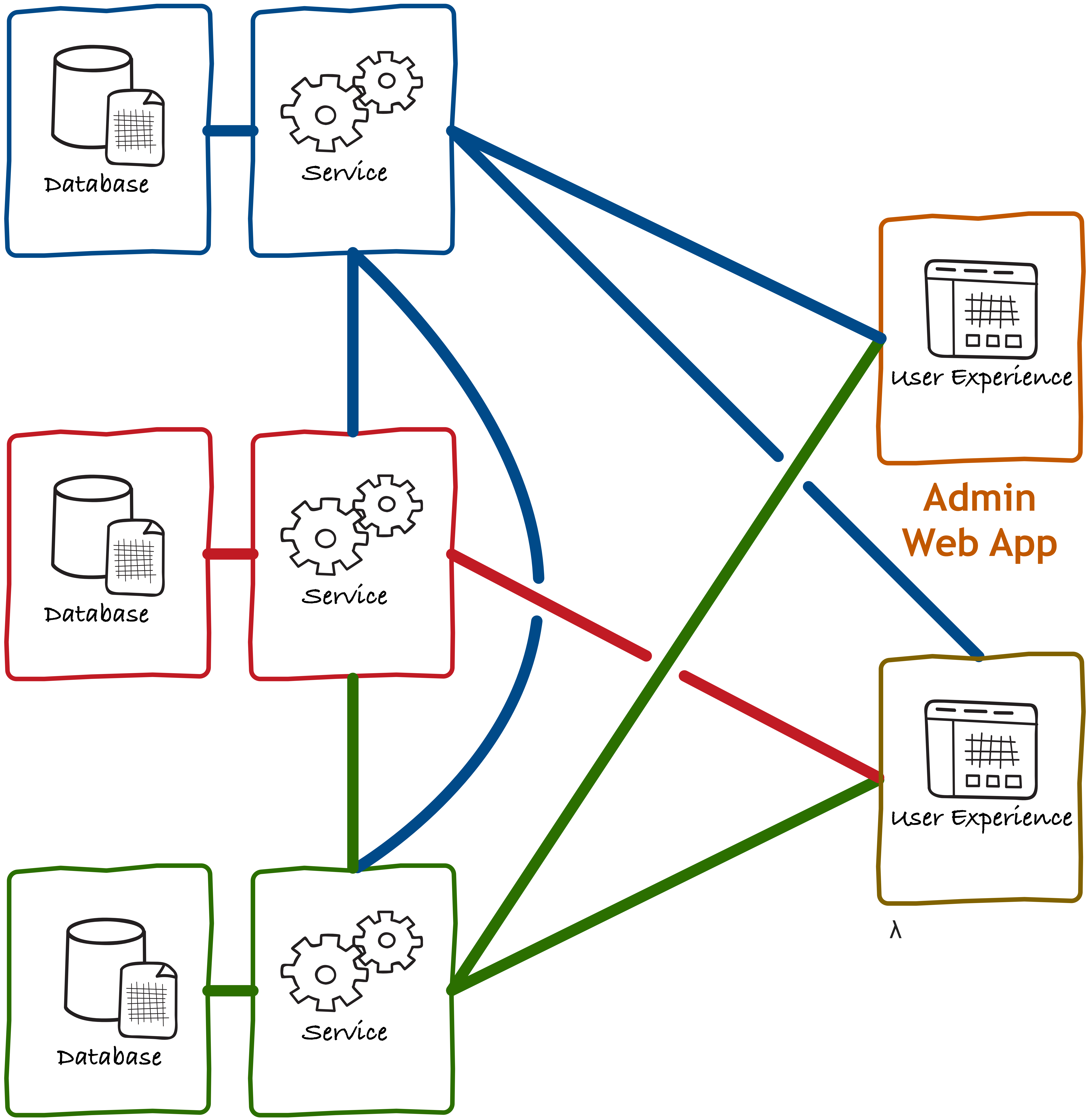
Point of Sale



Financial Reporting

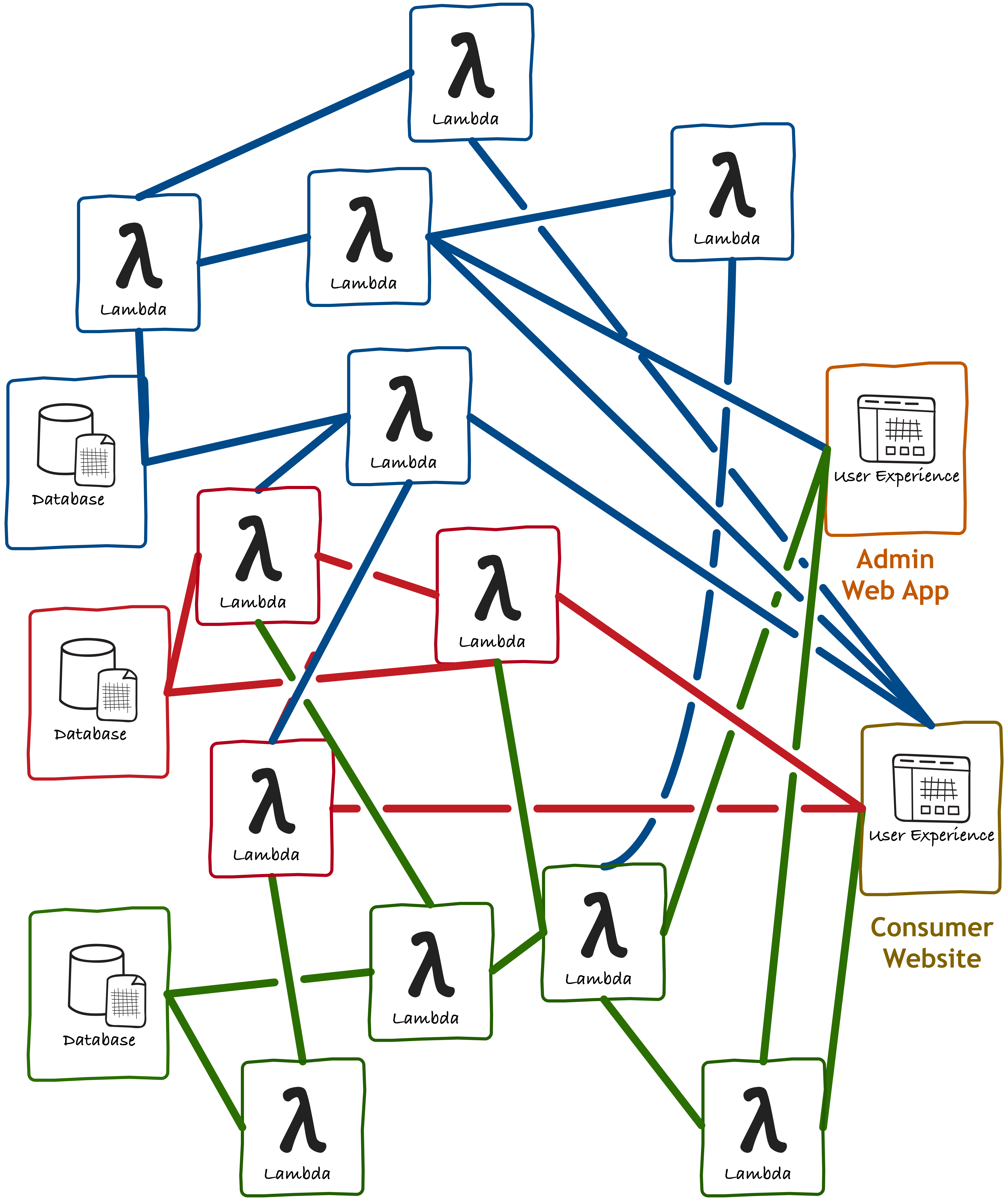
User Manager

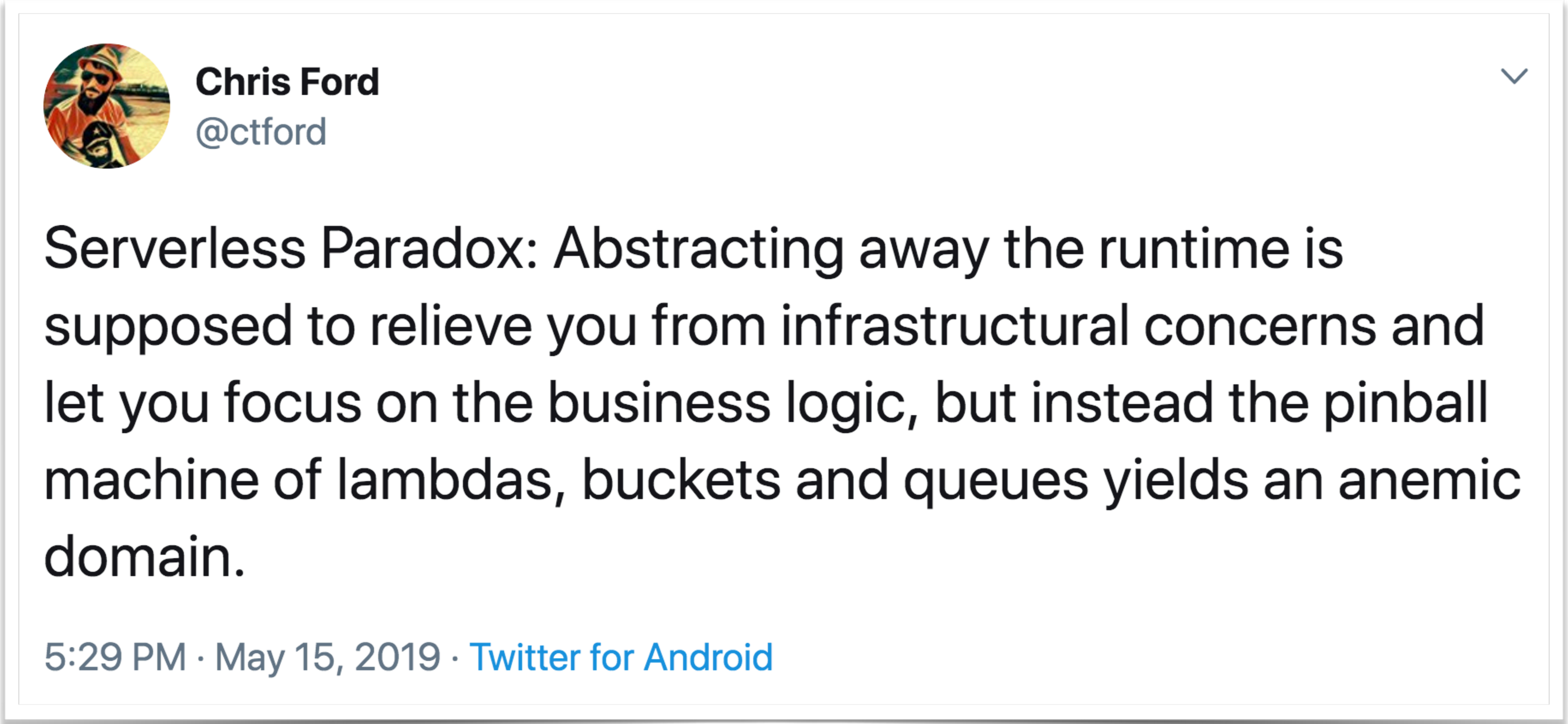
Point of Sale



Admin
Web App

λ


Lambda
Lambda
Lambda
Lambda
Database
Lambda
User Experience
Admin Web App
Lambda
Lambda
Database
Lambda
User Experience
Consumer Website
Database
Lambda
Lambda
Lambda
Lambda

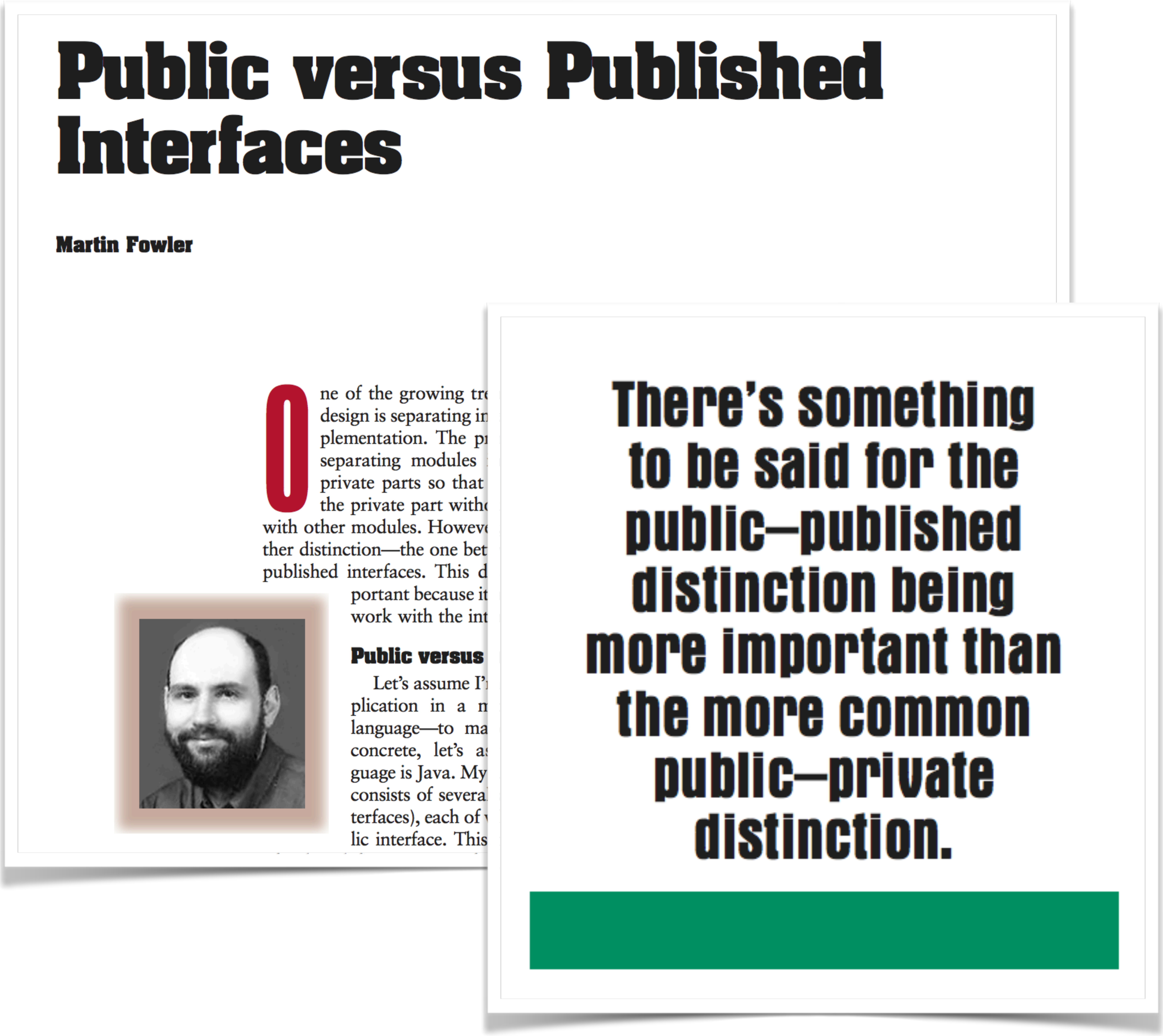


Chris Ford
@ctford

Serverless Paradox: Abstracting away the runtime is supposed to relieve you from infrastructural concerns and let you focus on the business logic, but instead the pinball machine of lambdas, buckets and queues yields an anemic domain.

5:29 PM · May 15, 2019 · Twitter for Android

# We were suffering from Pinball machine Architecture



# Public versus Published Interfaces

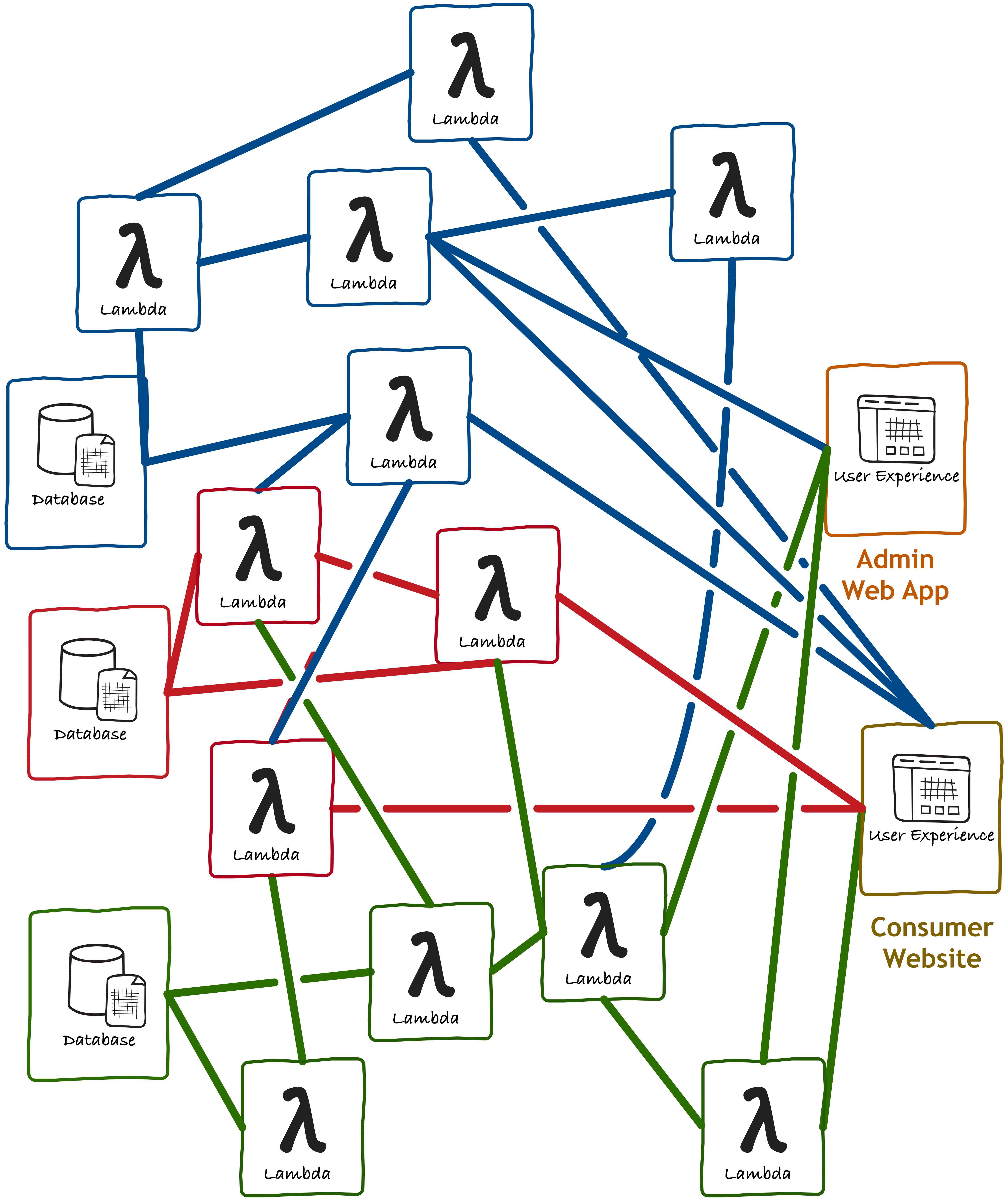
Martin Fowler

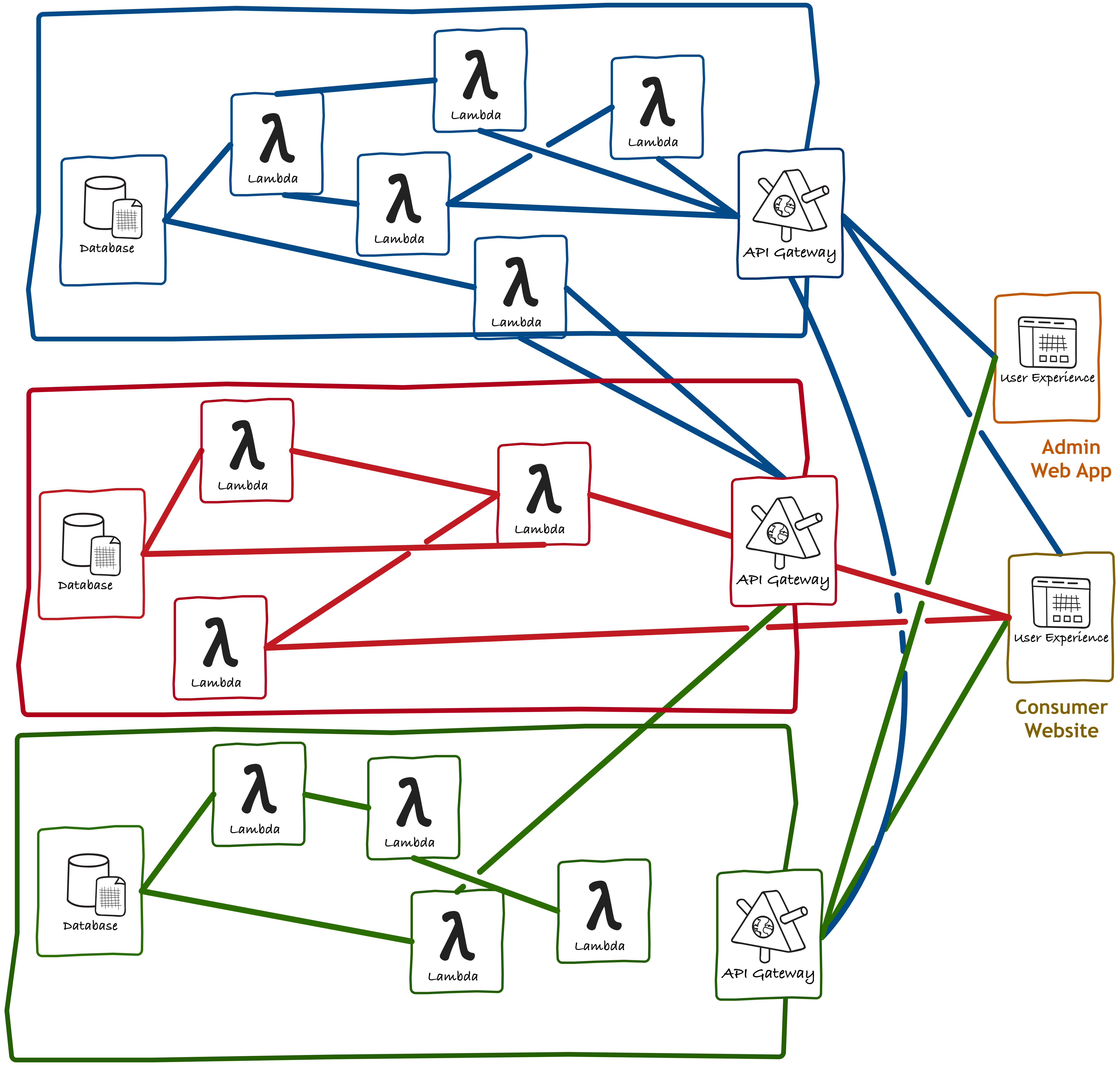
One of the growing tr
design is separating in
plementation. The pr
separating modules
private parts so that
the private part witho
with other modules. Howev
ther distinction—the one bet
published interfaces. This d
portant because it
work with the int

**Public versus**

Let's assume I'
plication in a m
language—to ma
concrete, let's a
guage is Java. My
consists of severa
terfaces), each of
lic interface. This

**There's something to be said for the public–published distinction being more important than the more common public–private distinction.**


Lambda
Lambda
Lambda
Lambda
Database
Lambda
User Experience
Admin Web App
Lambda
Lambda
Database
Lambda
User Experience
Consumer Website
Database
Lambda
Lambda
Lambda
Lambda


Lambda
Lambda
Lambda
Database
Lambda
Lambda
API Gateway
User Experience
Admin
Web App
Lambda
Lambda
Database
Lambda
API Gateway
User Experience
Consumer
Website
Lambda
Lambda
Database
Lambda
Lambda
API Gateway

# Use Serverless as building blocks for Microservices

# AWS Accounts are a great way to define service boundaries

API Gateway is actually quite expensive (at the moment)

~1/10 of the engineering teams was dedicated to platform

# But… is this really Serverless?

# Who cares? 🤷‍♀️

You can’t got from 2/10 to 10/10 in one jump.

Serverless looks like the future, but we’re not there yet

# Questions?