EXPLORE DDD
CONFERENCE

Bounded context is not enough!

github.com/BrewUp/DDD-Explore-2024

Disclaimer

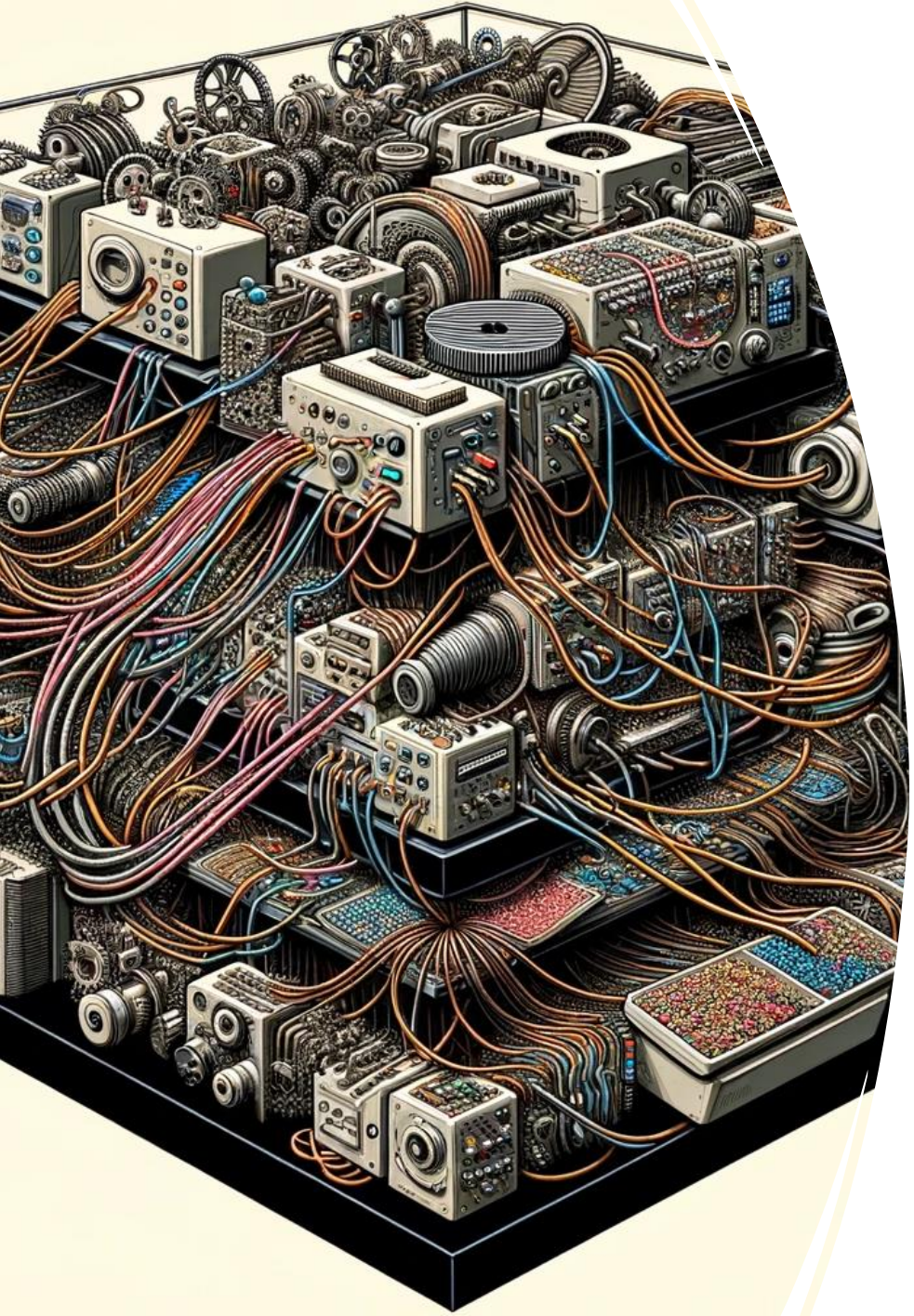
The "Bounded context" is needed, but in our opinion is not just a business transactional boundary anymore. We must keep in consideration its evolution and the static coupling with the infrastructure.

Awful monolith

Show me the code

Big ball of mud

Lack of a clear architecture, leading to a system that is haphazardly structured and difficult to understand or modify.



Lack of modularity

Failure to properly encapsulate different functionalities, leading to a system where changes in one module ripple through others.

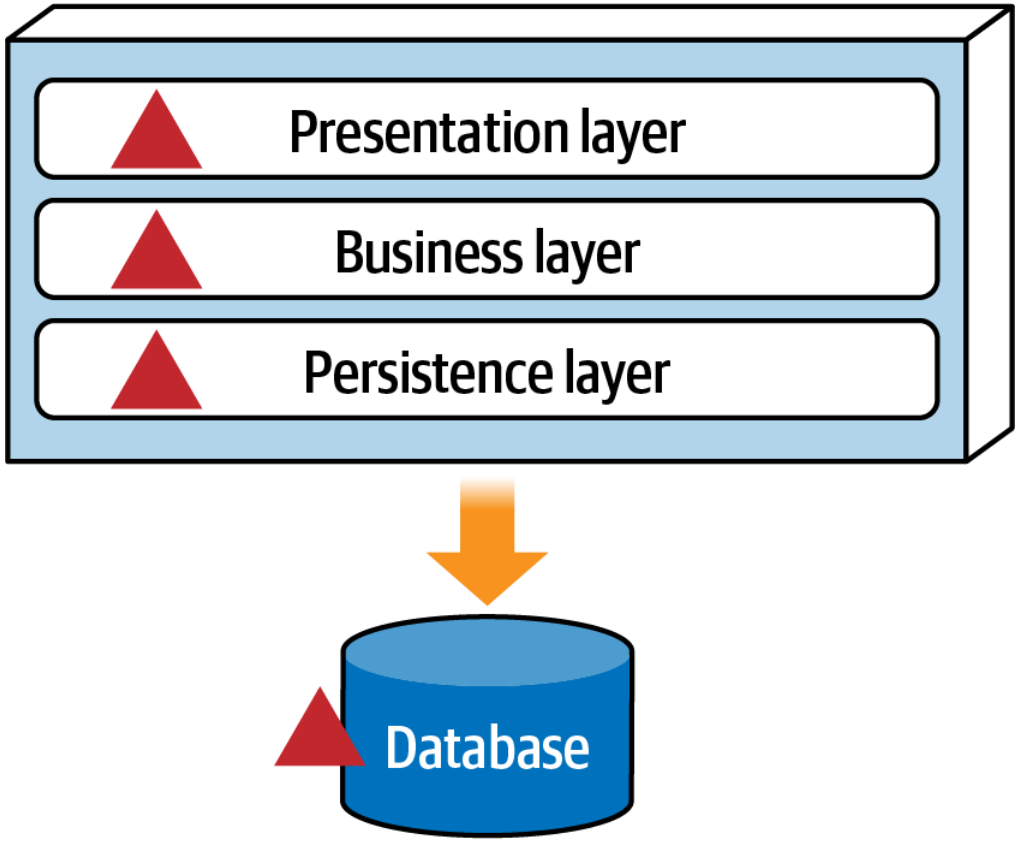
Shared data model

Different modules or functionalities directly accessing and modifying a shared data model, leading to high risk of data corruption and conflicts.

Monolithic database design

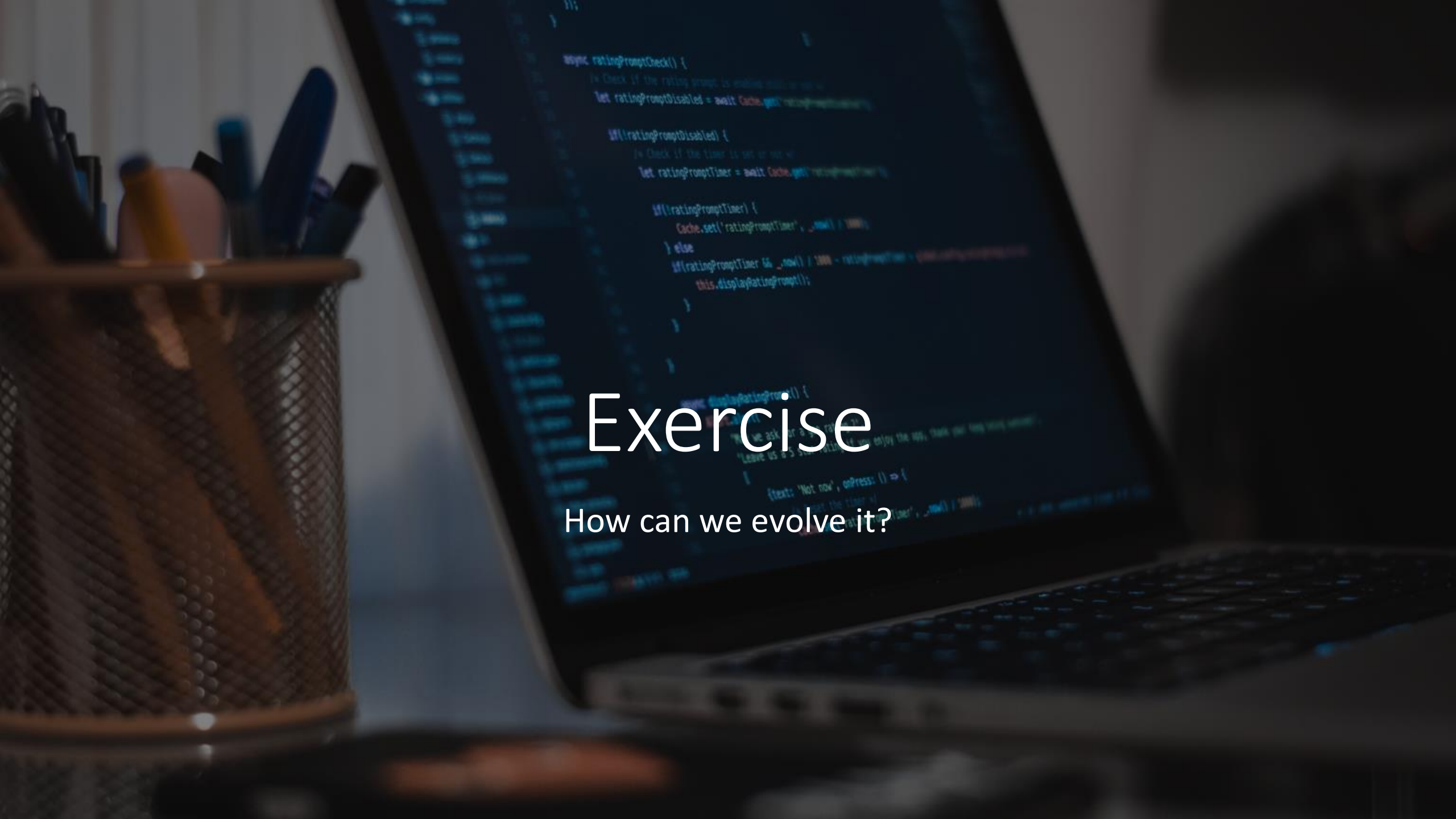
A single database for all application needs, creating a bottleneck and complicating any attempt to scale or separate concerns.

What is the impact of changes?



Application-level change scope

(Triangle represents where change occurs)



Exercise

How can we evolve it?

“Developers are drawn to complexity, like moths to a flame, often with the same outcome” - Neal Ford

Architecture decisions

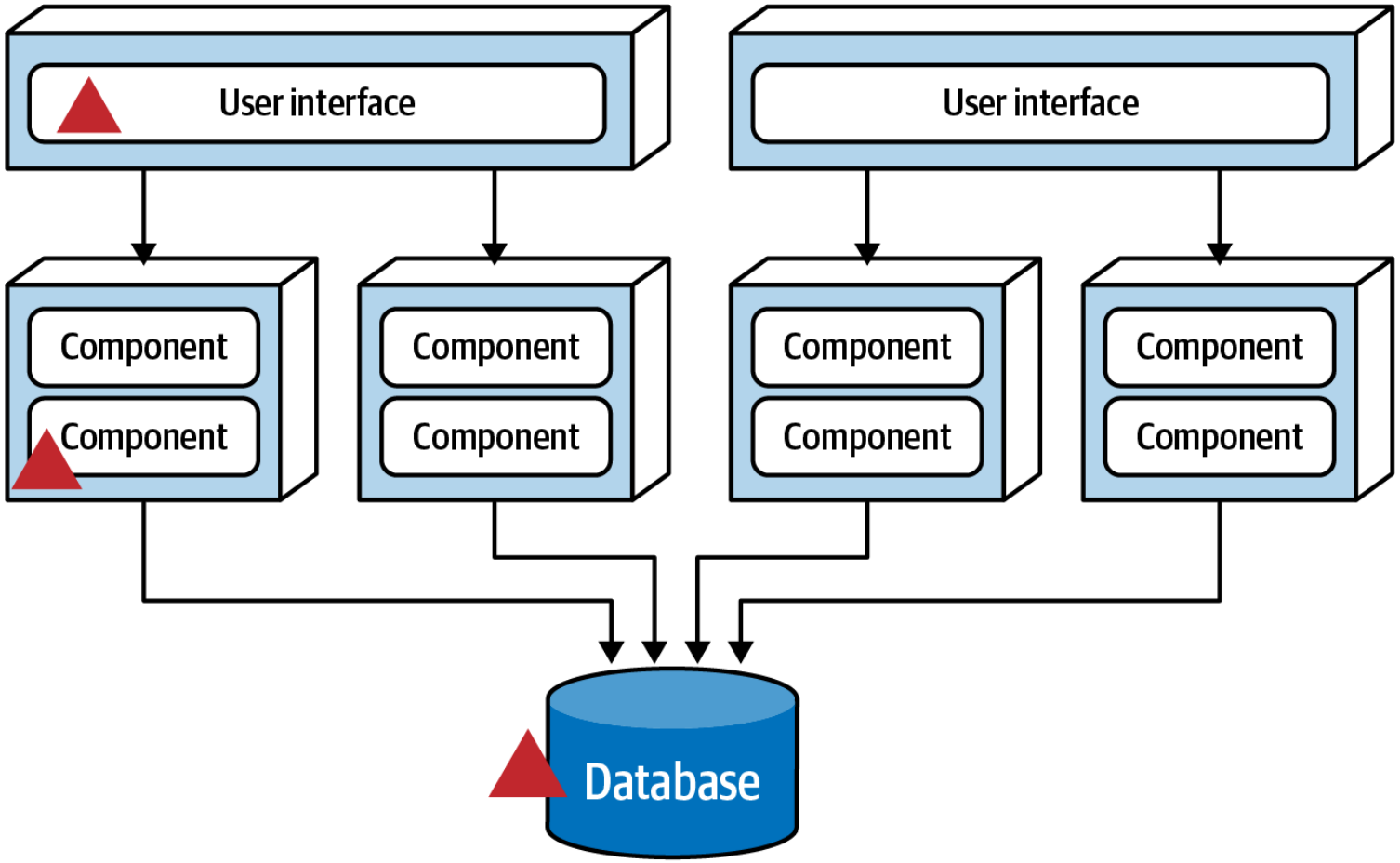
We must understand the principles/key concepts to make effective decisions

- **Service**: a cohesive collection of functionality deployed as an independent executable
- **Coupling**: a change in one service might require a change in an another to maintain proper functionality
- **Component**: building block of the application that does some sort of business or infrastructure function (e.g.: namespace or package). For example, the component Order History might be implemented through a set of class files located in the namespace
- **Synchronous communication**: the caller wait for the response before proceeding
- **Asynchronous communication**: the caller does not wait for the response
- **Orchestrated coordination**: it includes a service that coordinate the workflow
- **Choreographed coordination**: it lacks an orchestrator
- **Atomicity**: A workflow is atomic if all parts maintain a consistent state at all times (the opposite is eventual consistency)
- **Contract**: the interface between two software parts

Modular (good) monolith

Show me the code

What is the impact of changes?



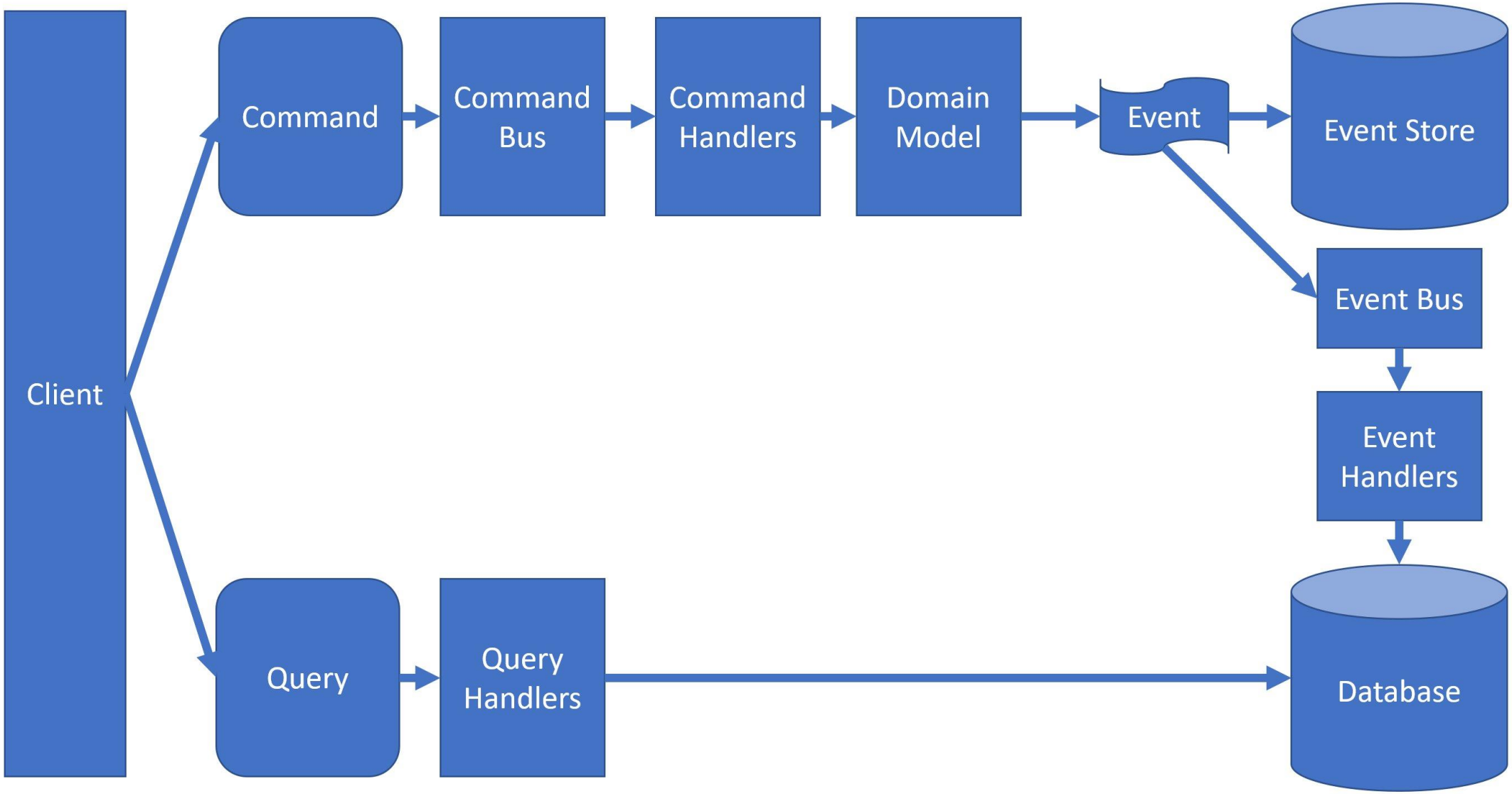
Domain-level change scope

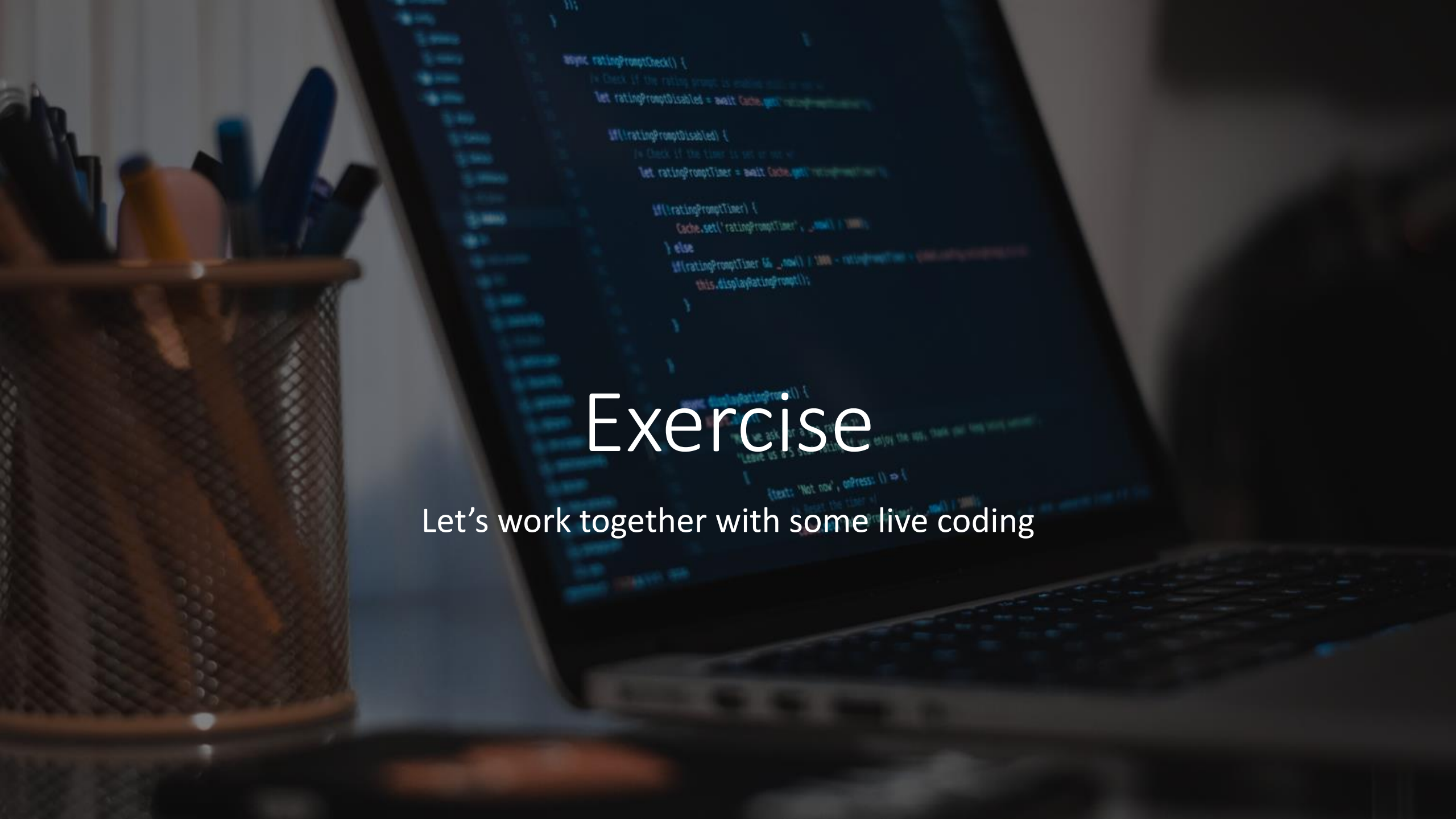
(Triangle represents where change occurs)

What we are missing?

What we can add to make modules fully independent?

CQRS+ES


Client
Command
Command Bus
Command Handlers
Domain Model
Event
Event Store
Event Bus
Event Handlers
Query
Query Handlers
Database



Exercise

Let’s work together with some live coding

Modular monolith with events

Show me the code

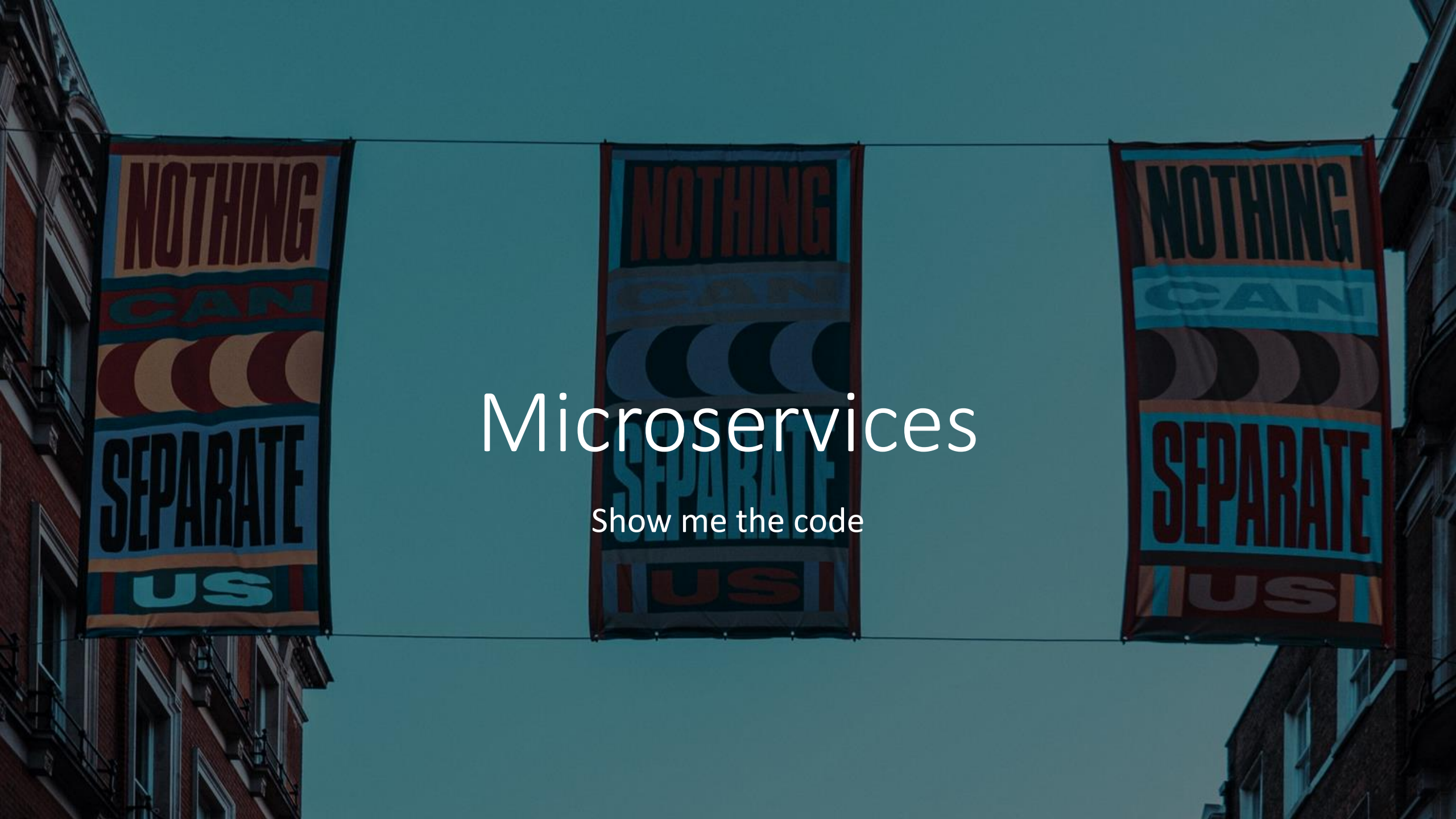
Fitness functions

The book «Building Evolutionary Architectures» defined the concept of an architectural fitness function

“any mechanism that performs an objective integrity assessment of some architecture characteristics or combination of architecture characteristics.”

Given, When, Then

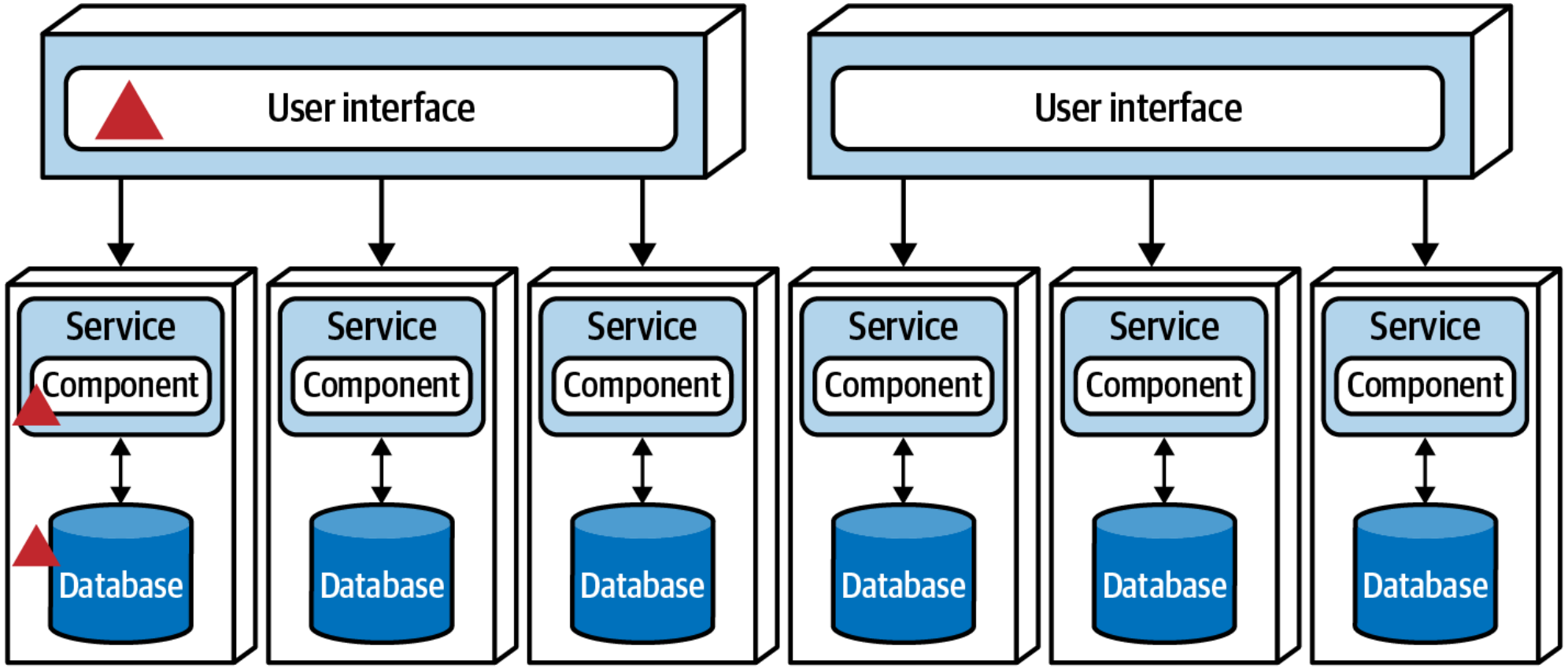
- The essential idea is to divide a scenario into three sections:
- **Given**: describes the state of the aggregate before sending the command. A kind of prerequisite of the test.
- **When**: represents the command to be sent to the aggregate.
- **Then/Expect**: describes the state changes expected as a result of the command.



Microservices

Show me the code

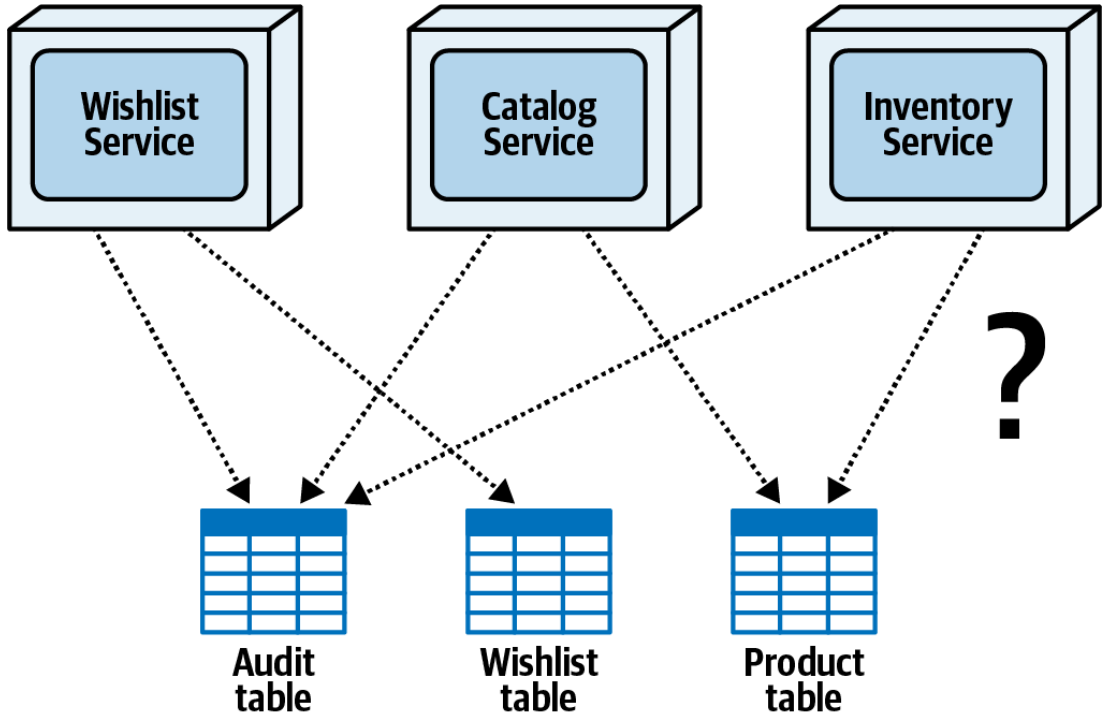
What is the impact of changes?



Function-level change scope

(Triangle represents where change occurs)

Well…How do I split my monolithic DB?



Common table ownership

Single table ownership

Joint ownership

Table split
Data domain
Delegate

“Software architecture it's abstract by nature and we must ground it with some implementation details to make it concrete”

Architecture quantum|quanta

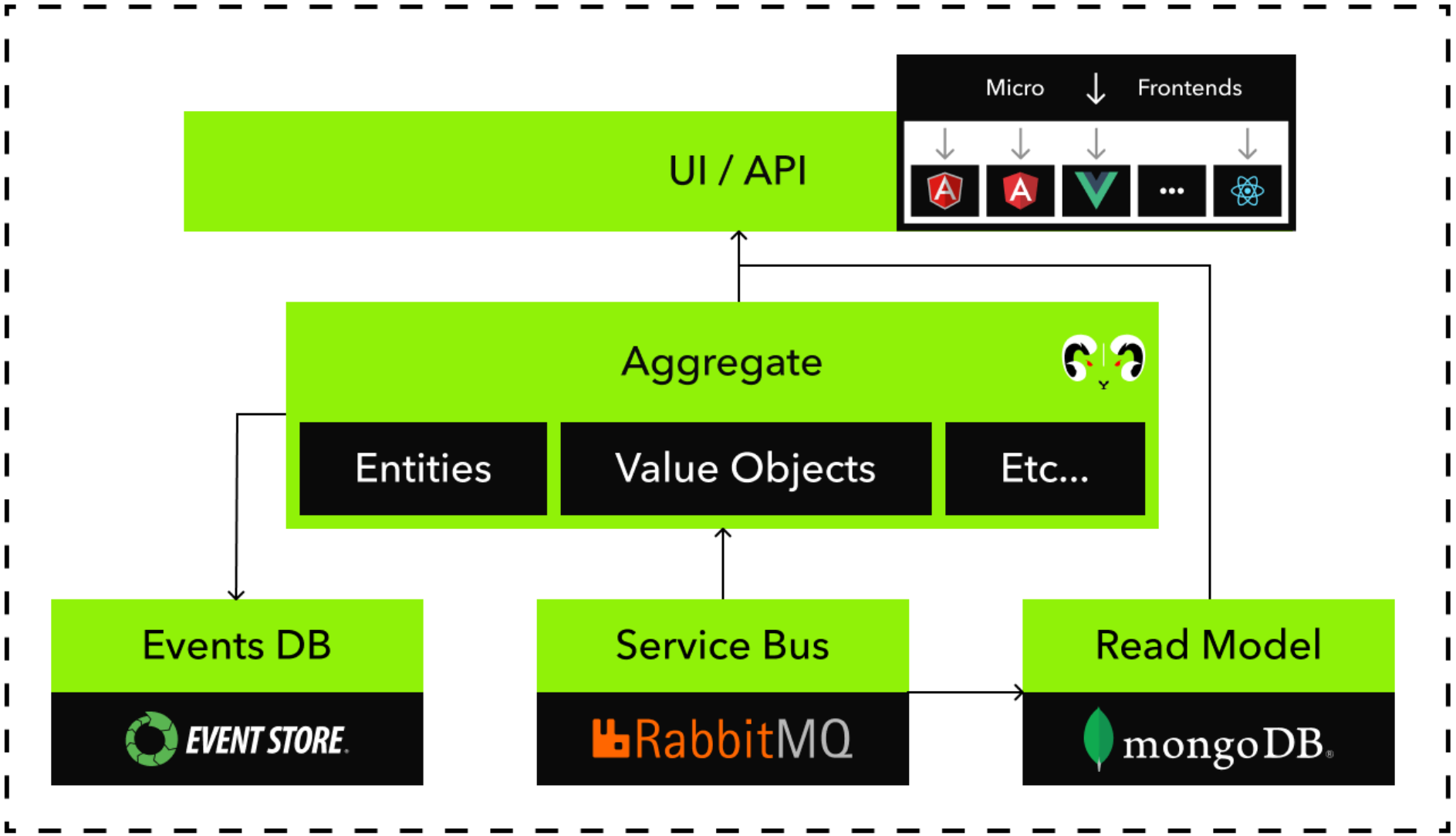
An architecture quantum measures several aspects of both topology and behavior in software architecture related to how parts connect and communicate with one another.

It is an independently deployable artifact with high functional cohesion, high static coupling, and synchronous dynamic coupling.

Bounded context is not enough

- Transactional boundary is not enough as a concept.
- We have to keep in consideration static and dynamic coupling of our components (e.g. DB, service bus, event store, etc.).
- We like to see it as «components cohesion» like the concept of functional cohesion

Components Cohesion / Quantum Arch.



Last responsibility moment

«Delay decisions as long as you can, but not longer. Maximize the information you have. Minimize technical debt from complexity.»

Postel's law or Robustness principle

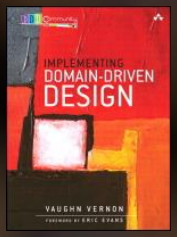
"Be conservative in what you do, be liberal in what you accept from others"
Only validate what you need

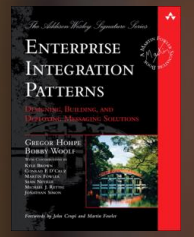
Trade-off everywhere

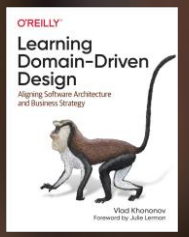
«Don't try to find the best design in software architecture.
Instead, strive for the least worst combination of trade-offs»

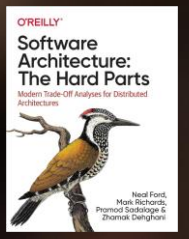
Bibliography

- Implementing Domain-Driven Design (V. Vernon)
  https://www.amazon.it/Implementing-Domain-Driven-Design-Vaughn-Vernon-ebook/dp/B00BCLEBN8/
- REST in Practice: Hypermedia and Systems Architecture (J.Webber)
  https://www.amazon.it/REST-Practice-Hypermedia-Systems-Architecture/dp/0596805829
- Enterprise Integration Patterns (G. Hohpe)
  https://www.amazon.it/Enterprise-Integration-Patterns-Designing-Deploying/dp/0321200683/
- Learning Domain-Driven Design: Aligning Software Architecture and Business Strategy (Vlad Khononov)
  https://www.amazon.com/Learning-Domain-Driven-Design-Aligning-Architecture/dp/1098100131
- Hands-On Domain-Driven Design with .NET Core (A. Zimarev)
  https://www.amazon.it/Hands-Domain-Driven-Design-NET/dp/1788834097
- Software Architecture: The Hard Parts (N. Ford, M. Richards, P. Sadalage, Z. Dehghani)
  https://www.amazon.com/Software-Architecture-Trade-Off-Distributed-Architectures/dp/1492086894










Thank
You

You can find us at

𝕏 @aacerbis
in https://www.linkedin.com/in/aacerbis/
✉ alberto.acerbis@intre.it

𝕏 @collaalessandro
in alessandrocolla
✉ alessandro.colla@evoluzione.agency