Armen Melkumyan • 1st
Technical / Solutions Architect
1mo •

🚀 Debunking Misconceptions in Microservices Architecture

I keep seeing this image circulating on LinkedIn, attempting to illustrate Monolithic vs. Microservices Architecture. While the intention is good, the microservices side of this diagram is fundamentally incorrect.

🔴 What's wrong with this illustration?

1) Redis and MySQL connected to RabbitMQ? ❌

Redis and MySQL don't directly interact with RabbitMQ. They are databases, not message producers/consumers. RabbitMQ is a message broker, typically handling communication between services, not acting as a direct link to databases.

2) Database confusion 🤔

Microservices should own their data, but this illustration makes it unclear which service owns which database. In real-world systems, each microservice has its own dedicated data storage, with APIs handling interactions, not direct database-to-database links.

🔷 What Should a Correct Microservices Architecture Look Like?

API Gateway orchestrates requests but doesn't directly handle databases.

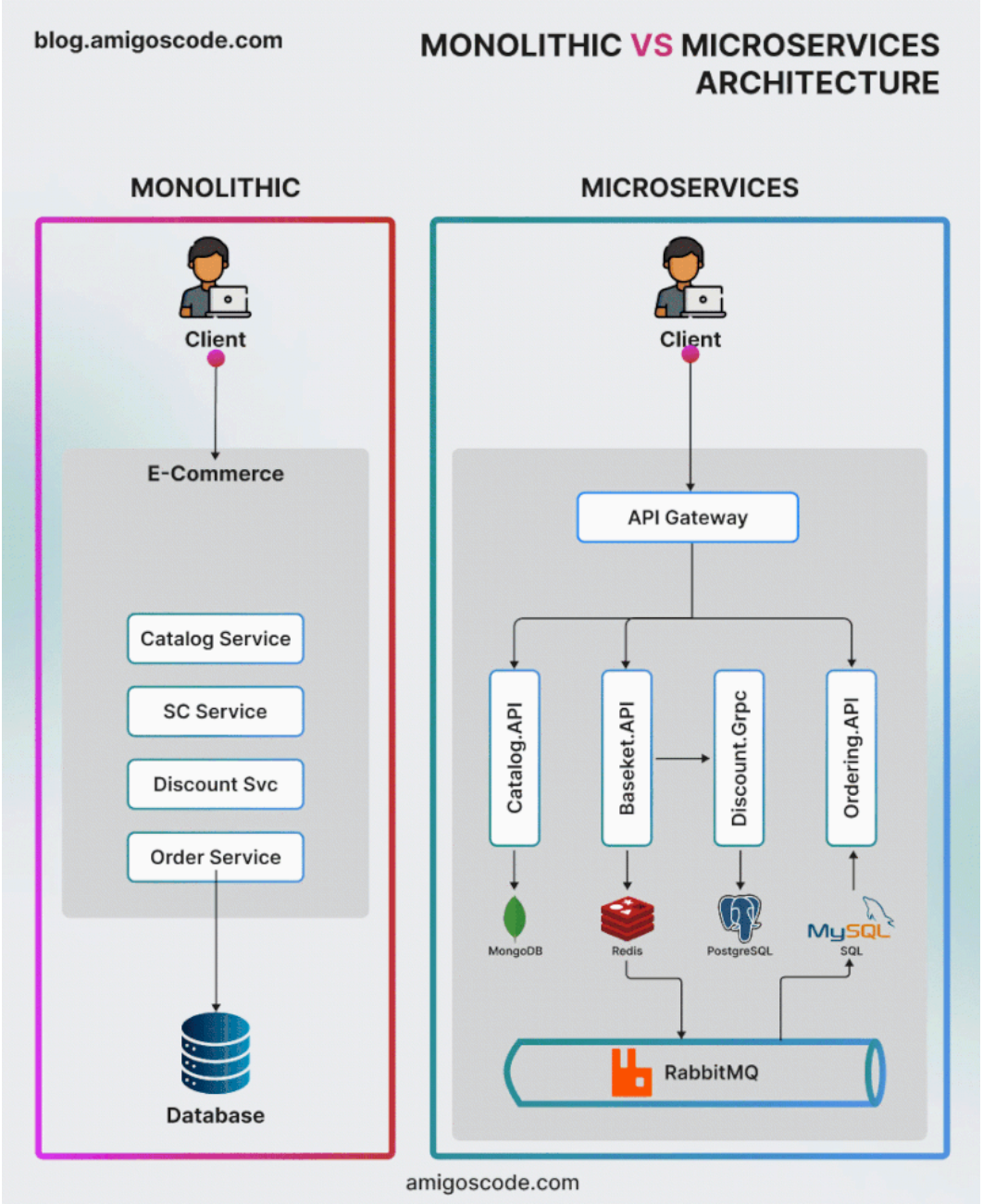
Each microservice owns its database (e.g., Catalog.API → MongoDB, Ordering.API → MySQL).

RabbitMQ should only facilitate event-driven communication (e.g., processing orders asynchronously, not serving as a direct DB connector).

Proper separation of concerns, ensuring scalability, resilience, and maintainability.

Why I wrote about it?

Many developers new to microservices see these types of illustrations and assume this is the correct approach. But poor architecture leads to scalability issues, unnecessary complexity, and brittle systems.



189 13 comments 11 reposts