Please stay away from Microservices!

Why should we be more cautious on adopting them, and when is the right time to use them?

What do we do?

- Collaboration platform for teams to collect, organize and understand their data.

Where are we?

- Thousands of businesses around the world.
- Millions of users.
- Dozens of services.
- All on a hybrid architecture.

Monolith vs. SOA vs. Microservice

- How big are they?
- How many services do we have? (1, 10, 100?)
- How automated are they? (DevOps)
- How are they connected?
- How many teams work on them?

Let’s start it up!

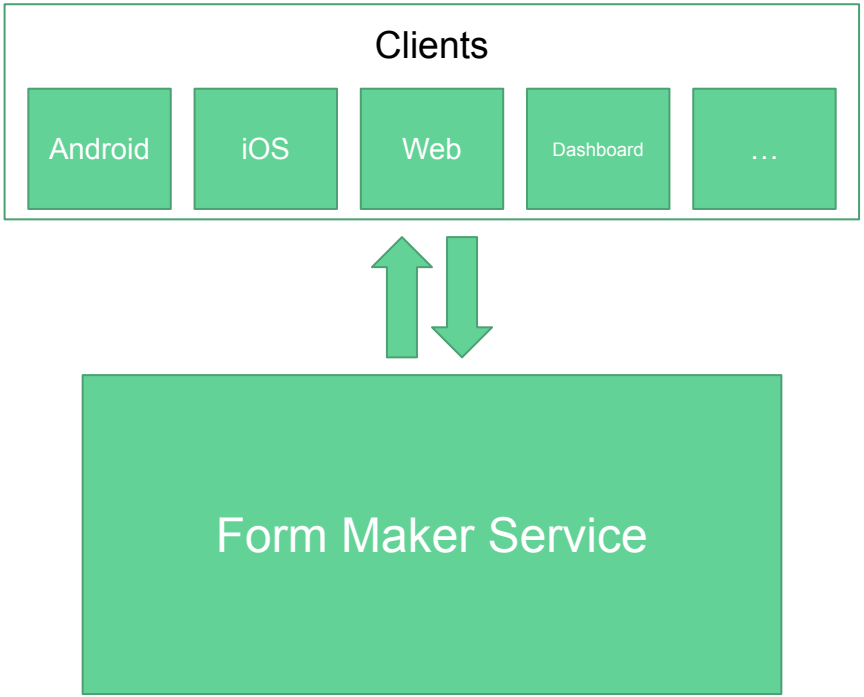
- Small domain
- Limited audience
- Small team
- Limited resources

= You may just need a simple, humble monolith!

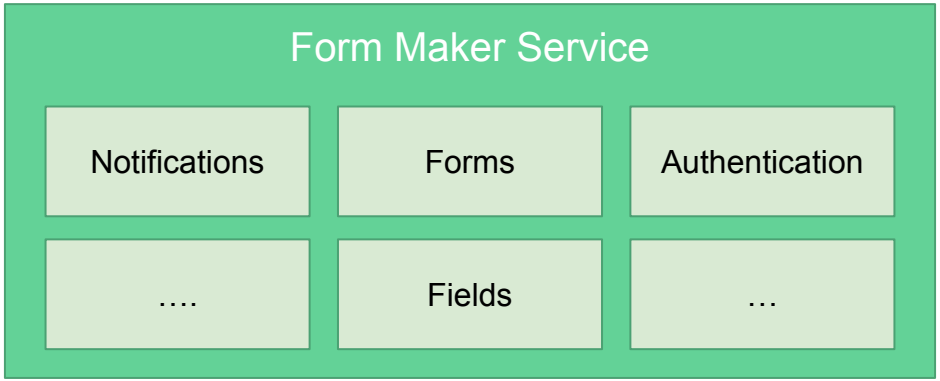
💡 Why not Microservices?

- It’s all about the trade offs.
- You can’t afford to solve the problems you don’t have.
- Small domain + Small audience + Small teams = You don’t **need** microservice.
- Limited resources = You can’t **afford** it!

This is how it began



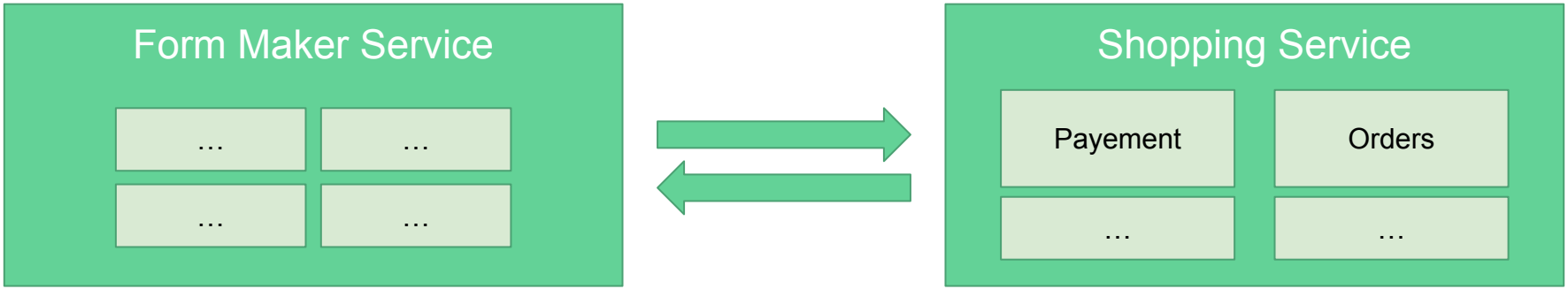
This is how it **really** began



- See the future, but don’t live in it!
- Modularity for the win!
- Writing component based code is an important investment.

Let’s get distributed!

- Need for payments and order management.
- Re-use what we have.



Here come the trade offs

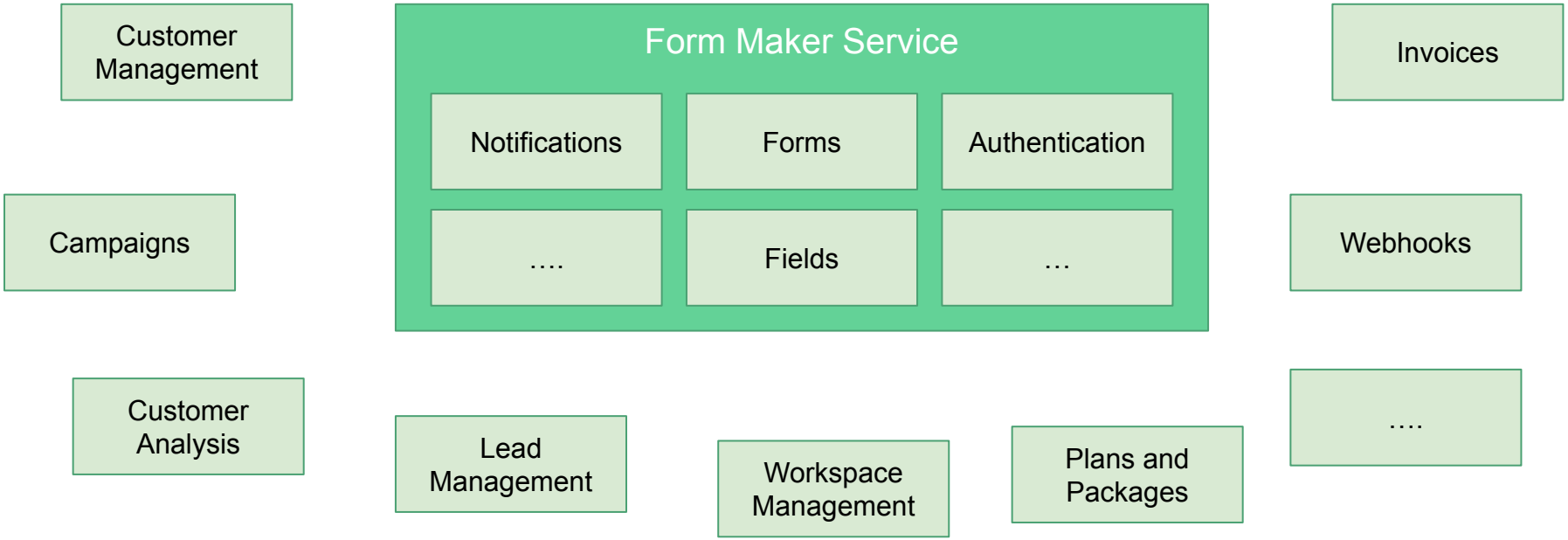
We gained

- Didn’t redo ourselves
- Didn’t overdo the job
- Saved time and effort
- Didn’t make our main service bigger

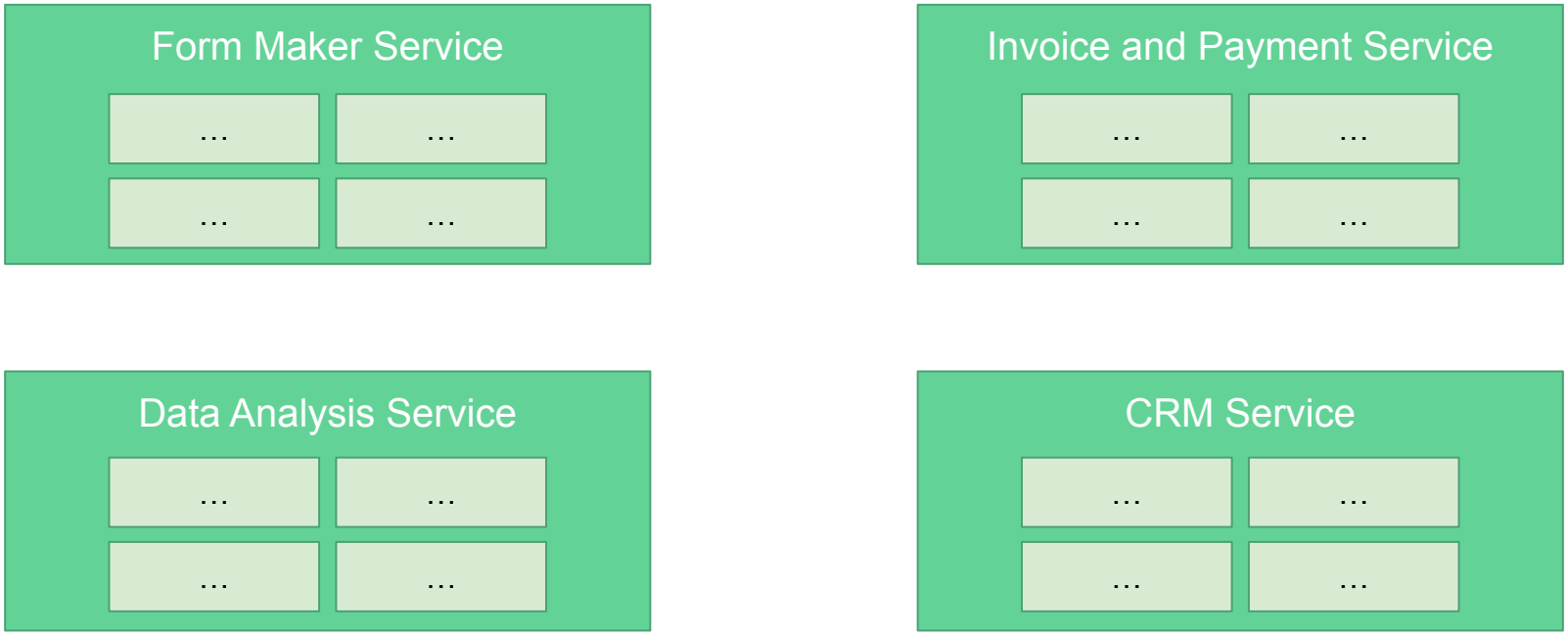
But it cost

- Performance reduction
- Less flexibility
- Less reliability

Here comes the many new features!



And let’s create the new services!



💡 Why not microservices?

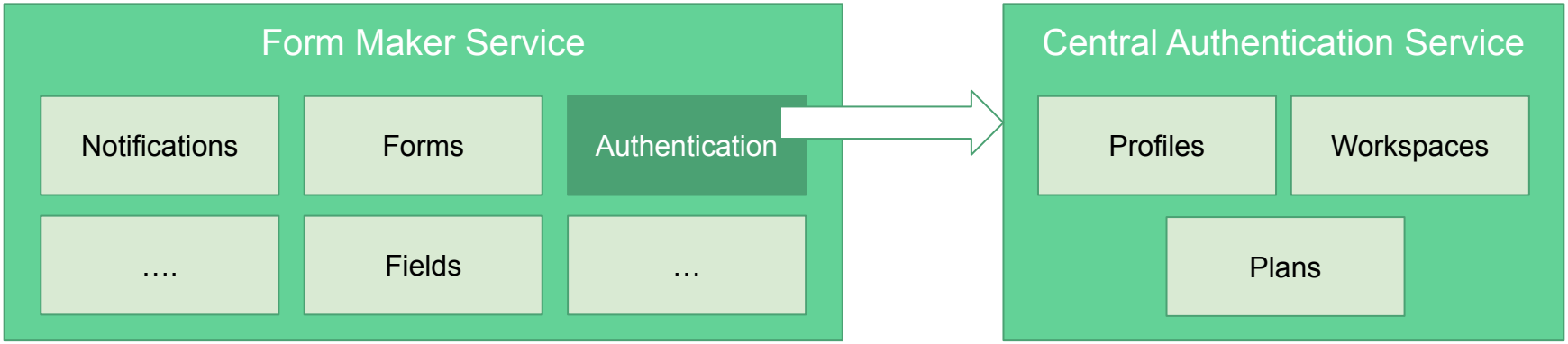
- Still didn’t really need it.
- It’s about **Automation** + **Size.**
- We needed the **Automation** part.
- Smaller sizes come with trade-offs that we neither needed nor could afford.

💡 What are the downside with smaller sizes?

- Communications
  - Latency
  - Performance Issues
  - Reliability Issues
  - Validity Issues
  - Security Issues
- More services to manage
  - One team, one service vs. One team, multiple services

The moment of truth: Breaking down a service!

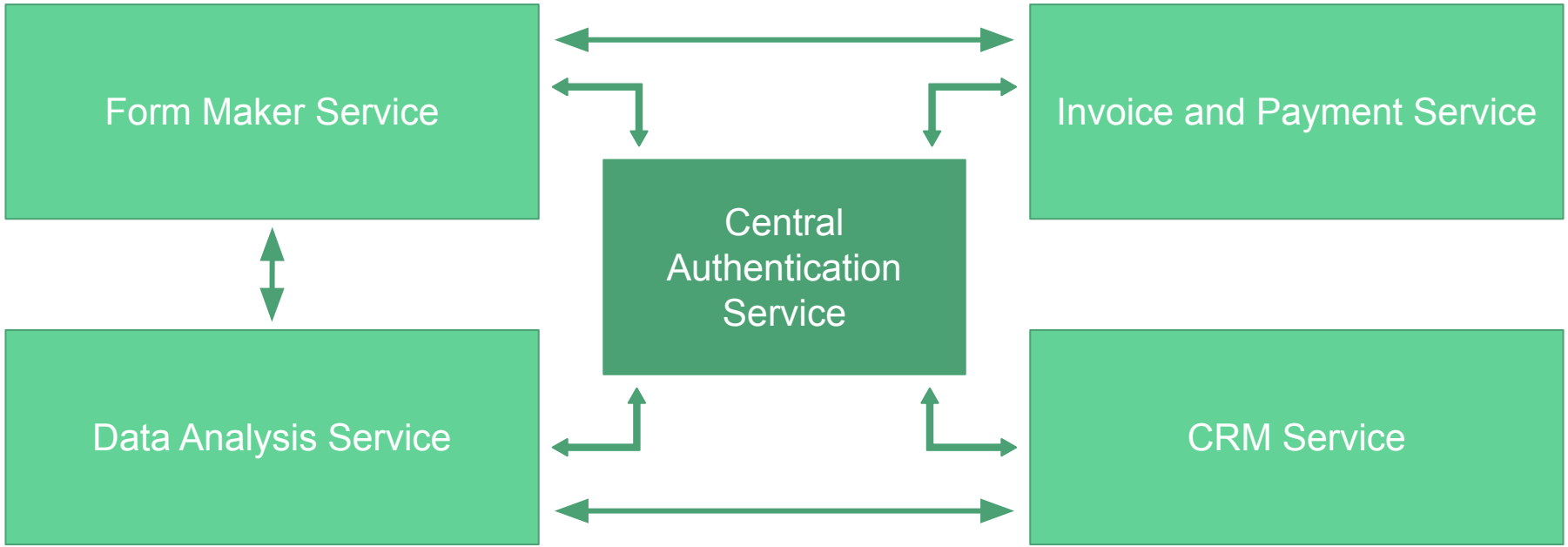
- We needed a central authentication for all services.
- We had to extract the **Authentication** module from the form maker service.



💡 Why didn’t we go “Micro”

- *“The term **Microservice** is a label, not a description!” ~Martin Fowler*
- Too small = Too much communication = Many new problems
- Microservice vs. Macroservice (though it’s not a real thing!)

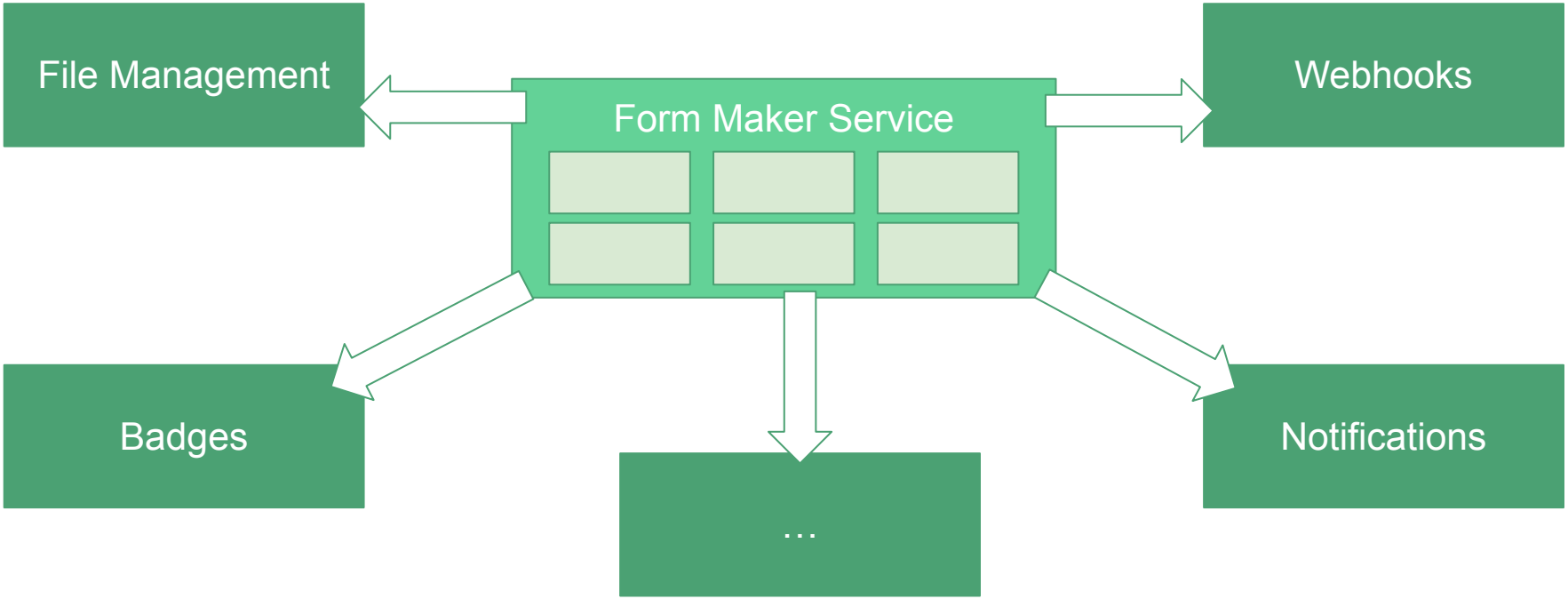
And here comes the first microservice!



And now we need to scale!

- As our audience grew, we could see how each module works
- We could predict which modules will become a bottleneck in the future
- We identified the potential microservices of the future

New and potential Microservices



💡 Somethings can’t be broken down

- Not all services can be separated to smaller parts.
- Some domains are large and highly coupled by nature.
- Breaking down the highly coupled services will cause many problems:
  - The communication between microservices is costly.
  - If two microservices need too much communications, they shouldn’t have been separated.
  - The updates between the services will need too much coordinations.
  - ...
- We need to do something else.

💡 What to do with “unbreakable” services?

- We can use hybrid architectures.
- Some architectures styles (like space-based) may be a better solution for your bigger services.
- It always depends!

How do we add a new service?

- If it’s fairly independent from current services: New microservice.
- If it’s highly dependent to current services: It depends!
- We always look out for the current services to break them to new microservices down if needed.

💡 When to avoid Microservices?

- Your service size is not that large.
- You don’t (and won’t) have a huge amount of users.
- It doesn’t solve an imminent problem.
- You don’t know how to do it (just yet)!

💡 How to be ready for microservices?

- Always create modular services.
- Lookout for trends in your services to identify the potential bottlenecks.
- Have a plan for the near and far future.
- Keep your knowledge updated.
- Make it safer and easier to test new stuff.

💡 So, should I use microservices?

- It depends!
- Software architecture is always about trade-offs.
- It’s always about choosing the least worst solution!
- They’re just a tool, it can all come down to how well can you use them.
- Not everyone needs them.

Learn more

- Fundamentals of Software Architecture An Engineering Approach by Mark Richards, Neal Ford
- https://microservices.io/
- https://martinfowler.com/
- Read article about those who have done it.
- Test what you learning action!

- Hasan Noori
- Co-founder and CTO
- hasan@formaloo.com
- https://xishma.medium.com/