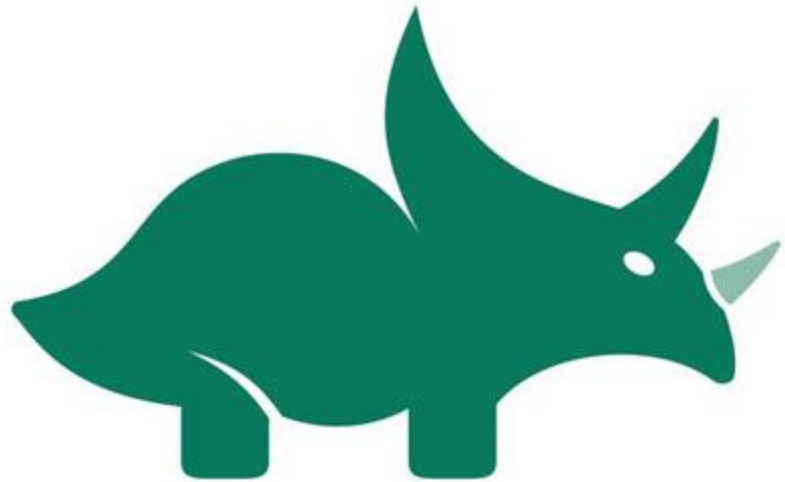


ITSAUR

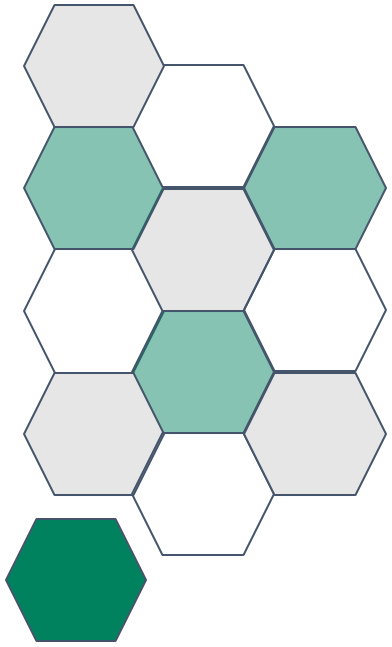
SOFTWARE SOLUTIONS



Grigoriadis Grigoris

Co-Founder & Software Architect @ ITSAUR
SOFTWARE SOLUTIONS

Microservices enthusiast !



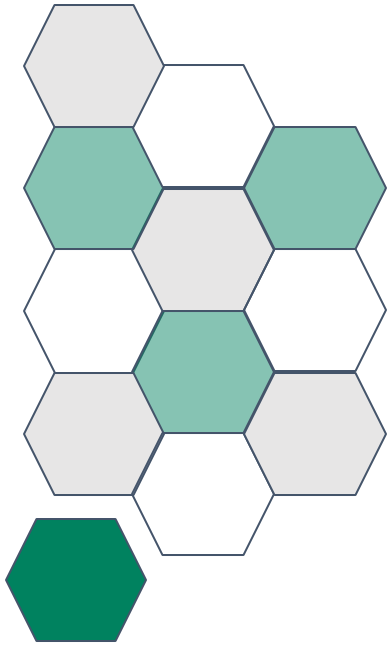
Introduction to Microservices

What we will cover

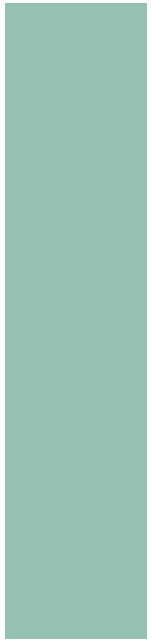


Monolithic vs Microservice Architecture

- Part 1
  - Introduction to Software Architecture
  - How to choose
  - Definition of Monolithic Architecture
  - Pros & Cons of Monolithic Architecture
- Part 2
  - Definition of Microservices Architecture
  - e-commerce Example
  - Pros & Cons of Microservices Architecture



Microservice
VS
Monolithic



Software Architecture



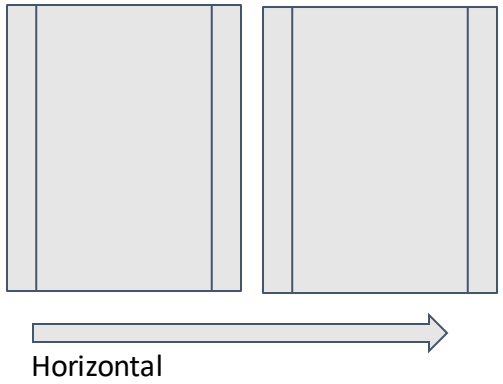
High level structure of a software system:

- Definition of Components
- Relationships between the components
- Principles for design and evolution of the components
- Define the important “ilities” of the system

The important stuff—whatever that is

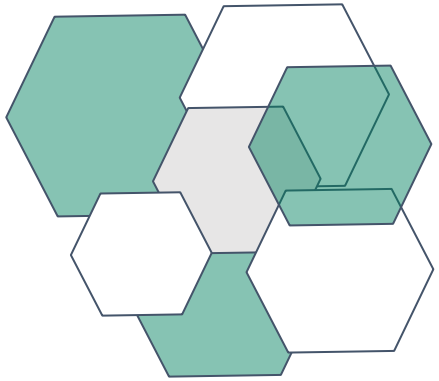
Architecture “illities”

Scalability



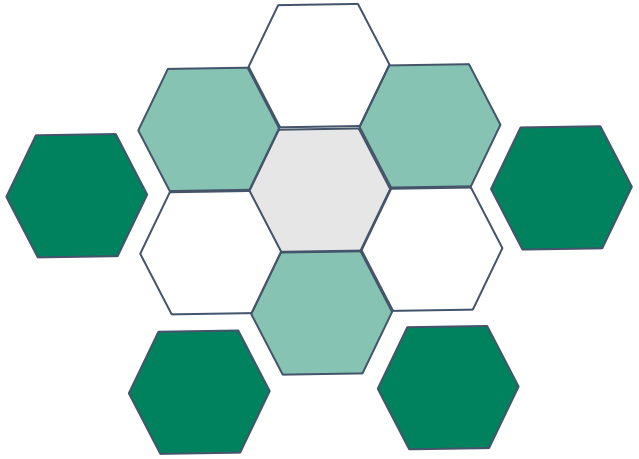
How well the system handles a growing amount of data/requests?

Maintainability



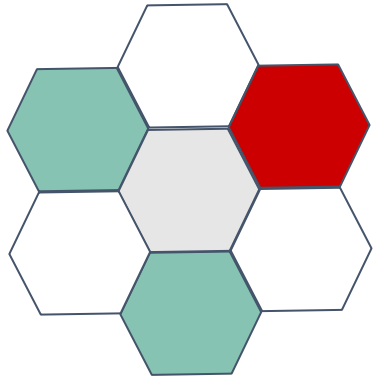
Is the system simple and easy to understand?

Evolvability



Is the system design in such a way that can evolve along with the product requirements and the changing ecosystem?

Fault Tolerance



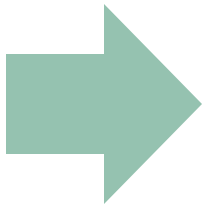
How well the system can handle failures

How to choose

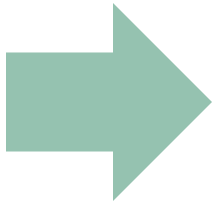


Take **Factors** Into account to decide the **Important Stuff** consider the trade-offs, make **Architectural Decisions**

- Project Requirements
- Budget
- Timeline
- Available Resources
- Company Organization
- Compliance
- Team Size & Skills
- Integration



- Maintainability
- Performance
- Fault tolerance
- Security

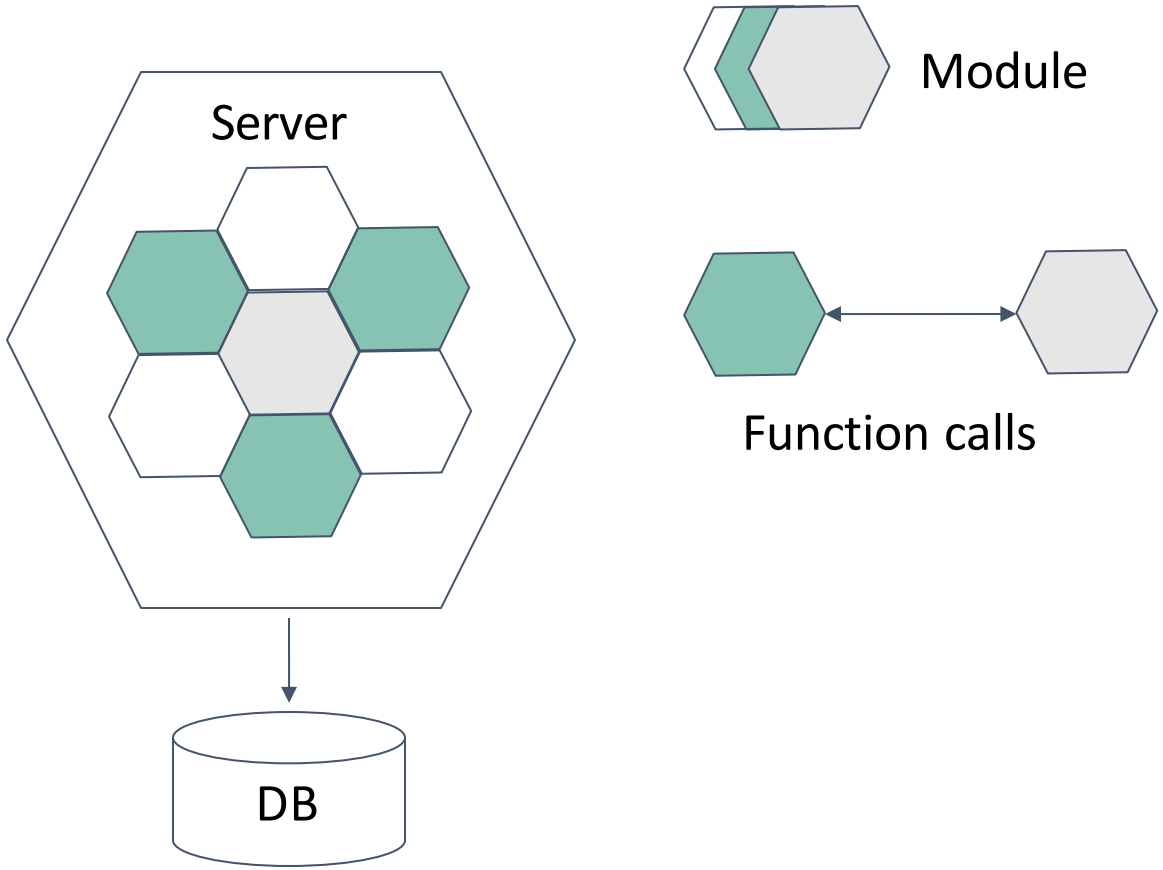


- Java
- Client Server
- Microservices
- Event Driven
- NoSql
- Containers
- REST
- RDBMS



Q&A

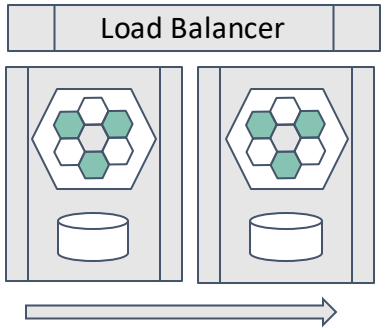
Monolithic Architecture



- Building block is the code modules
- Modules communicate with each other using function calls
- System is deployed and run as a single OS process
- A Database is used in order to store the system state, usually a relational database

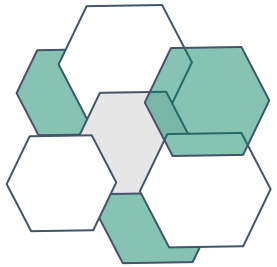
Monolithic “ilities”

Scalability



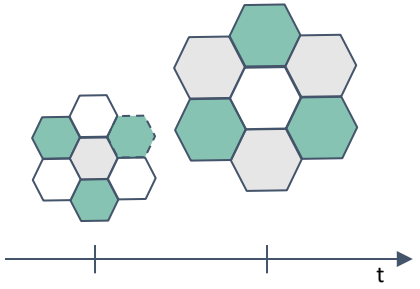
- Easier to scale
- The entire system must be scaled
- Each module has different resource requirements

Maintainability



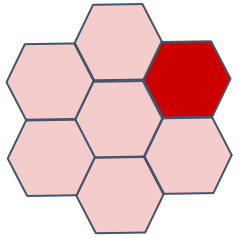
- Easier to debug
- Boundaries between modules tend to break
- Code size becomes intimidating
- A change can impact the whole system

Evolvability



- Any change (programming language, DB etc.) must be applied to the whole system
- Refactoring of database requires coordination between multiple teams

Fault Tolerance

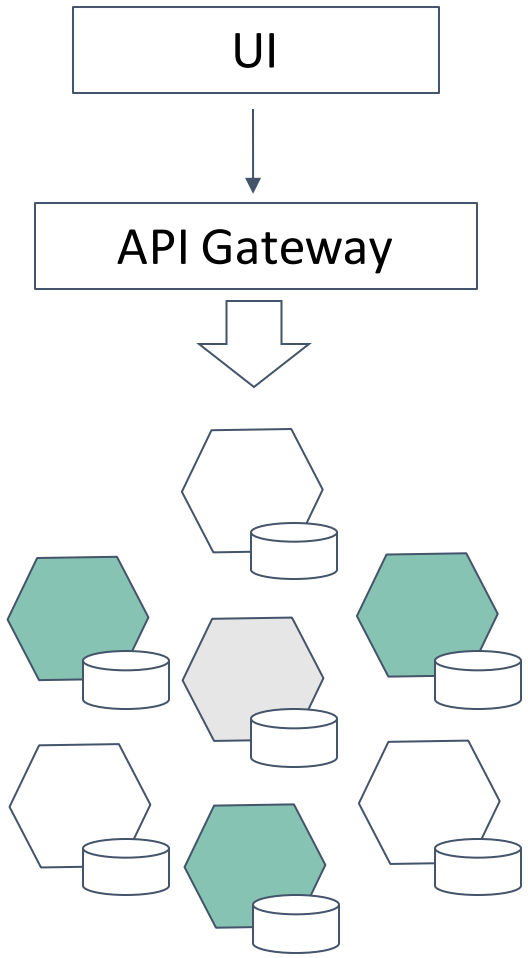


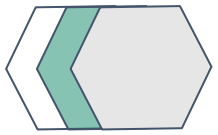
- If a module has a memory leak or malfunctions the whole system might stop working



Q&A

Microservice Architecture - Bare Minimum







HTTP calls

- Building block is the microservices
- Services communicate with each other via a lightweight interoperable communication protocol, usually HTTP
- Each service MUST have its own database
- An API Gateway provides a unified API for our system

REST API



GET /catalog/products/1

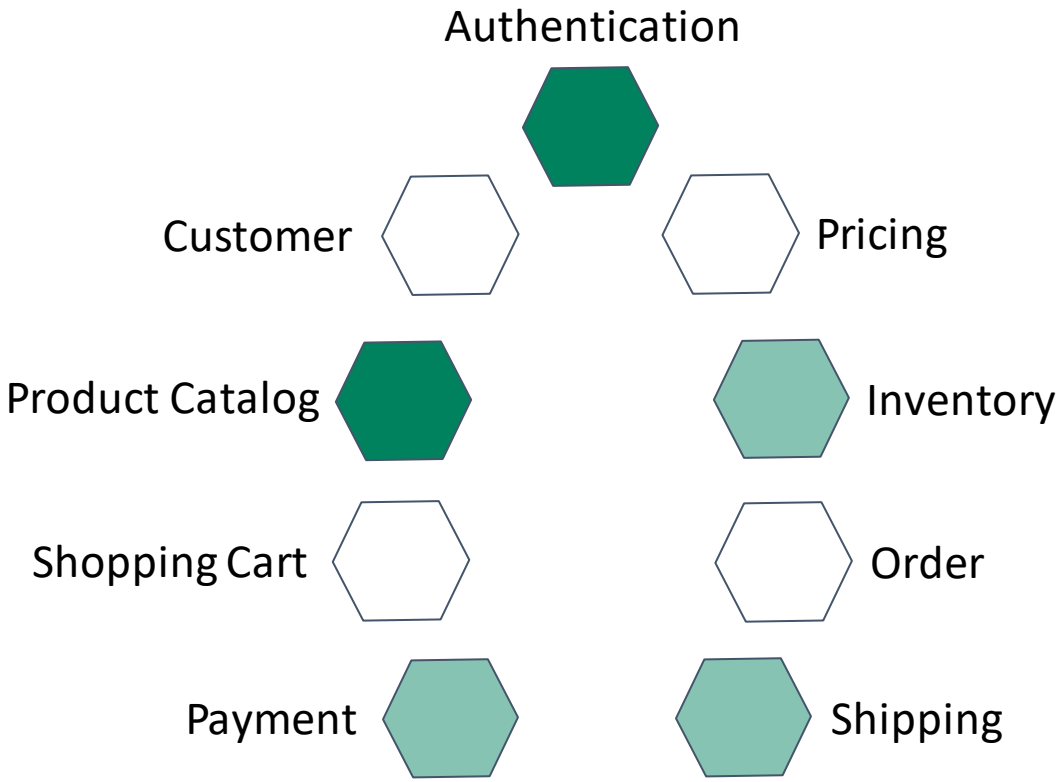
{method} /{service}/{resource}/{id}

Method	Action
POST	Create
GET	Read
PUT	Update
DELETE	Delete



Q&A

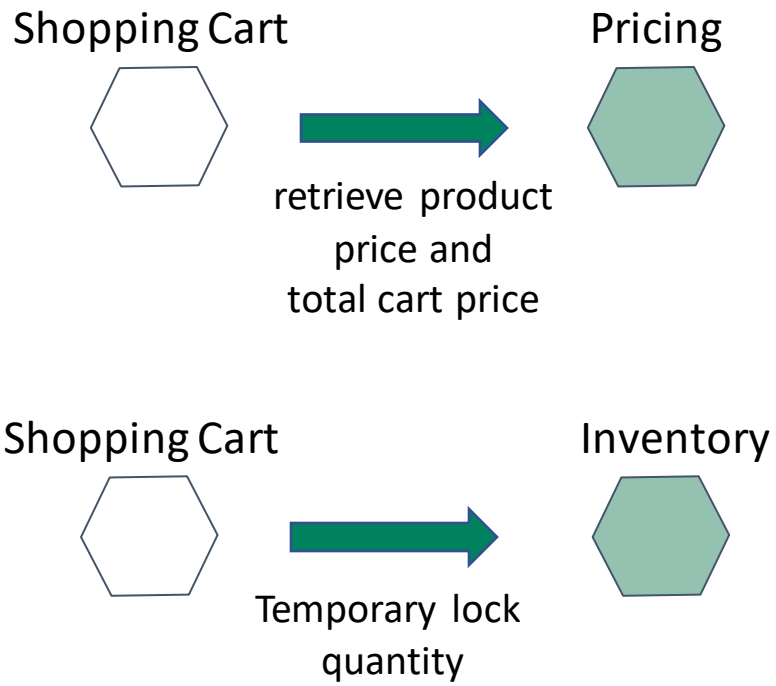
e-commerce Example



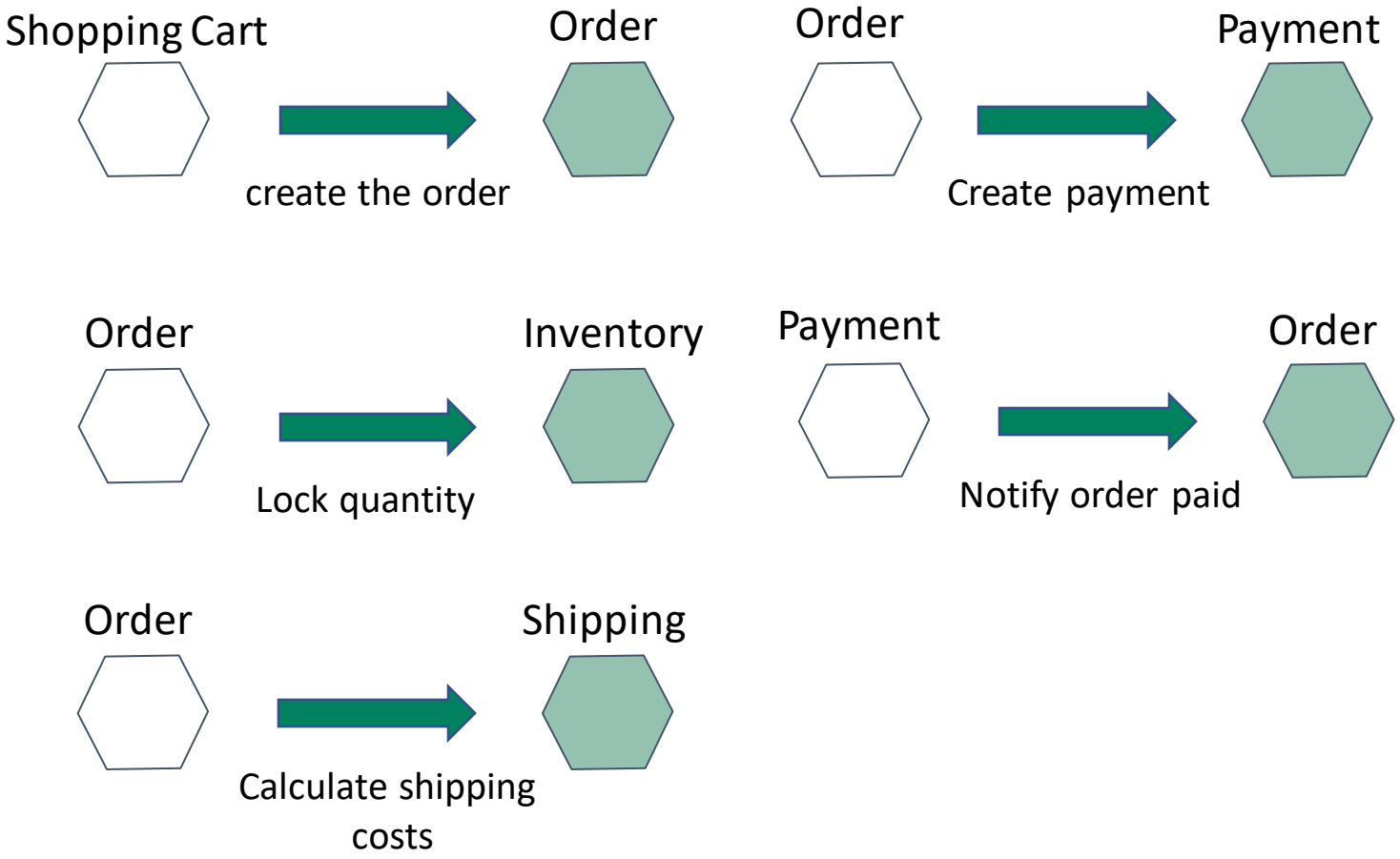
- Each microservice represents a specific functionality of the system
- Responsibilities of each microservice should be clear

e-commerce Example

Add product to shopping cart



Checkout shopping cart

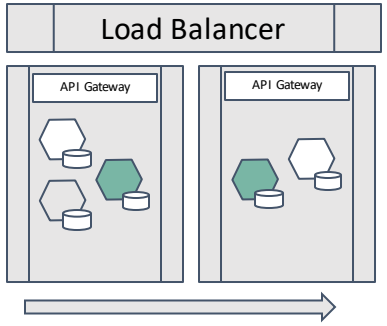




Q&A

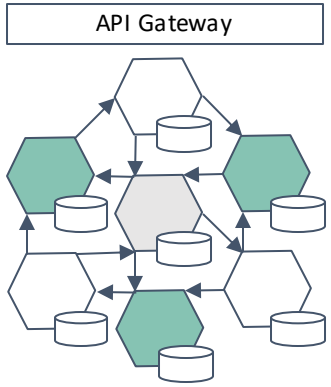
Microservice “ilities”

Scalability



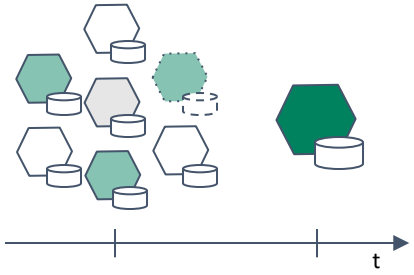
- Services can be deployed on different machines
- Services can be scaled independently
- Complicated to manage properly (requires orchestrators, service discovery, circuit breaker)

Maintainability



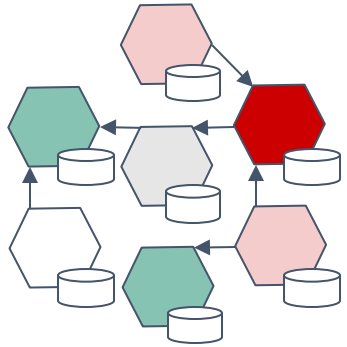
- Service code base is small
- Easier to work with multiple teams
- Communication between services might become difficult to follow
- Maintaining data consistency between microservices can be difficult

Evolvability



- Microservice can be rewritten individually using a different language or database
- Refactoring database is much easier
- New features can be added more easily as separate microservice without affecting the system

Fault Tolerance



- Part of the system will continue to work even if some services fail
- Faster recovery, individual services can be restarted without affecting the whole system



Q&A

Thank you!

References

Books

- Building Microservices: Designing Fine-Grained Systems
- Building Evolutionary Architectures: Support Constant Change
- Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems
- Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series)

Online sources

- https://martinfowler.com/architecture/
- https://microservices.io
- https://en.wikipedia.org/wiki/List_of_system_quality_attributes
- https://en.wikipedia.org/wiki/Software_architecture

Java Libraries for microservices

- Eclipse Vertx
- Micronaut
- Spring Boot
- Quarkus