NodeJS - 201

Book 1

B1. Introduction to Microservices

What are we covering in this session

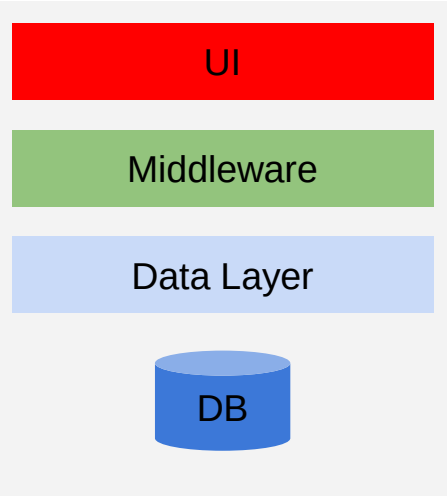
- Architectural problem background
  - Individual & Independent Evolution
- Identifying Microservices
  - Domain Model
  - Bounded Context
  - Coupling & Cohesion
- Testability

Case Study

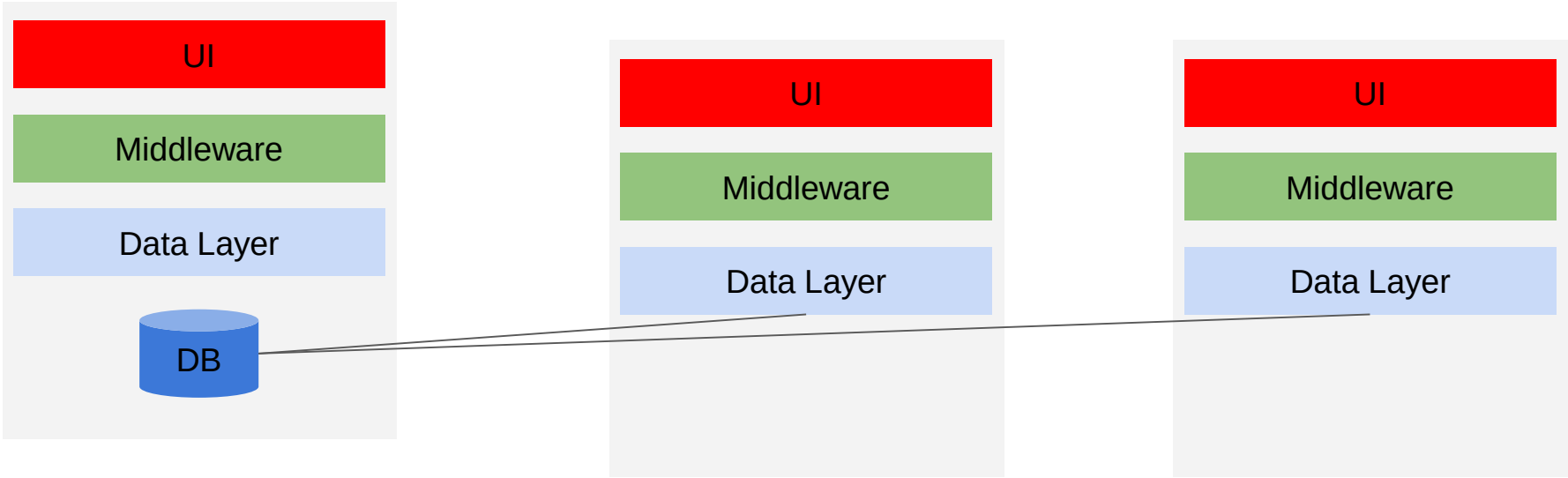
- Design Mobile payment wallet system, which supports cashback for certain eligibility criteria on payment made

- Design an Cab aggregating system, which can support digital payments, tracking and operates in multiple cities

Architectural Problem Background of µServices



- Monolithics reduce *Agility* of the product to meet its business or consumer demand
- Degree of Agility = Coupling & Cohesion of components
- Testability & Extensibility will be less
- Data model is highly denormalized
- Teams are big and chaotic



Solution

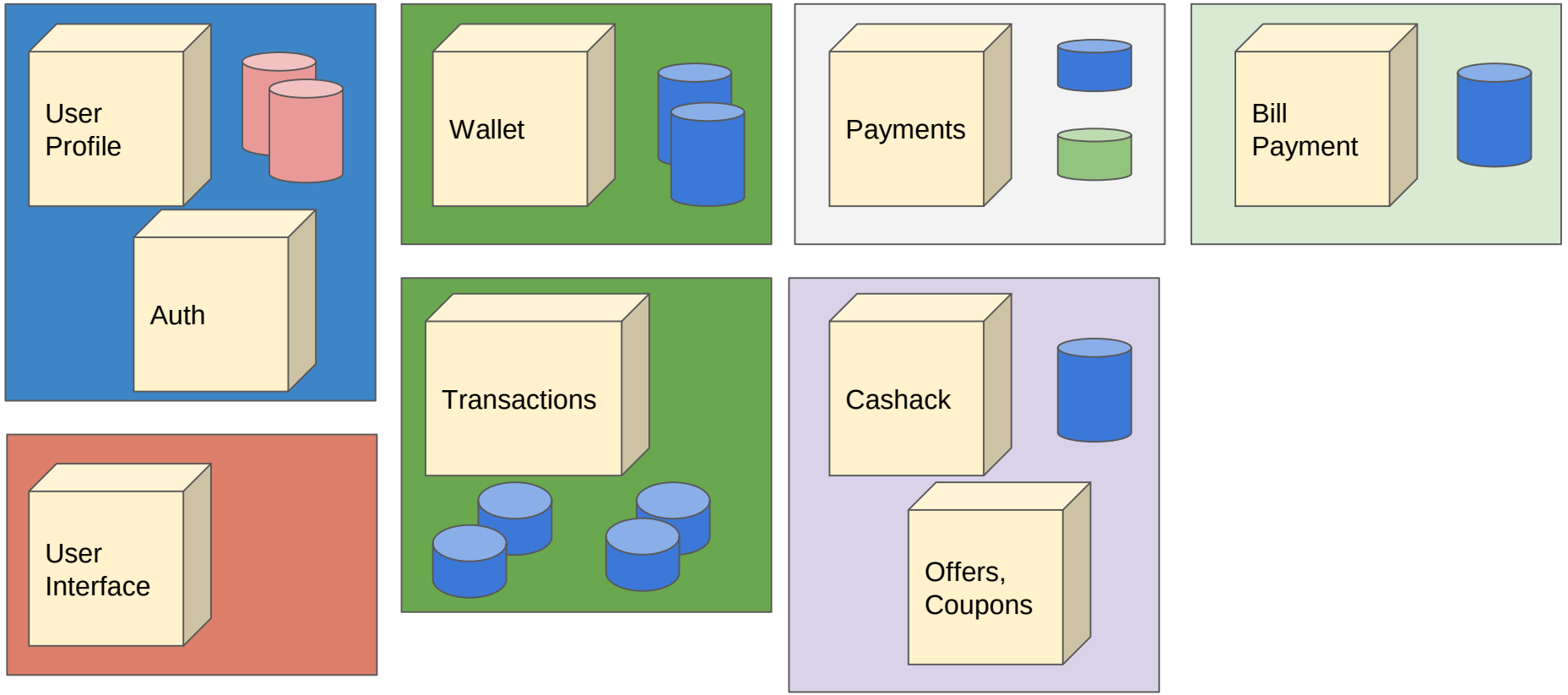
- Part & Whole (separation of concerns)
- Single Responsibility Principle
- Decoupled & Distributed systems
- Design by contract

µServices is one of the way to solve the problem

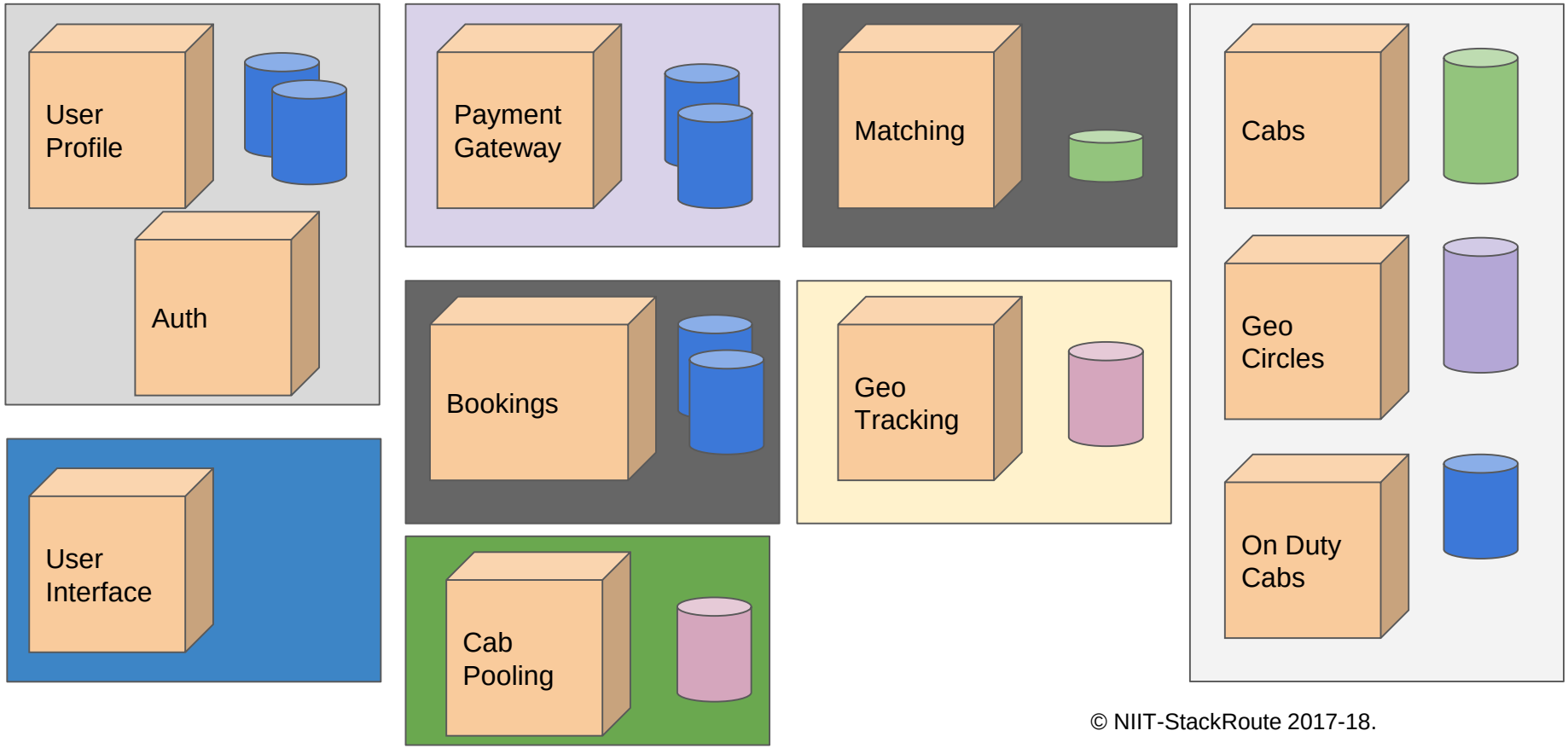
What is “micro” in µServices

- Independent system of smaller complexity
- Independently testable in isolation
- System has smaller data boundary
- Two american pizzas enough to feed the team
- Can release(deploy, distribute) Independently
- Another service cannot write or read data of another service

Mobile Wallet



Cab Aggregation system



Identifying Micro Services

- Single Responsibility (business capability)
- Gather together those things that change for the same reason.
- Separate those things that change for different reasons
- Can it be **independently evolved**?
- Identify Domain Entities, Process Entities
- Identify Bounded Contexts

Benefits of µServices

- Independently Evolve
- Completely testable
- Manageable by smaller team
- Can replace easily & completely
- Polyglot tech stack
- Extensible by adding new services
- Can scale only those which have higher load

How do system work as a whole then?

- Event Driven
- Message Driven
- API
- Orchestration
- Choreography

Overheads

- Multiple Services
- Automated Testing
- Automated Deployment
- Polyglot deployment environment
- Need to design for tolerating failure of service
- Testing at different levels (Testing Pyramid)

References

- [Details reading on microservices architecture](https://martinfowler.com/articles/microservices.html)
- [Video on how complex system can be designed as microservices](https://www.youtube.com/watch?v=CZ3wIuvmHeM)
- [What is bounded context](https://martinfowler.com/bliki/BoundedContext.html)
- [Reactive Systems](https://www.reactivemanifesto.org)
- [What is event driven](https://martinfowler.com/articles/201701-event-driven.html)
- [Testing Microservices](https://martinfowler.com/articles/microservice-testing/)

Reaching Out?

@SR-Dinesh
@SR-Basav
@SR-Neelanjan
@SR-Amisha