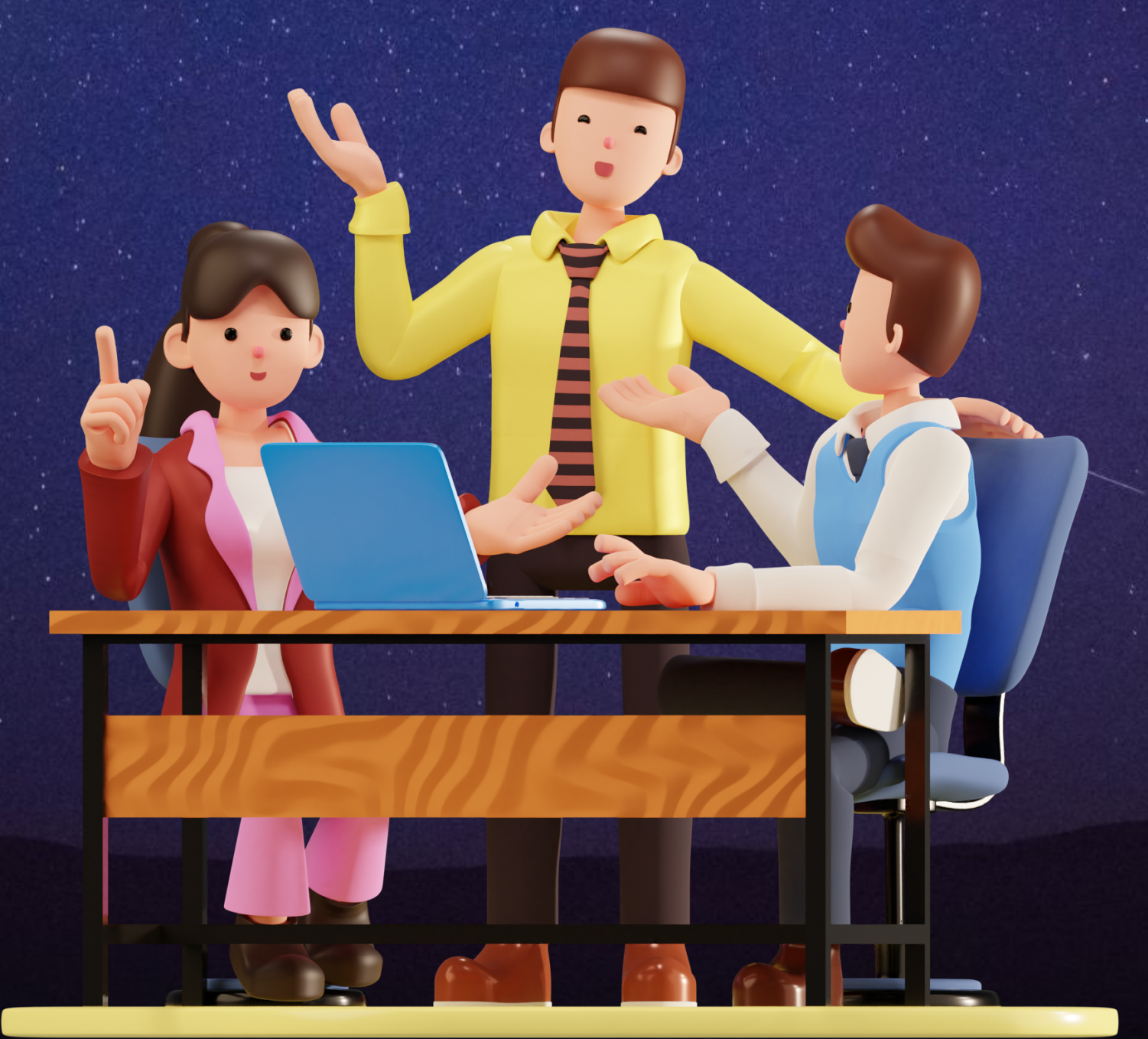
CODEWITH_RAJESH

RAJESH KUMAR

SWIPE >>>

MICROSERVICES DESIGN PATTERN

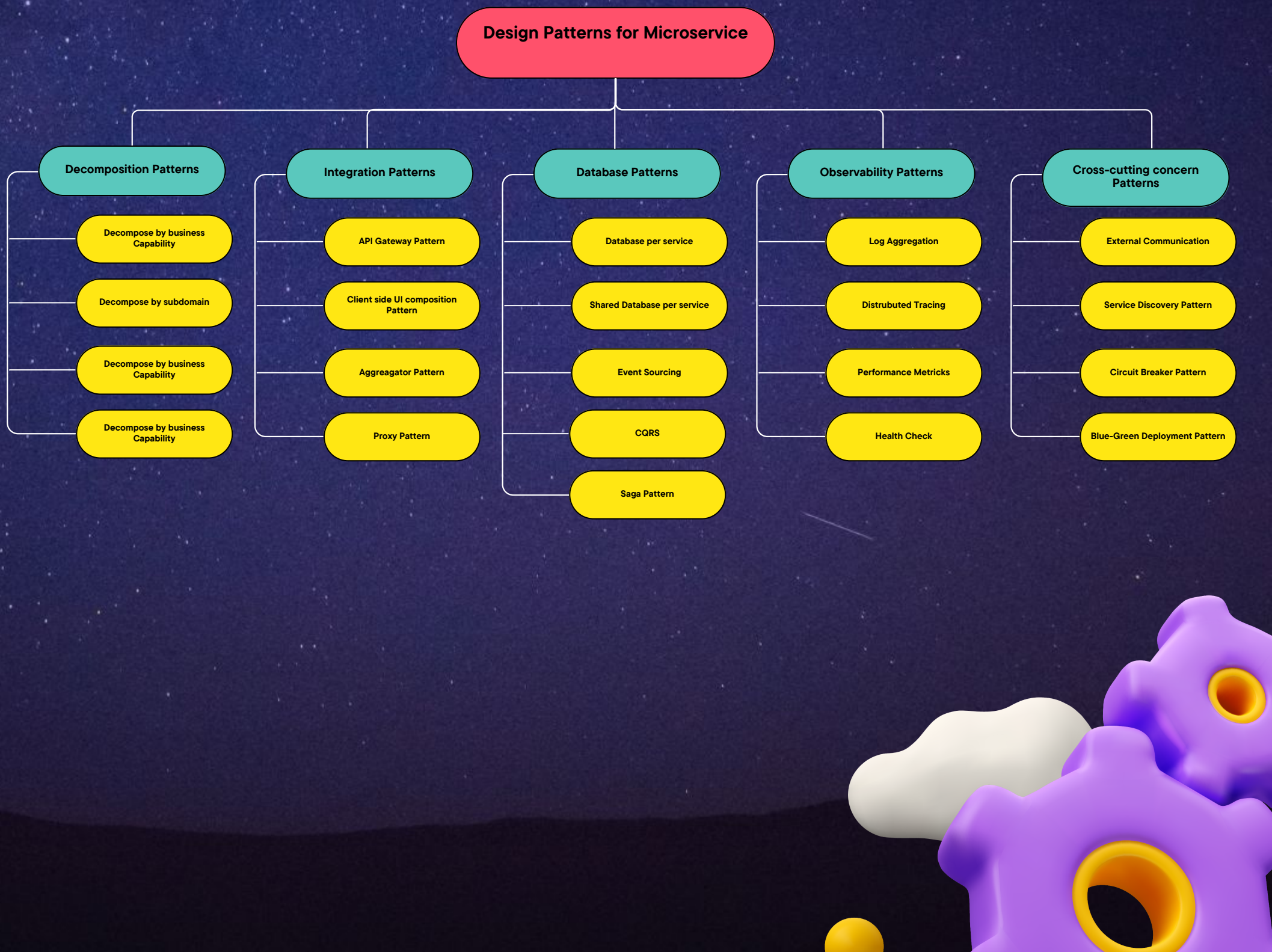


CODEWITH_RAJESH

RAJESH KUMAR

SWIPE >>>

Design patterns for Microservices



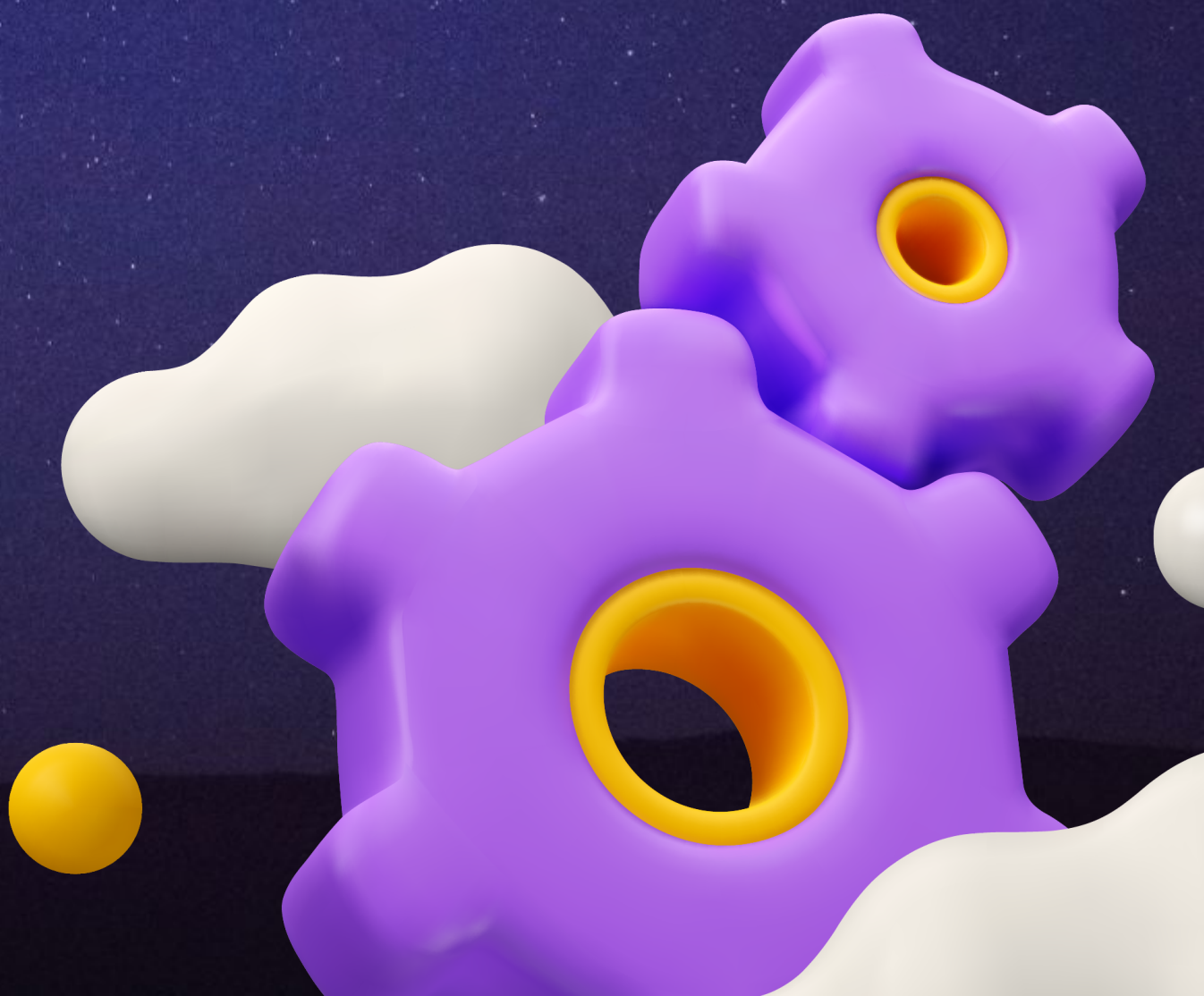
CODEWITH_RAJESH

RAJESH KUMAR

SWIPE >>>

1. Decomposition Pattern

- Divide the application into smaller, independent services.
- Each service focuses on a specific business capability or domain.
- Decoupling of services allows for scalability, independent deployment, and better fault isolation

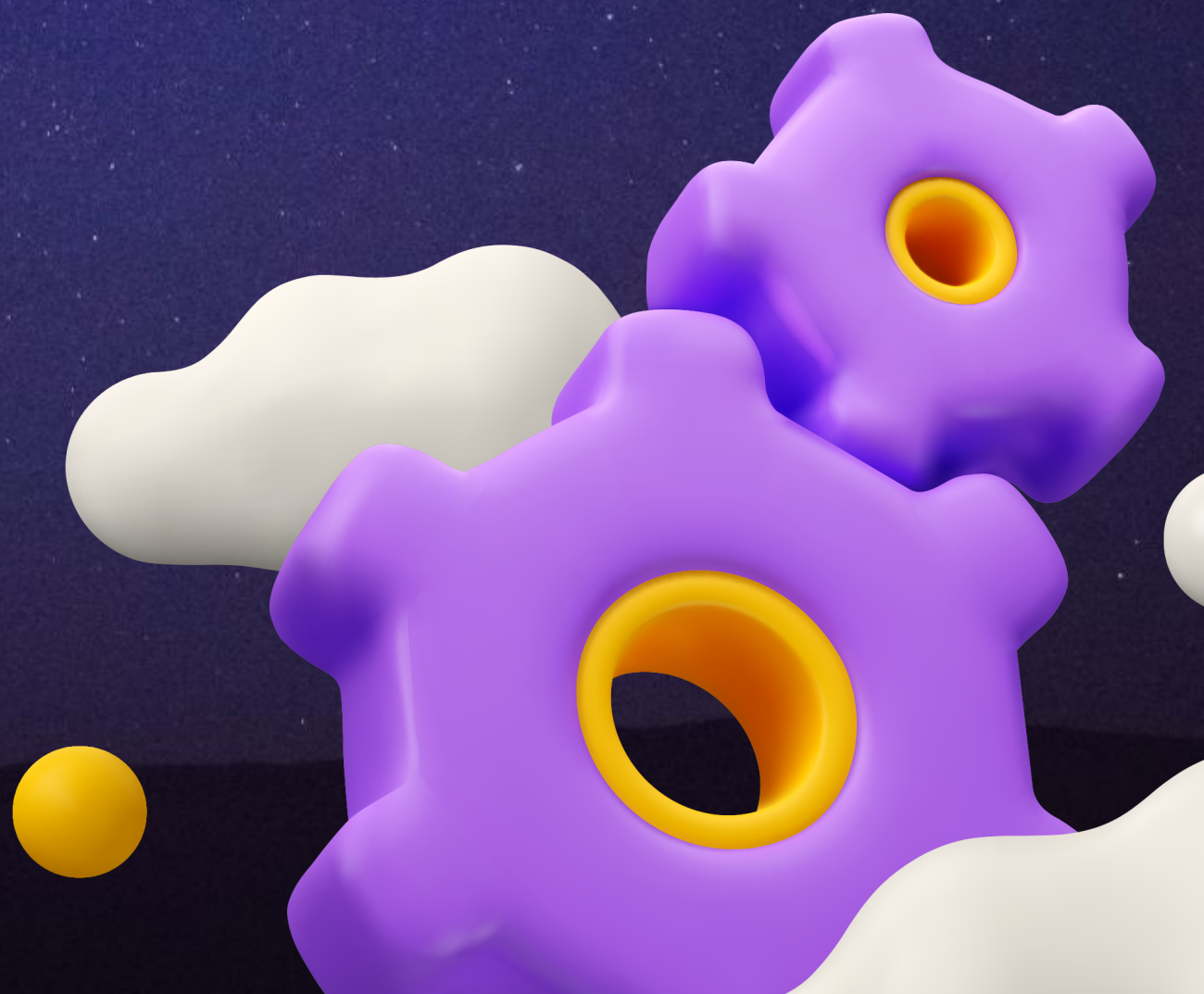


CODEWITH_RAJESH
RAJESH KUMAR

SWIPE >>>

2. Integration Pattern

- Use messaging systems or APIs for communication between services.
- Implement event-driven architectures or publish/subscribe patterns.
- Choose appropriate protocols and formats for data exchange, such as JSON or AMQP.

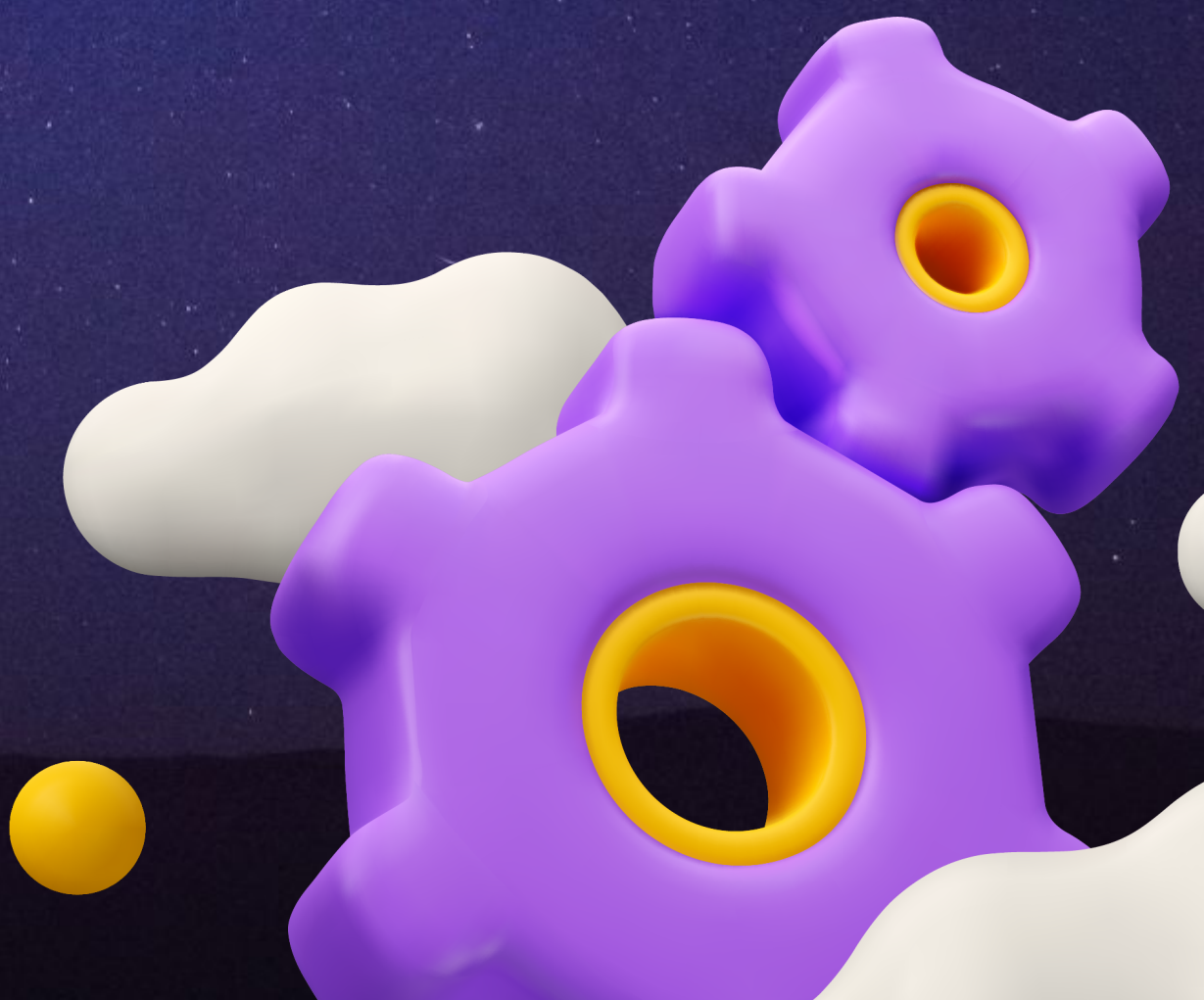


CODEWITH_RAJESH
RAJESH KUMAR

SWIPE >>>

3. Database Pattern

- Utilize a separate database per microservice to ensure loose coupling.
- Consider different types of databases for different data storage needs (e.g., relational, NoSQL).
- Implement data replication or synchronization mechanisms for maintaining consistency.

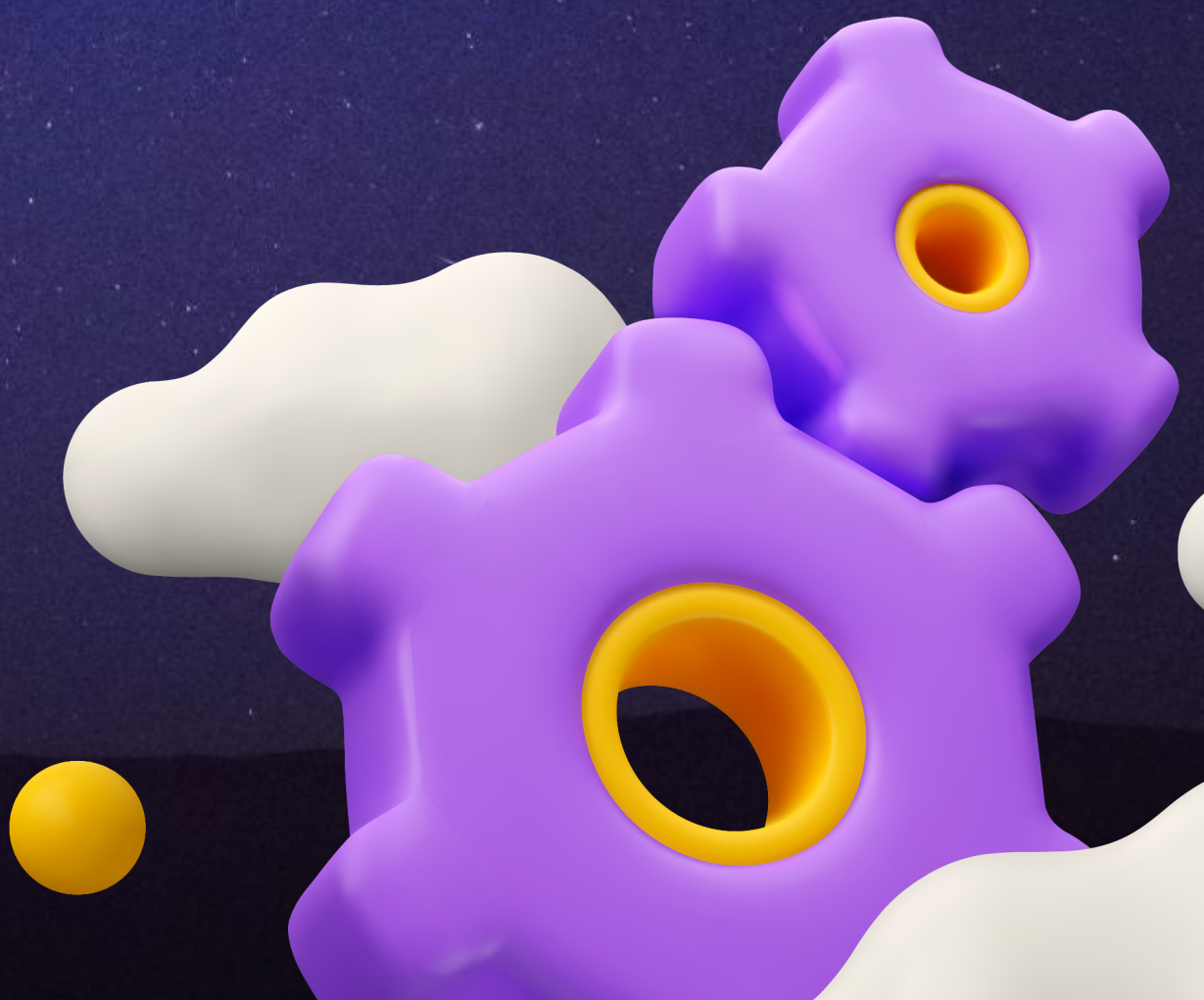


CODEWITH_RAJESH
RAJESH KUMAR

SWIPE >>>

4. Observability Pattern

- Incorporate logging, monitoring, and metrics to gain visibility into the system.
- Use centralized logging to collect and analyze logs from all microservices.
- Implement distributed tracing to track requests across multiple services for better troubleshooting.



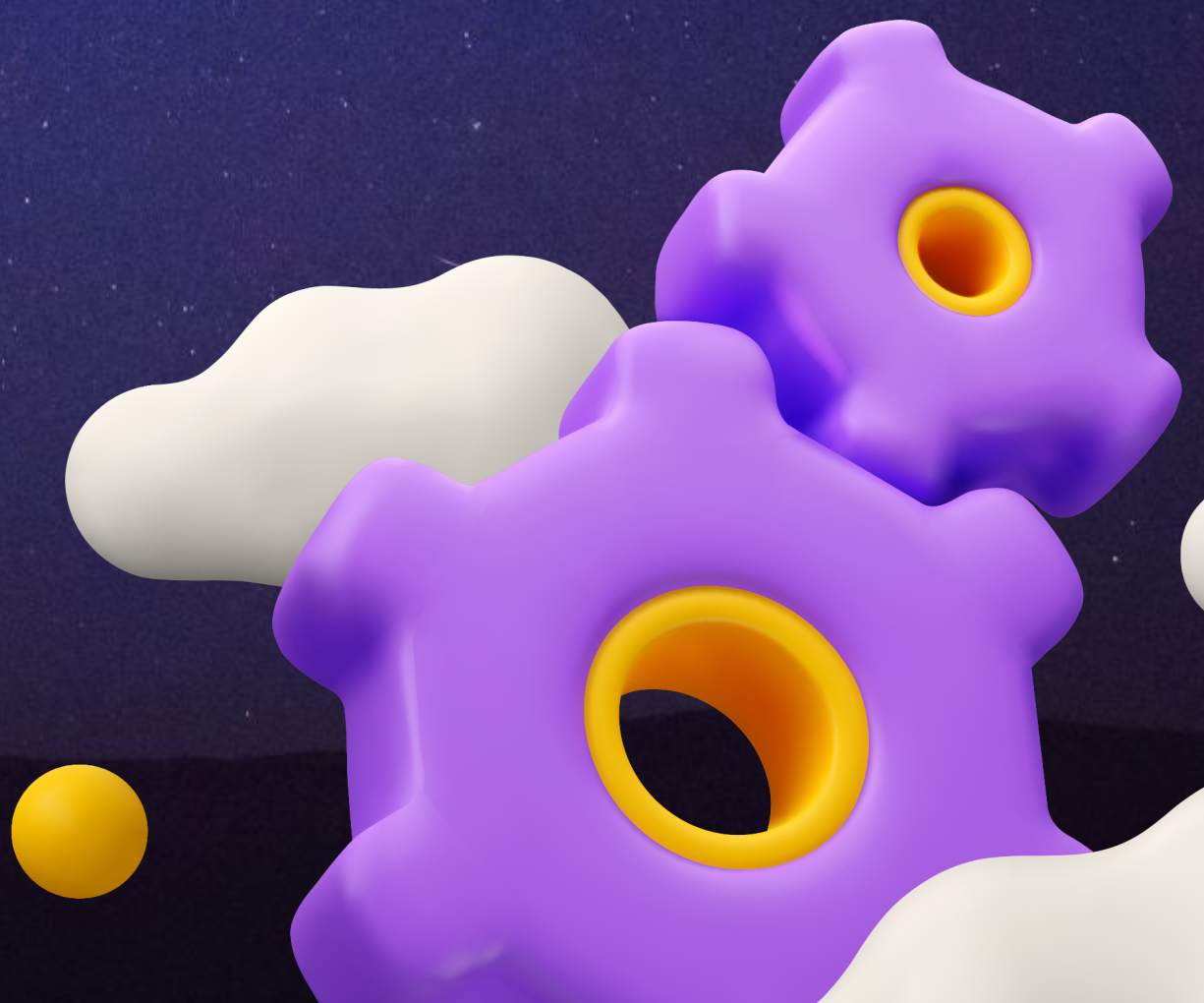
CODEWITH_RAJESH

RAJESH KUMAR

SWIPE >>>

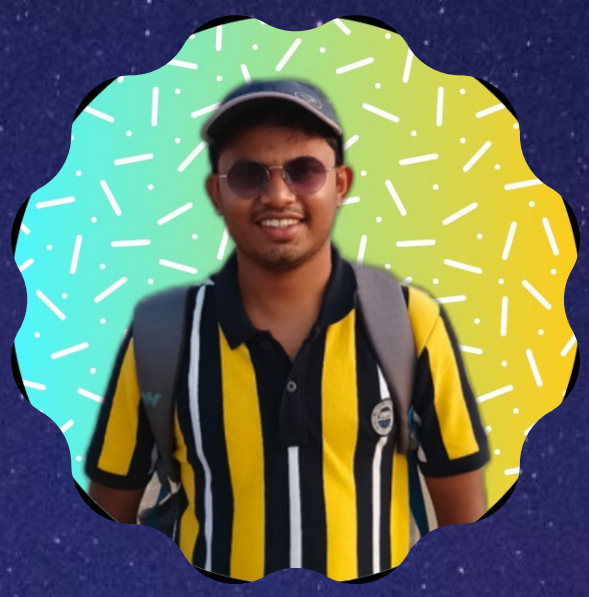
5. Cross-Cutting Concern Pattern

- Identify common functionality or concerns shared across multiple services.
- Extract these concerns into reusable components or libraries.
- Examples of cross-cutting concerns include authentication, caching, and logging.



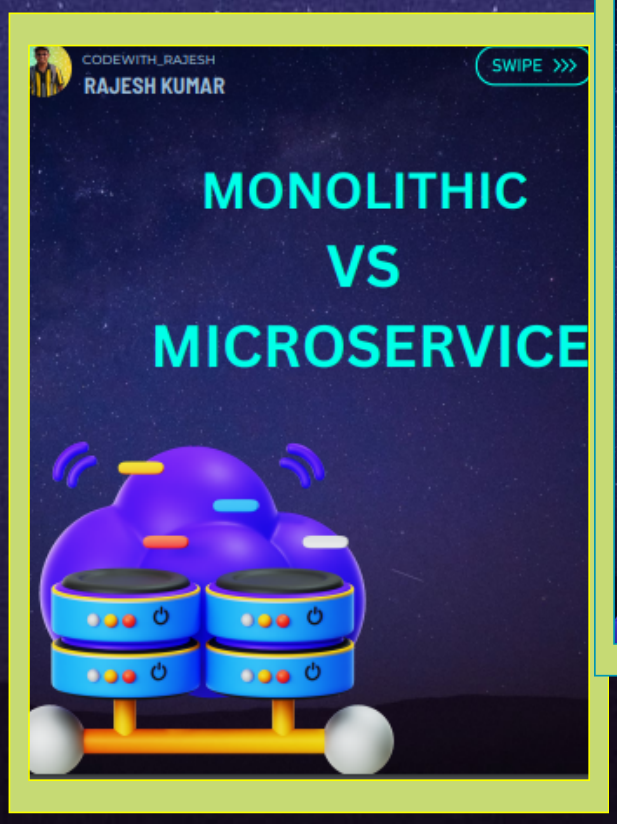
❤ Thanks for reading!

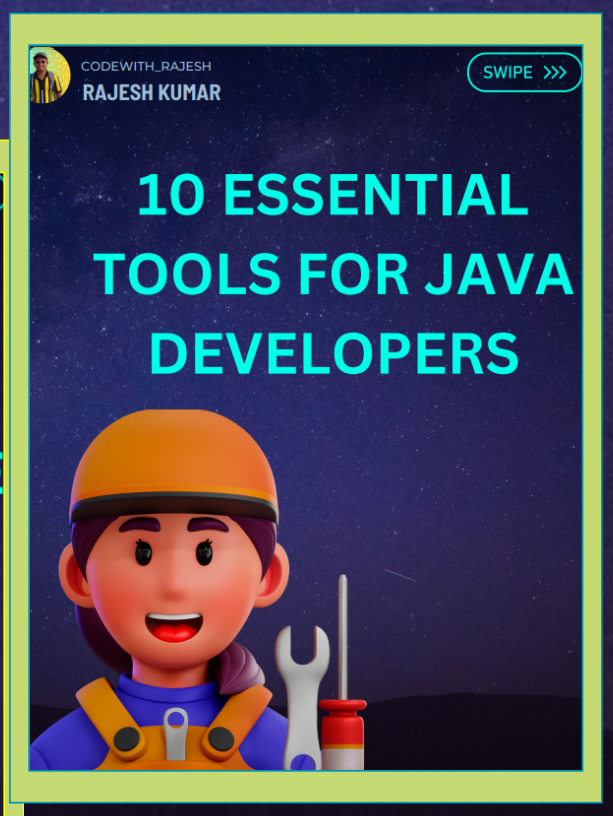
Stay up-to-date with the latest advancements in Java full stack development by following me on the handles below, where I'll be sharing my expertise and experience in the field.



codewith_rajesh

Rajesh Kumar





Like | Save | Share