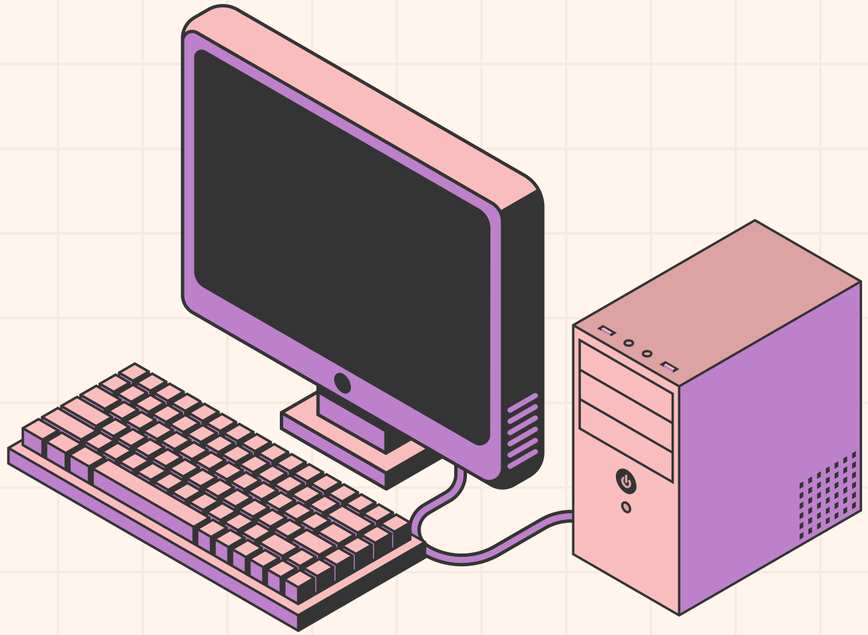
HOME SERVICE ABOUT US CONTACT US

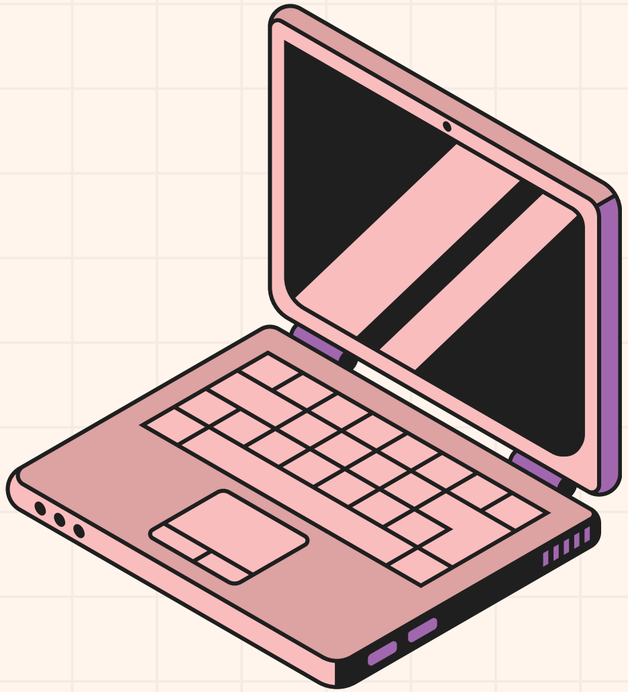
MODULAR MONOLITH

STRUCTURING CODE
BY DOMAIN WITHIN
A SINGLE DEPLOYABLE UNIT



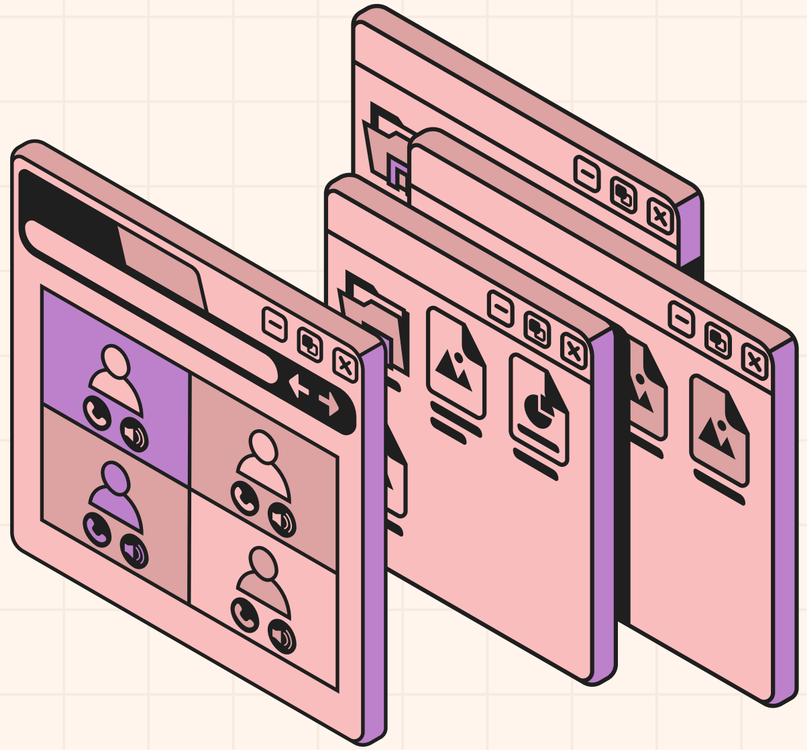
WHAT IS MODULAR MONOLITH?

Modular Monolith is an architecture where the system is deployed as a single application, but internally divided into clear business-domain modules. Each module has its own business logic and responsibilities. Modules communicate through interfaces or events.

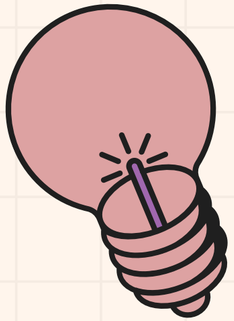


ARCHITECTURE

- Single deployable application
- Divided into independent, well-defined modules
- Clear module boundaries (DDD concept)
- In-process communication (no network overhead)
- Controlled data ownership per module
- High cohesion, loose coupling



STRUCTURING CODE BY DOMAIN

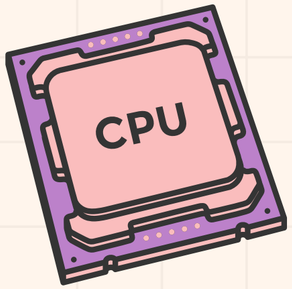


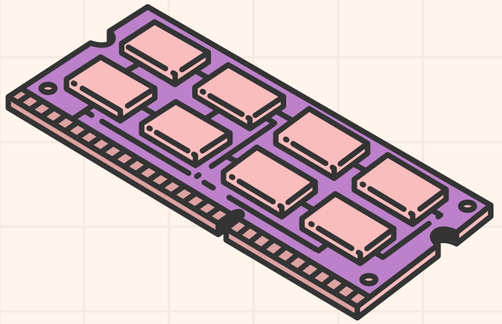
Example: E-commerce system

```
src/
 ├── user/
 │    ├── User.java
 │    └── UserService.java
 │
 ├── order/
 │    ├── Order.java
 │    └── OrderService.java
 │
 ├── payment/
 │    ├── PaymentService.java
 │
 └── MainApplication.java

 The application runs as a single deployable unit.
```

WHY AND WHY NOT MODULAR MONOLITH?





ADVANTAGES

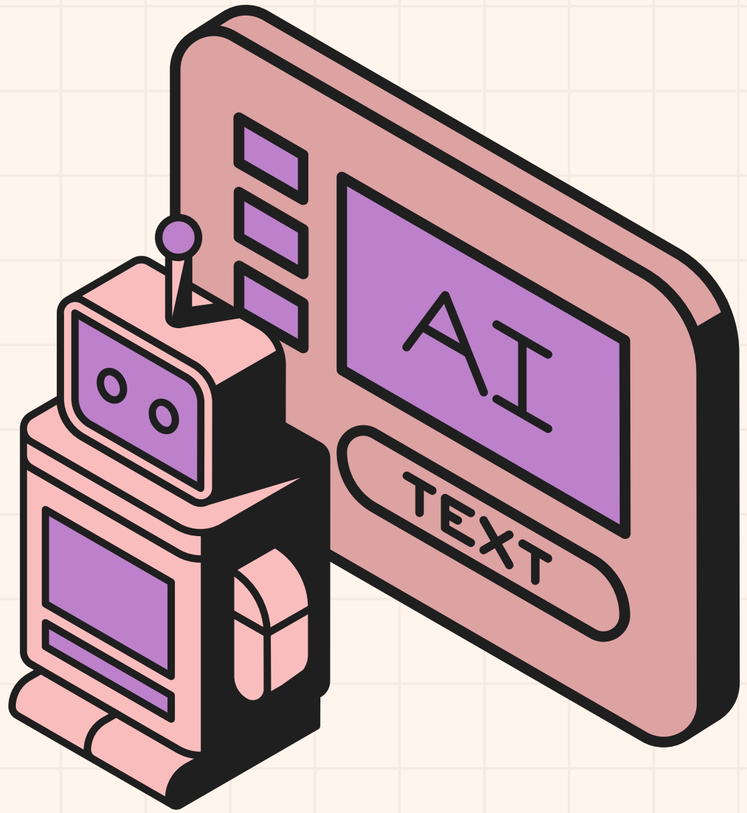
- Simple deployment (single application)
- Easier debugging and testing
- Clear domain separation
- Better maintainability than traditional monolith
- Easier transaction management
- Can evolve into microservices in the future

DISADVANTAGES

- Entire system must be deployed together
- Cannot scale modules independently
- Requires strong architectural discipline
- If poorly designed, it can become tightly coupled like a traditional monolith

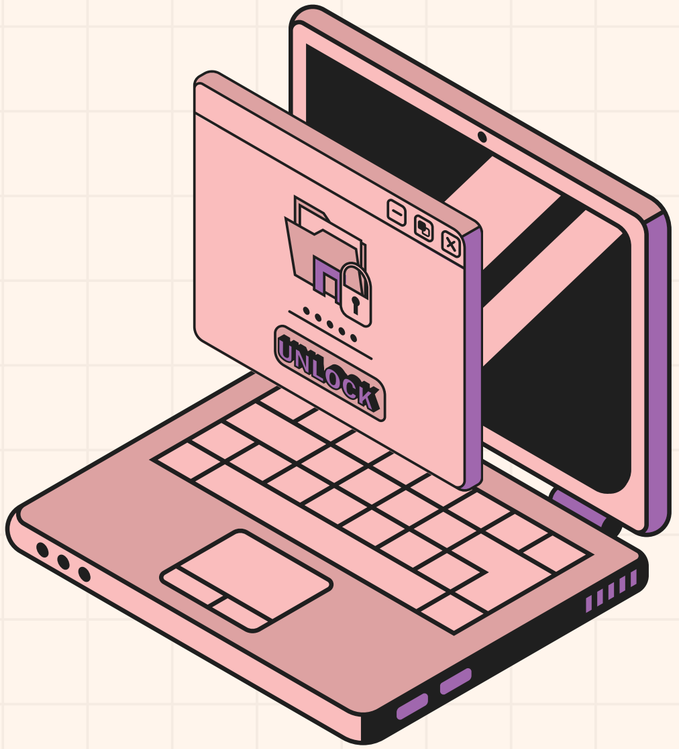
WHY USE IT?

- Balances simplicity and structure with a single deployment and clear domain separation.
- Easier development and debugging without network complexity.
- Better maintainability through well-defined module boundaries.
- Supports future evolution into microservices if needed, following the idea of Martin Fowler.

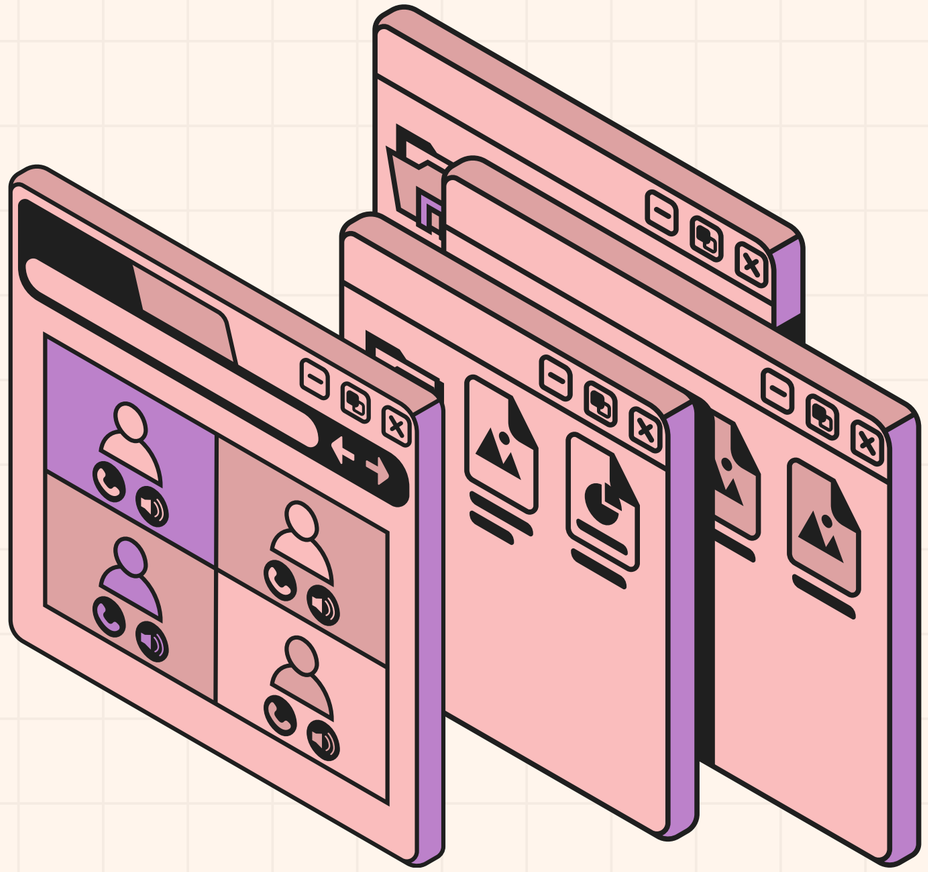


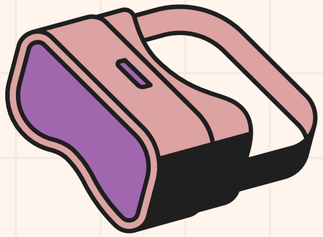
WHEN TO USE ?

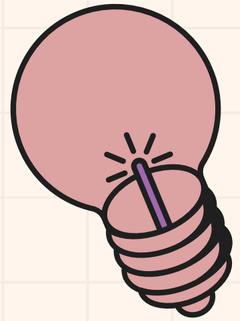
- Ideal for startups or growing projects that need fast development, simple deployment, and lower infrastructure costs without the overhead of microservices.
- Suitable for small to medium-sized teams (around 1–50 developers) .
- Appropriate when the system is small still needs clear domain separation for maintainability and future scalability.
- the system to evolve into microservices later if necessary .



Q&A







MEMBER

6731503034 WATTHANA JAIWANNA
6731503124 SIRAWIST JAISAMAK
6731503103 NUTTIDA BUTTHANOO
6731503105 THANCHANOK KAKAEW
6731503043 SUPITCHA CHUMMONGKON