



الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Week 8

كلية الهندسة المعلوماتية

مقرر بنیان البرمجيات

Microservices Architecture: introduction

د. رياض سنبل

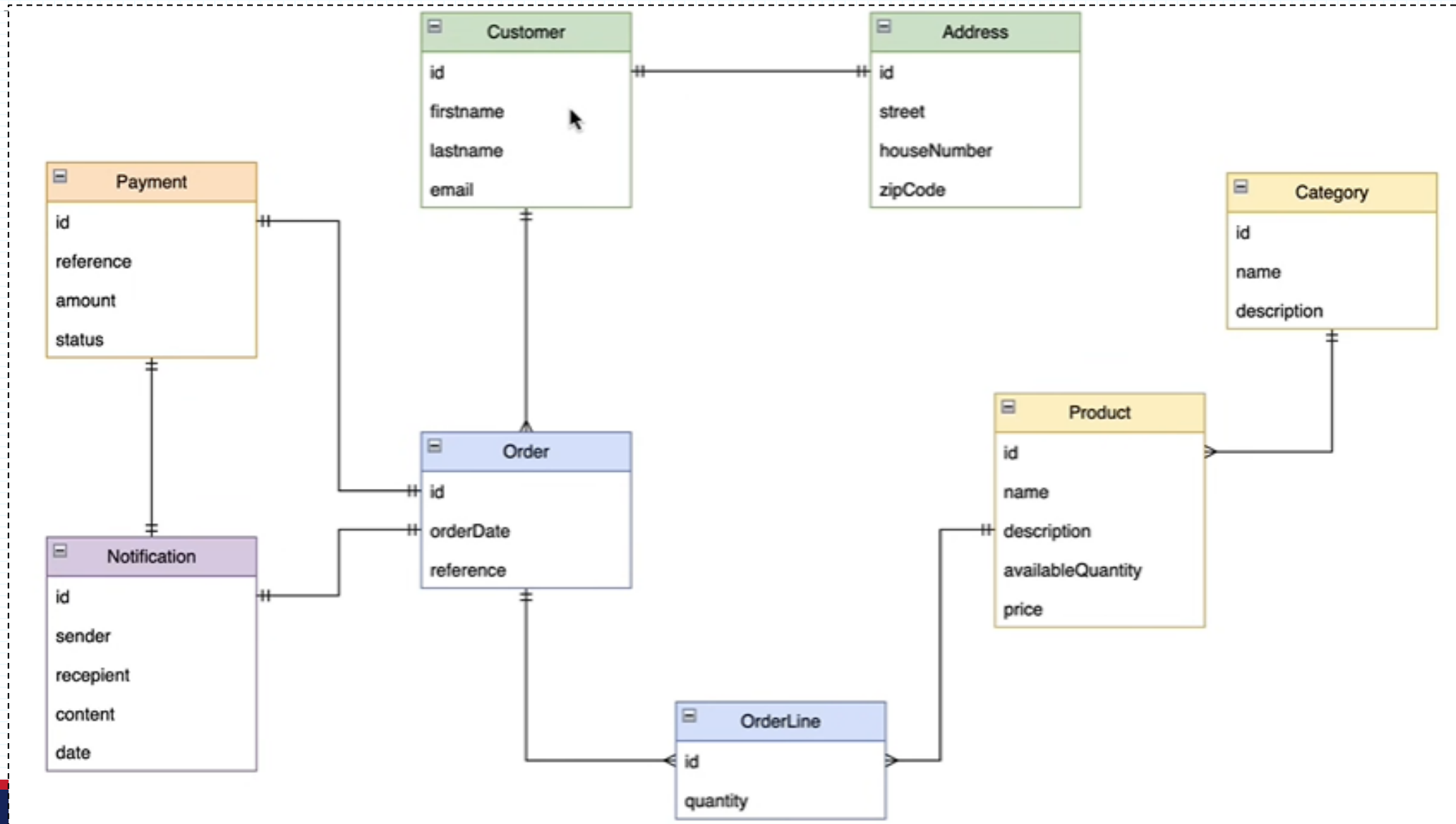
What is Domain-Driven Design?

- Domain-driven design (DDD) is a major software design approach, focusing on **modeling software to match a domain** according to input from that domain's experts.
- Under domain-driven design, the structure and language of software code (class names, class methods, class variables) should **match the business domain**
- Approach to software development focusing on **domain** complexity
- The term was coined by Eric Evans in his book of the same name published in 2003.

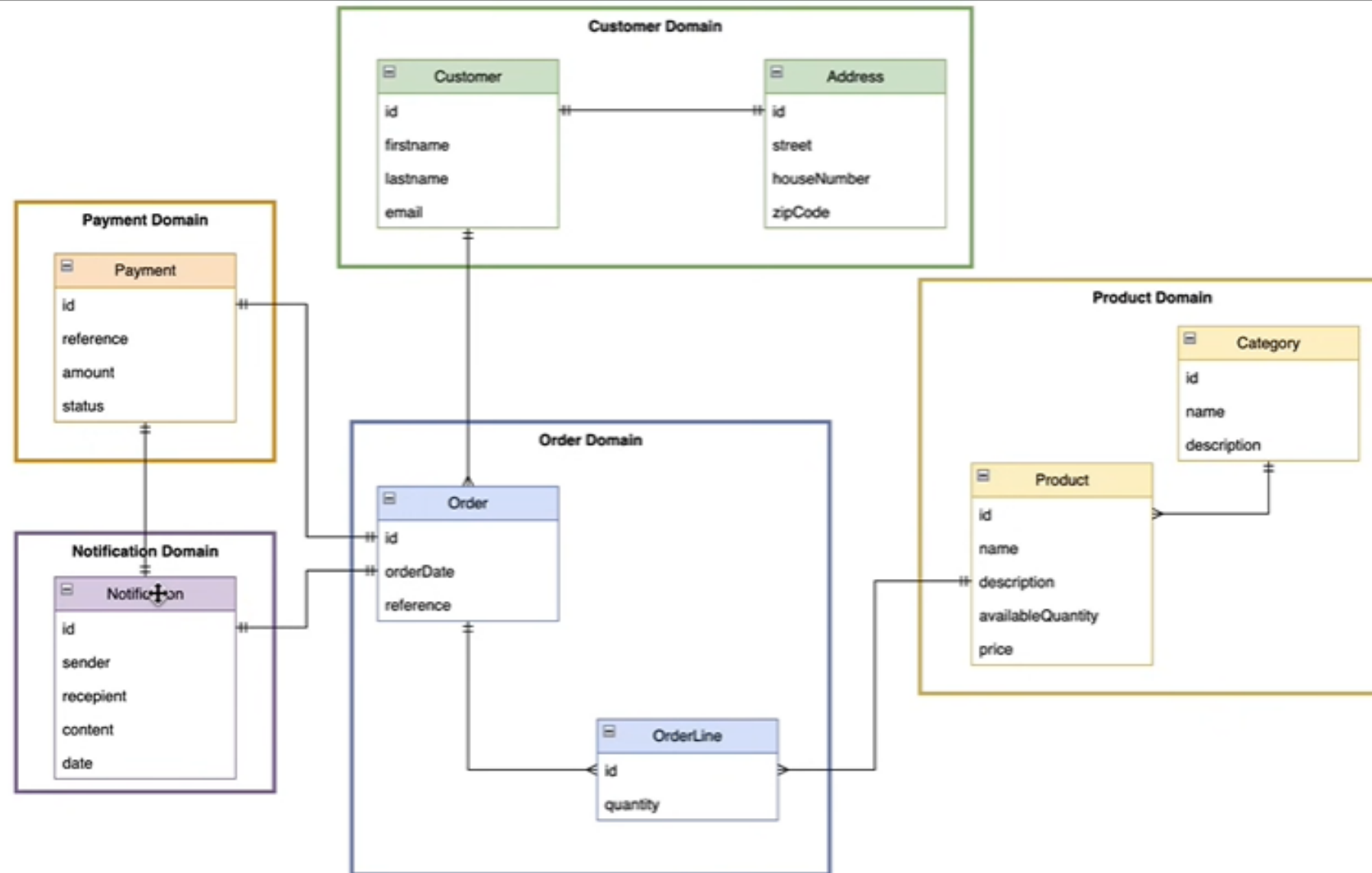
Why Use DDD?

- **Manages complexity:** DDD breaks down complex business logic into smaller, understandable models within clearly defined boundaries.
- **Clear communication using domain language:** It promotes a shared vocabulary (ubiquitous language) between developers and domain experts to avoid misunderstandings.
- **Aligns software with business goals:** DDD ensures the software design reflects real-world business processes and priorities.
- **Foundation for microservices architecture:** Each bounded context in DDD can map naturally to a microservice, encouraging modular and scalable systems.

Example

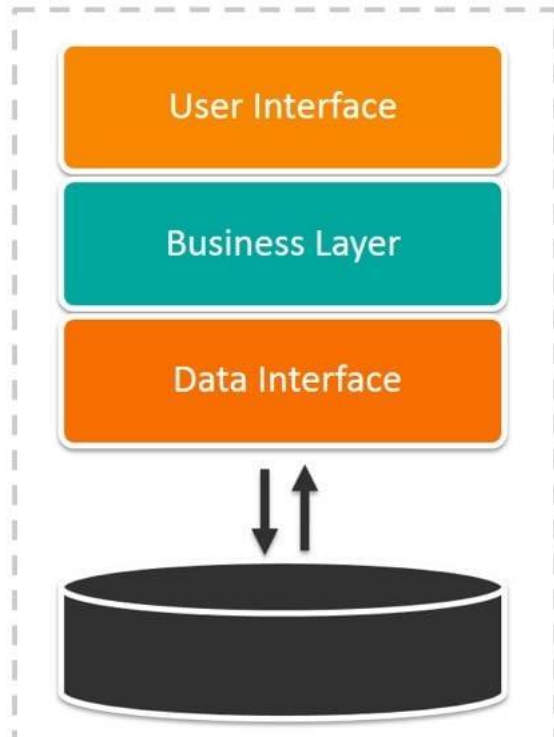


Example



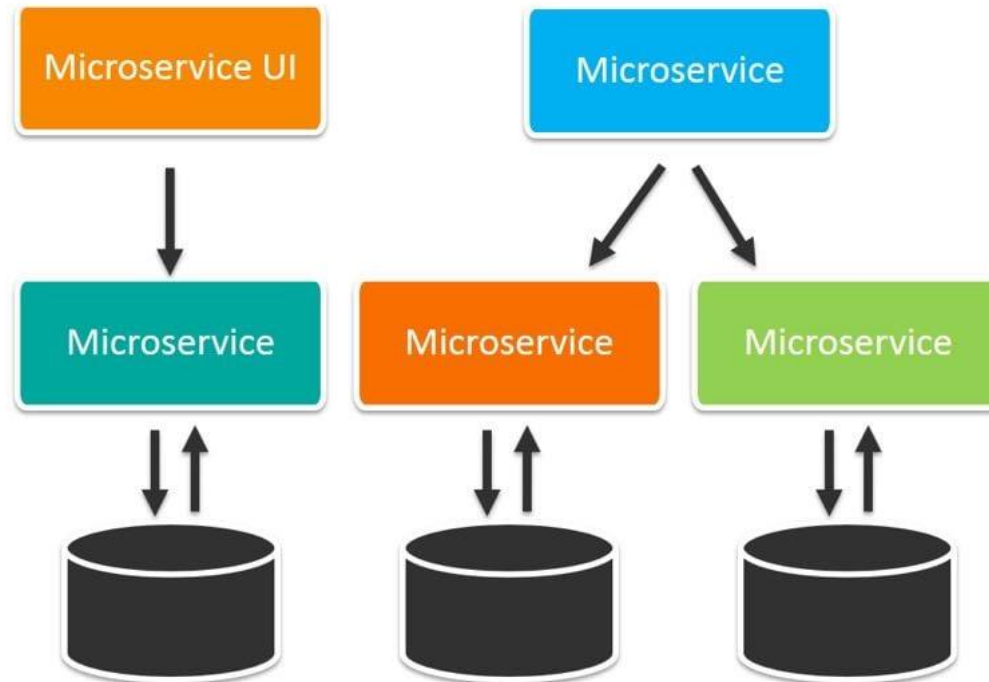
Monolithic vs Microservices

Monolithic Architecture



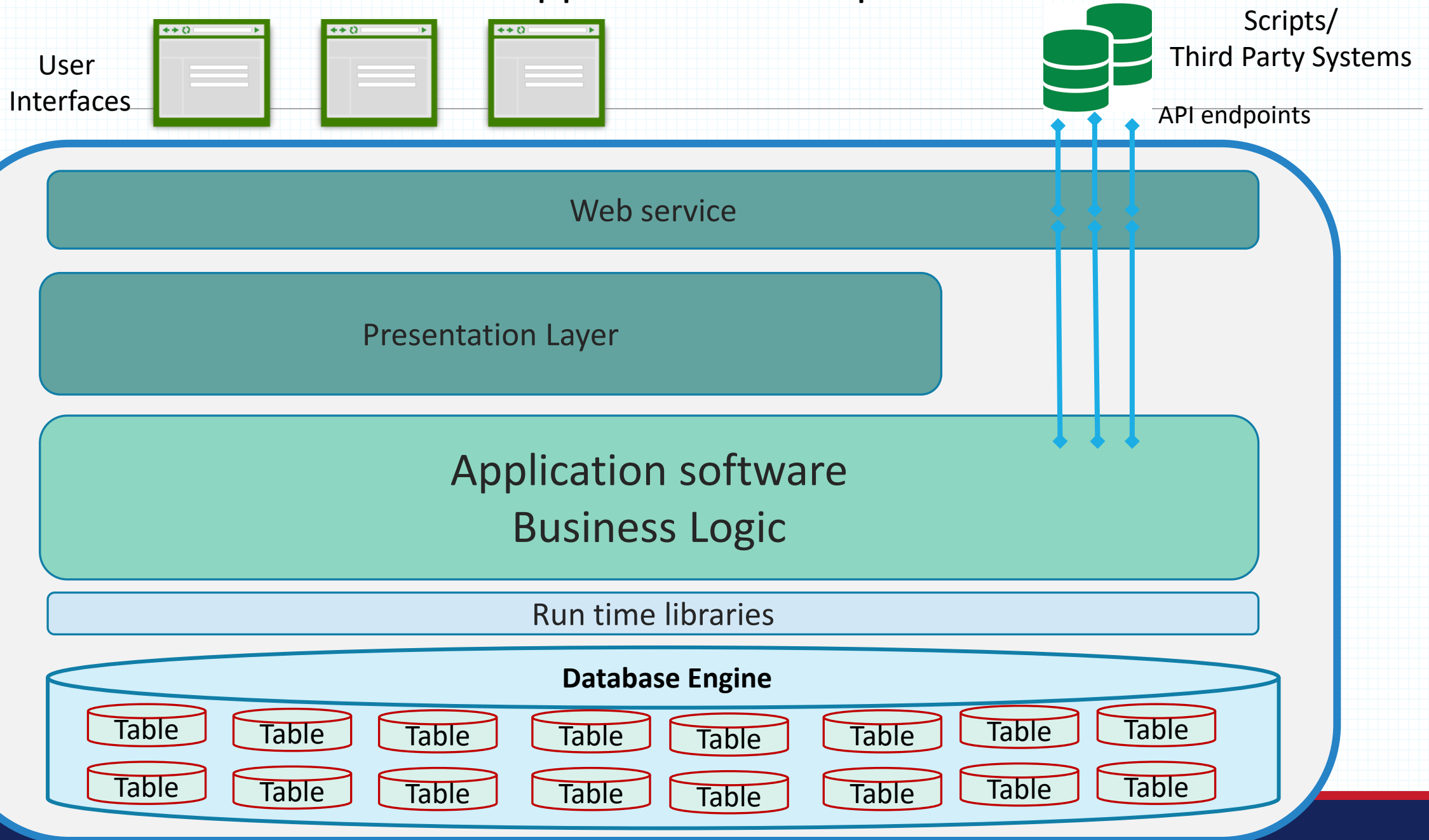
deployed as a single bundle of executables and libraries on a unified platform

Microservices Architecture

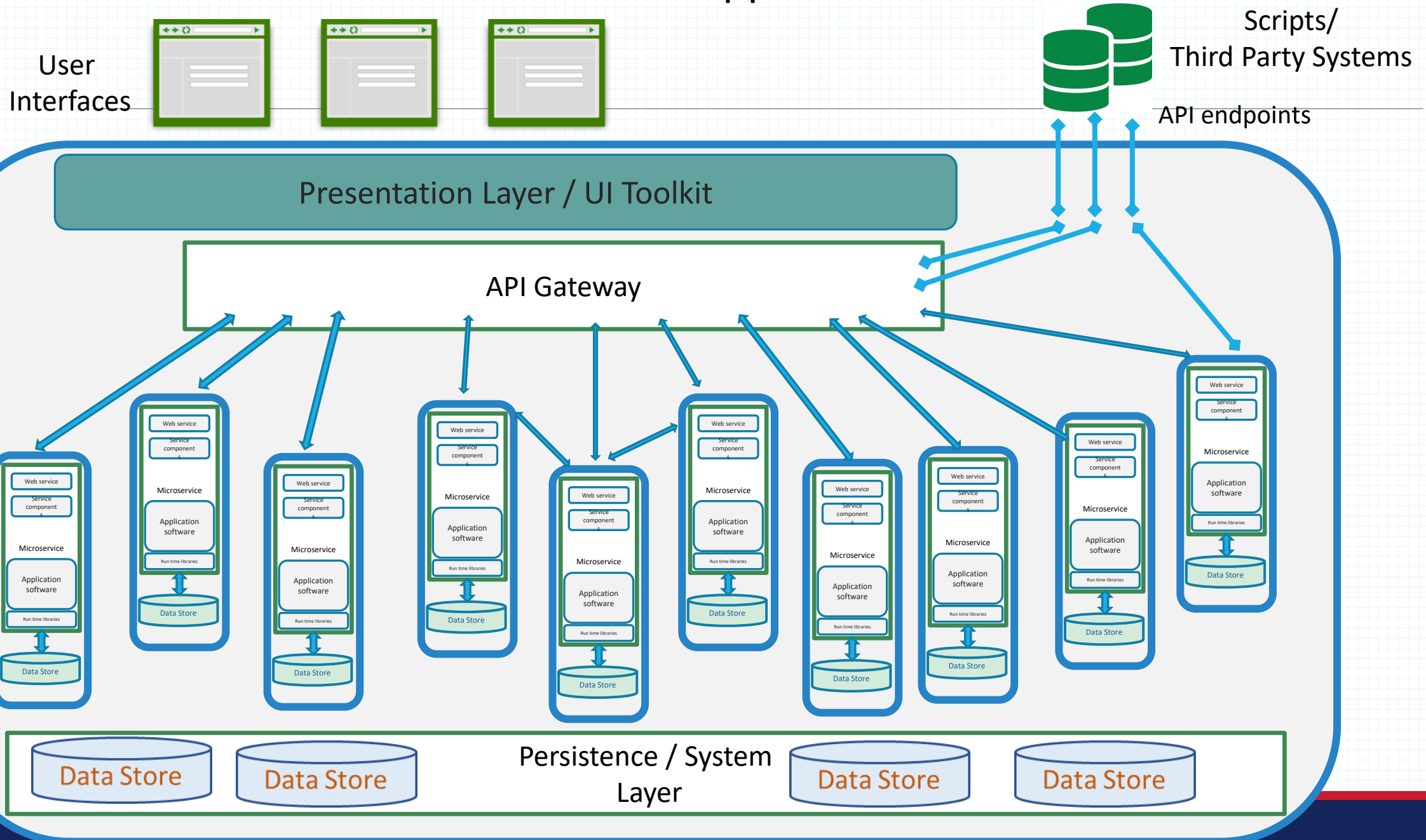


Multiple independent software components orchestrated to form a unified application

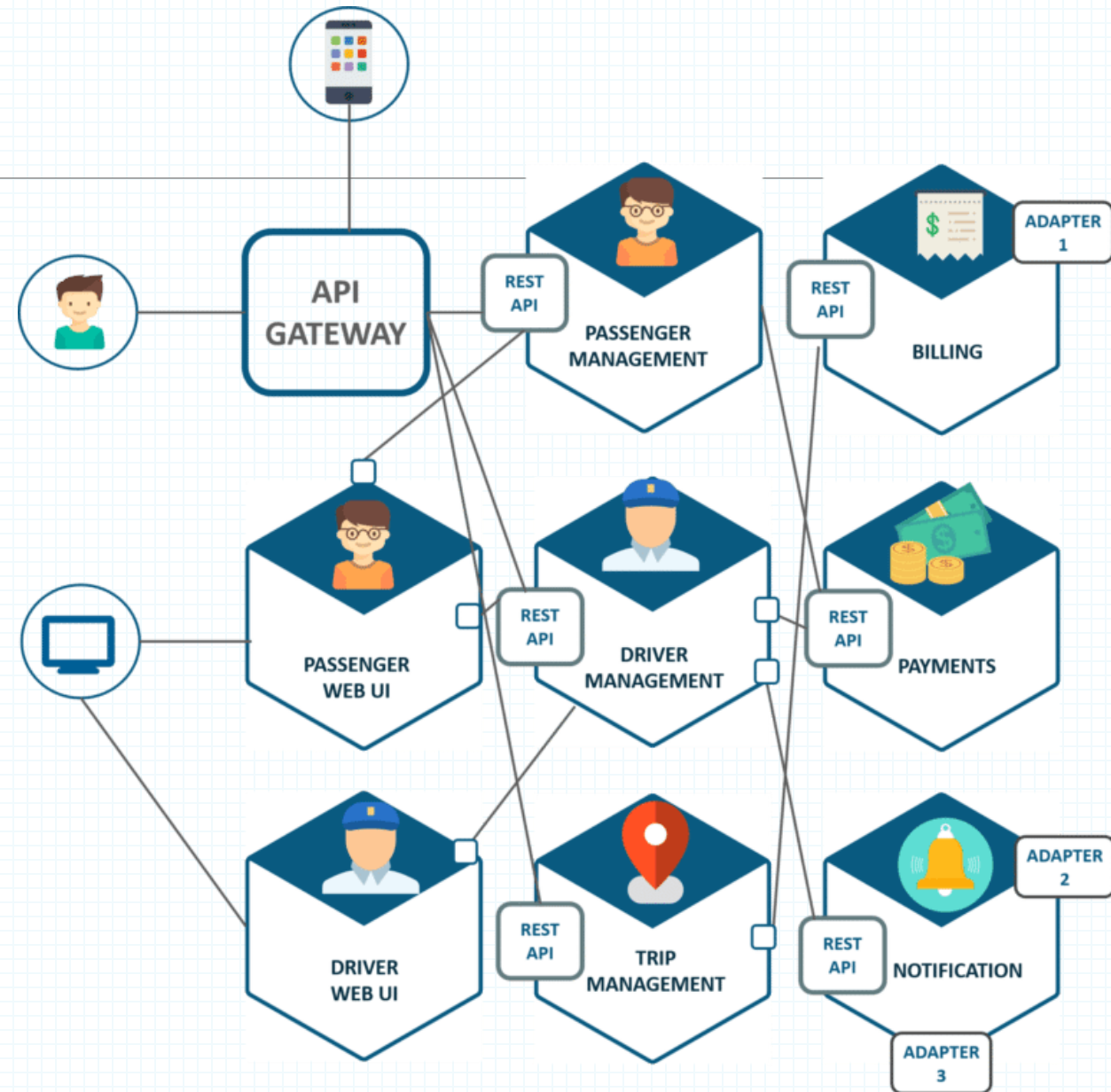
Monolithic Application Conceptual Model



Microservices-based Application



Example



Why Microservices!

- Successful applications often live a very long time + Technology changes
 - ⇒ Need to be able to easily “modernize” application!
 - ⇒ Need to deliver changes rapidly, frequently, and reliably.
- More complex Applications.. App keep growing up
 - ⇒ Need to **divide team**
 - ⇒ Need to improve **testability, deployability, maintainability, modularity, evolvability.**

Characteristics of Microservices

- **Independently deployable:** Each microservice can be developed, deployed, and updated without affecting other services.
- **Decentralized data management:** Every microservice owns and manages its own database, avoiding tight data coupling between services.
- **Technology agnostic:** Microservices can be built using different programming languages, databases, and frameworks, depending on the service's needs.
- **Focused on a single business capability:** Each microservice is designed to handle one specific business function, making the system modular and easier to understand.