Software Architecture Masterclass

Understanding Software Architecture & Design

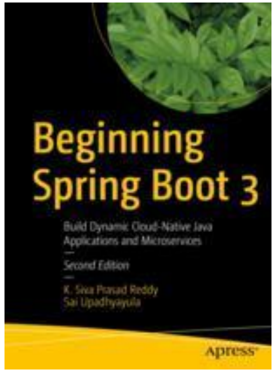
By

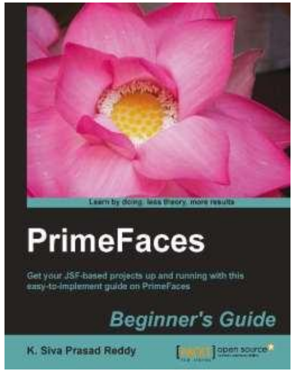
K. Siva Prasad Reddy

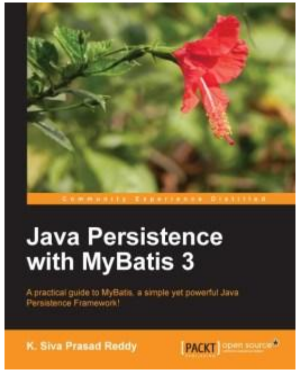
About Me

K. Siva Prasad Reddy

- Software Developer since 2006
- Published Author
- Blog: www.sivalabs.in
- Twitter: @sivalabs
- YouTube: youtube.com/sivalabs




Learn by doing: less theory, more results
PrimeFaces
Get your JSF-based projects up and running with this easy-to-implement guide on PrimeFaces
Beginner's Guide
K. Siva Prasad Reddy


Java Persistence with MyBatis 3
A practical guide to MyBatis, a simple yet powerful Java Persistence Framework!
K. Siva Prasad Reddy
PACKT open source

Introduction

- Why should I learn about Software Architecture?
- Who are the target audience?
- What is the goal of this presentation?

Session 1 - Agenda

- Introduction to Software Architecture
- Software Architecture Vs Design
- The Role of an Architect
    - A bridge between Biz and Tech
    - A torchbearer
    - Hands-on?
- The Hard Parts
    - No silver-bullet solution
    - Gauging Trade-Offs
    - Selecting architecture-design for the given context

Session 2 - Agenda

- Architecture Styles
    - Monoliths
    - Microservices
    - Modular Monoliths
    - Event Driven
    - CQRS & Event Sourcing

Session 3 - Agenda

- Software Design
  - Layered Architecture
  - Clean/Hexagonal Architectures
  - Domain Driven Design

Session 4 - Agenda

- Team Workflow Standardization
- Documenting Architecture
- Enforcing Architecture Principles
- Deployment Considerations

DAY 1

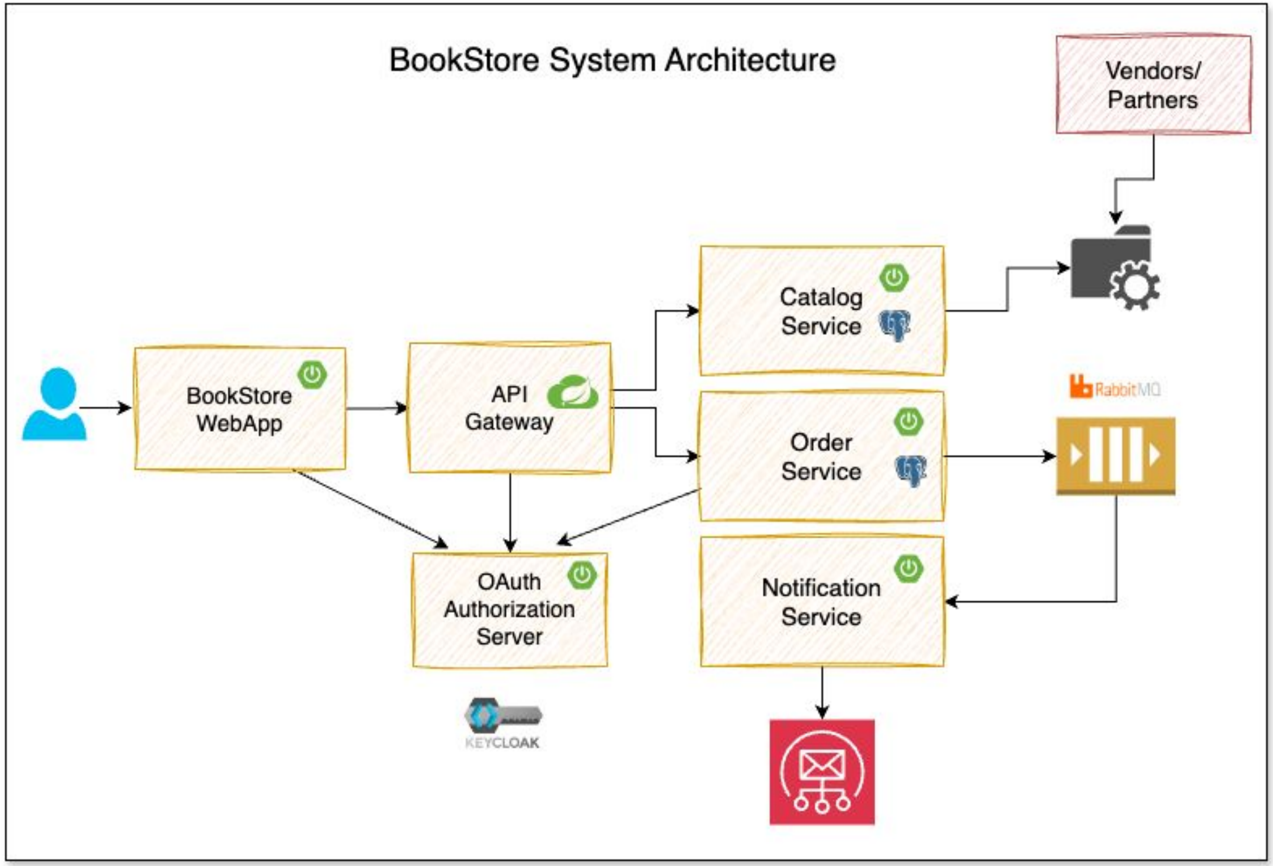
Software Architecture

What is Software Architecture?

Software Architecture is the organization of a system that includes all components, how they interact with each other, the environment in which they operate, and the principles used to design the software.

Source: https://www.castsoftware.com/glossary/what-is-software-architecture-tools-design-definition-explanation-best

Software Architecture Diagram



Software Architecture (cont…)

Difference between Software Architecture & Design?

- **Software Architecture** defines what are all the components in a system and how they interact with each other.
- **Software Design** represents implementation specific principles used to design the components.

Software Architecture (cont…)

- Who is a Software Architect?
    - A bridge between Biz and Tech
    - A torchbearer
    - Hands-on??
- Roles and Responsibilities of Software Architect
    - Evaluate functional requirements
    - Gathering non-functional requirements (NFRs)
    - Design high-level architecture
    - Selecting Tech Stack and Deployment Strategy
    - Compliance

Software Architecture (cont...)

- The Hard Parts
  - No silver-bullet solution
  - Gauging Trade-Offs
  - Selecting architecture-design considering the context and limitations

Software Architecture (cont…)

- Exercises
  - Design a shopping cart application for a small-medium sized business
  - Build an enterprise e-commerce application for a large company with millions of active users

Software Architecture (cont…)

- Software Architecture Considerations
  - Size of the user base
  - Expected scalability
  - Buy vs Build vs Open Source Solutions
  - Security
  - Performance & SLAs
  - Resilience and Time to recover
  - Cost Effectiveness
  - Time to market
  - Partner/3rd party integrations
  - Tech Stack

Q & A

DAY 2

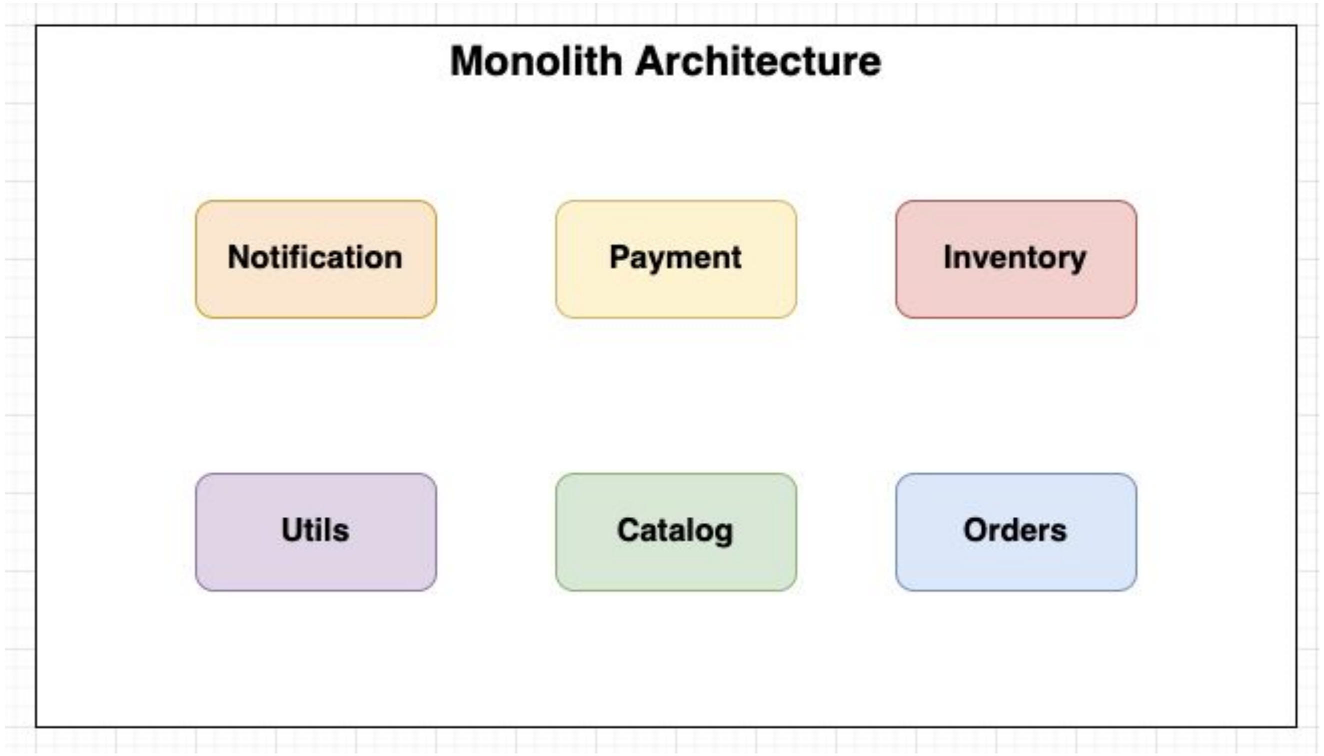
Architecture Styles

- Monoliths
- Microservices
- Modular Monoliths
- Event Driven
- CQRS & Event Sourcing

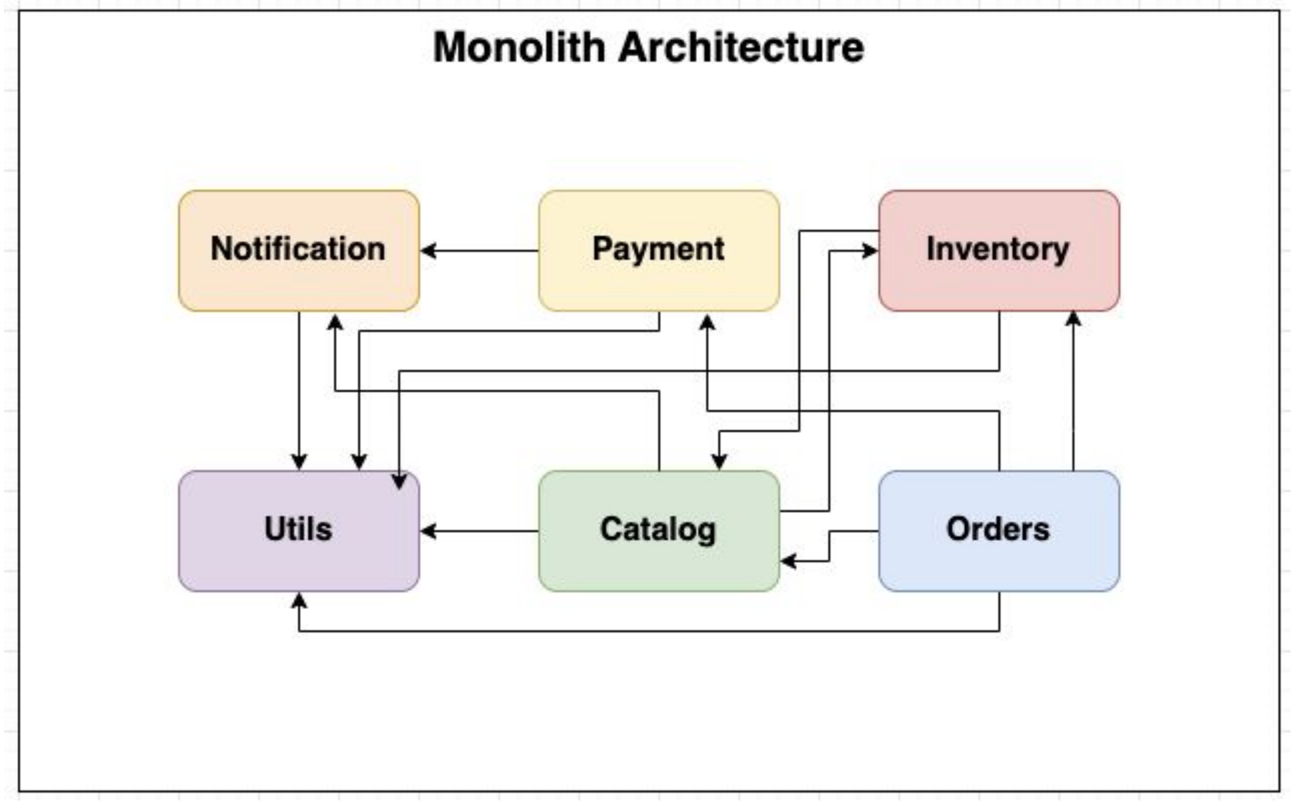
Monolith Architecture

- Single Deployment Unit
- Misconception: Monoliths are Bad
- Monoliths can become “Big Ball of Mud” unless care is taken
- Team Coordination Overhead
- Relatively simple deployment process
- Can’t scale a specific sub-module/system

Monolith Architecture



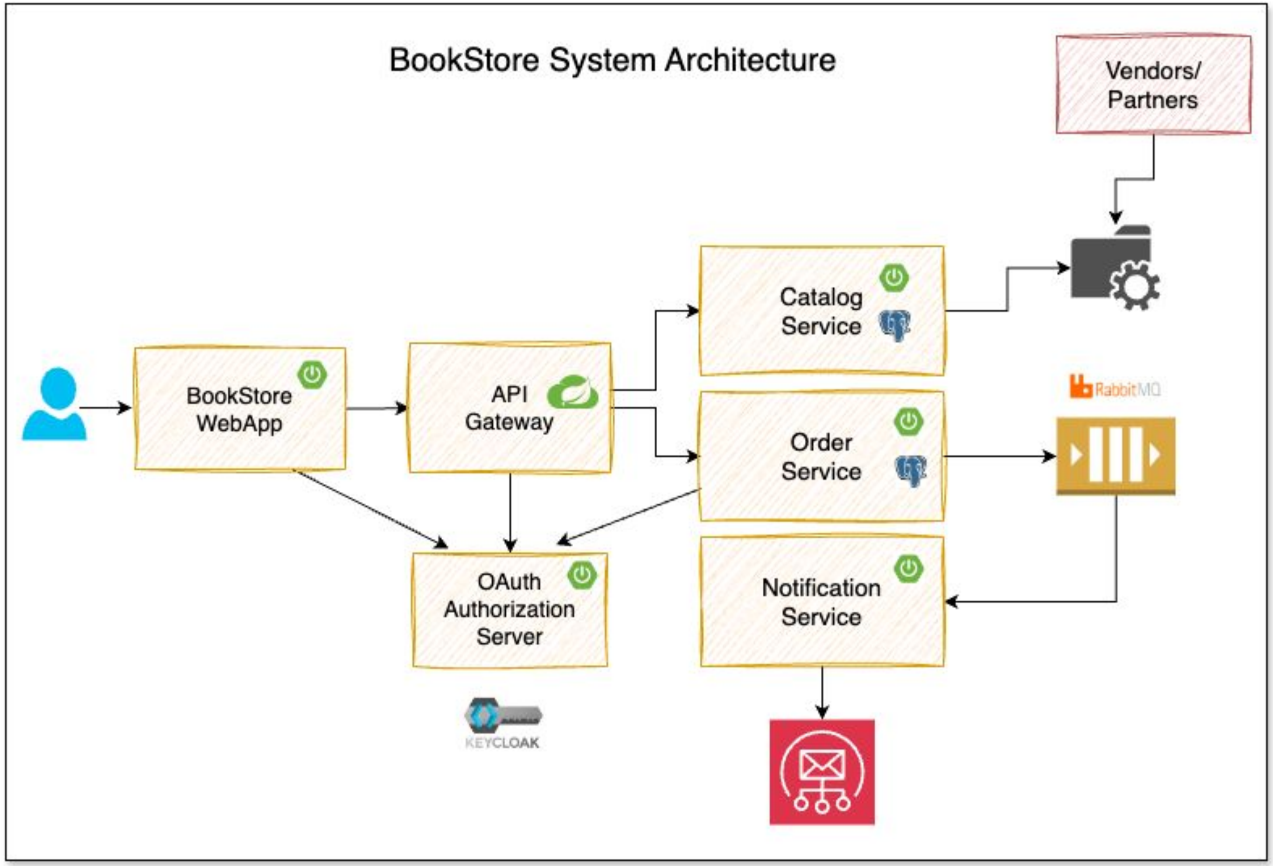
Monolith Architecture (Reality)



Microservices Architecture

- A microservice is an independently deployable unit proving a business capability
- Misconception: Microservices is the solution to fix Monolith problems
- Microservices can become “Distributed Big Ball of Mud”
- Can scale a specific sub-module/system based on usage pattern
- Each Microservice should be designed as self-sufficient as possible
- Prefer asynchronous processing over synchronized communication
- Brings its own (inherent distributed) challenges:
    - Centralised log management
    - Distributed Tracing
    - Complex Testing and Deployment process
    - Design for failures

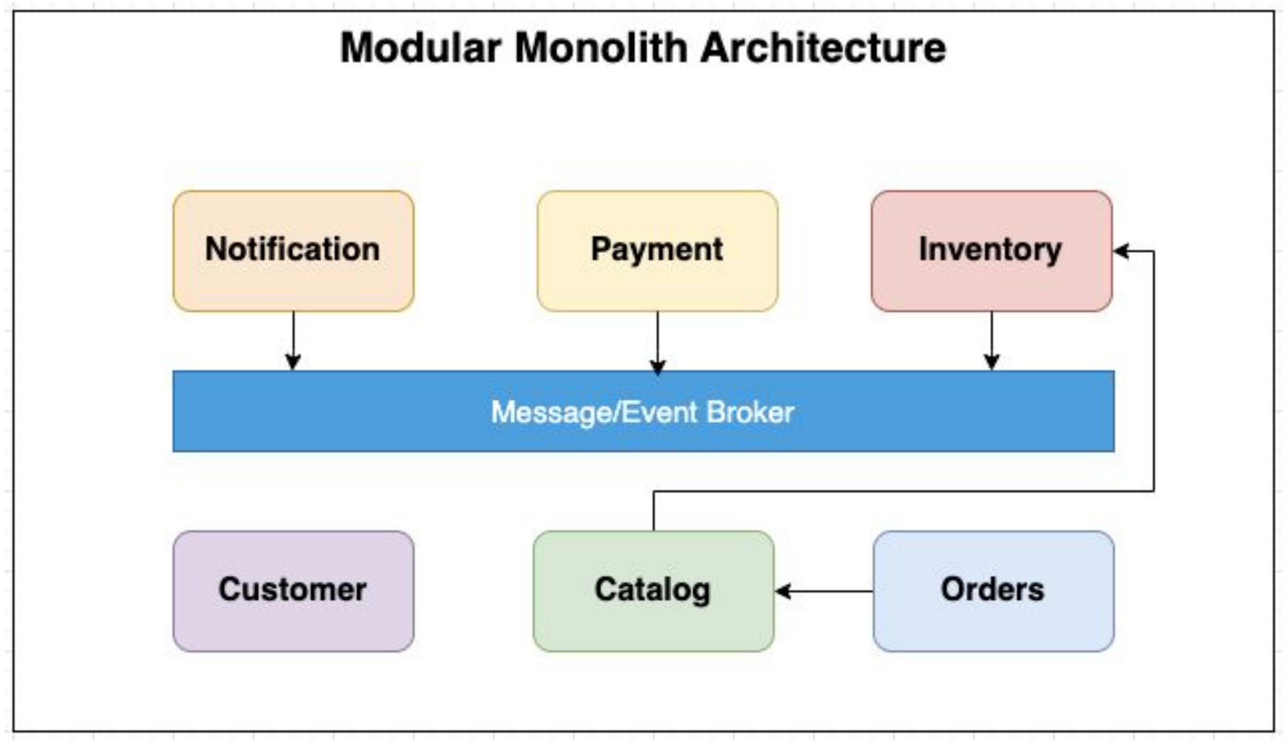
Microservices Architecture



Modular Monolith Architecture

- Build with **strict modularization**, but **deploy as a single deployable unit**
- Easy to refactor as the understanding of module boundaries become clear
- Strict modularization can avoid “Big Ball of Mud”
- A progressive approach to Microservices
- Different teams can work on different modules
- Inter-module communication using events

Modular Monolith Architecture



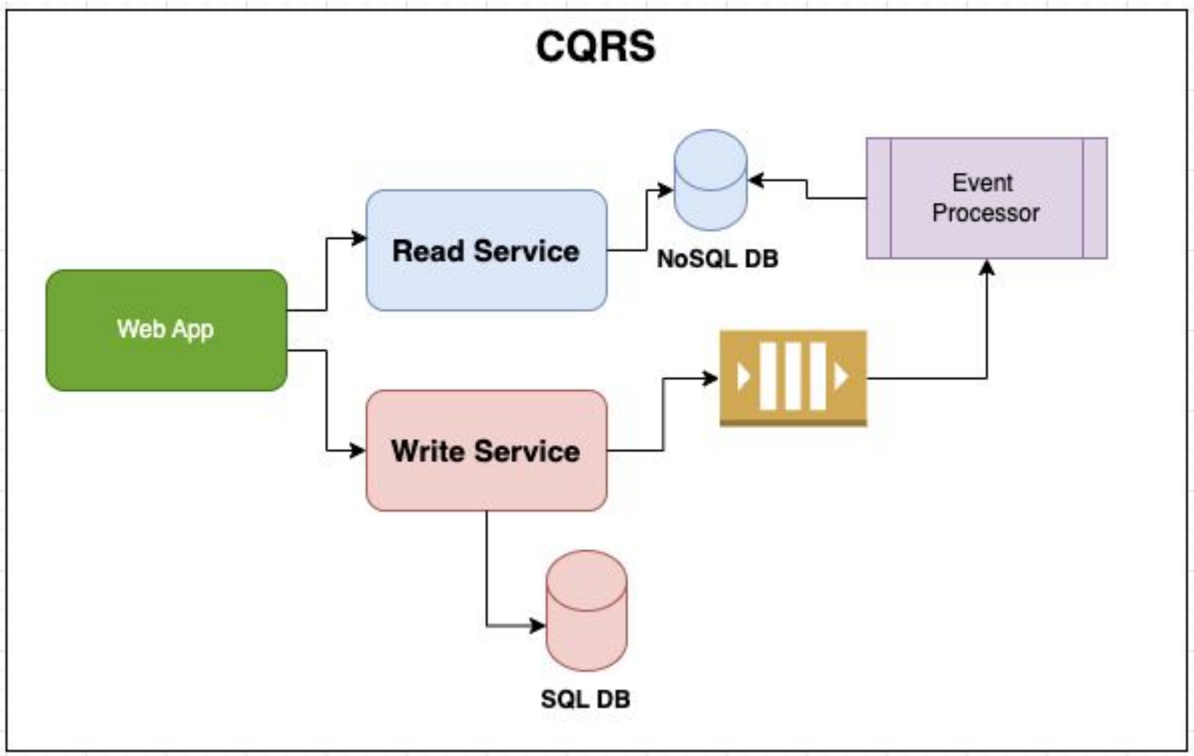
Event Driven Architecture

- Asynchronous processing using events
- Loose coupling between modules
- Ensuring Business Transaction Integrity using Saga Pattern:
    - Orchestration
    - Choreography
- Challenges
    - Guaranteed delivery
    - Exactly once delivery
    - Handling events in orders, idempotency

CQRS Architecture

- CQRS (Command Query Responsibility Segregation)
  - Separate Reads from Writes
  - Use optimized storage and processing for Reads & Writes
  - Independently scalable
  - Eventual consistency
  - Optimized for performant reads
- Event Sourcing
  - Store entity state changes as a sequence of events
  - Captures history of events providing insights into user activity
  - Eventual consistency
  - Overkill for simple CRUD applications

CQRS Architecture



Q & A

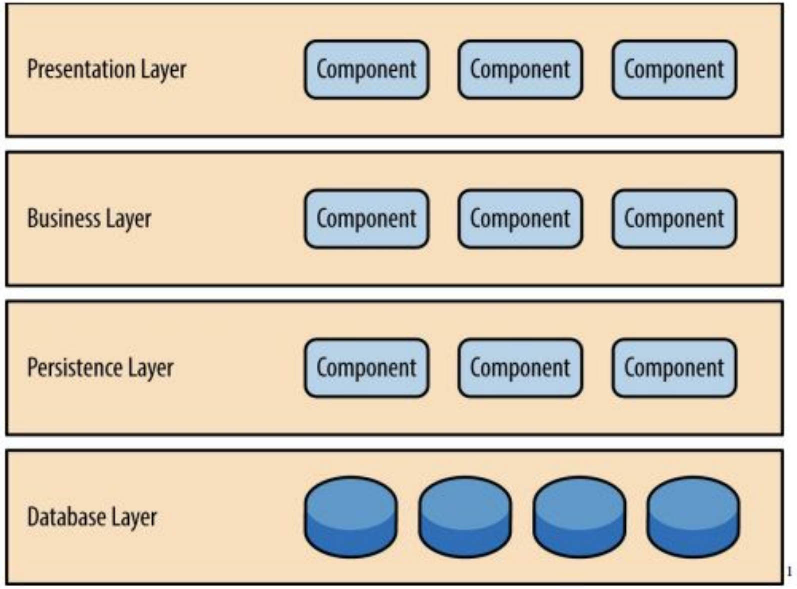
DAY 3

Software Design

- Layered Architecture
- Clean/Hexagonal/Onion/Ports-and-Adapters Architecture
- Domain Driven Design (DDD)

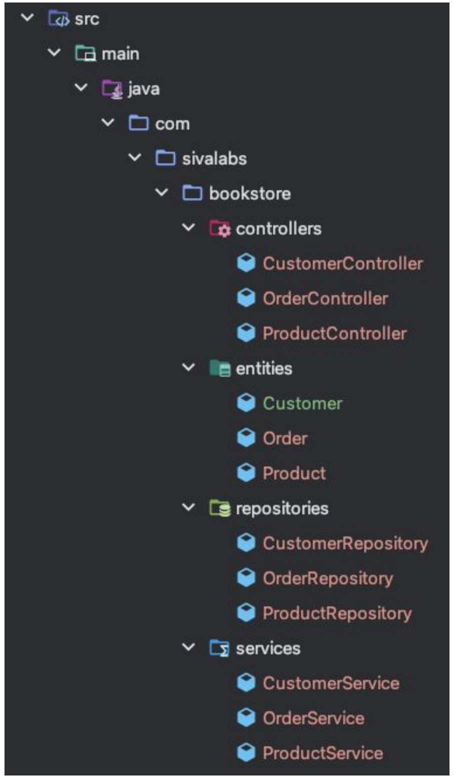
Layered Architecture

- Layered Architecture
  - Separation of concerns
  - Anemic Domain Model
  - Transaction Script Pattern



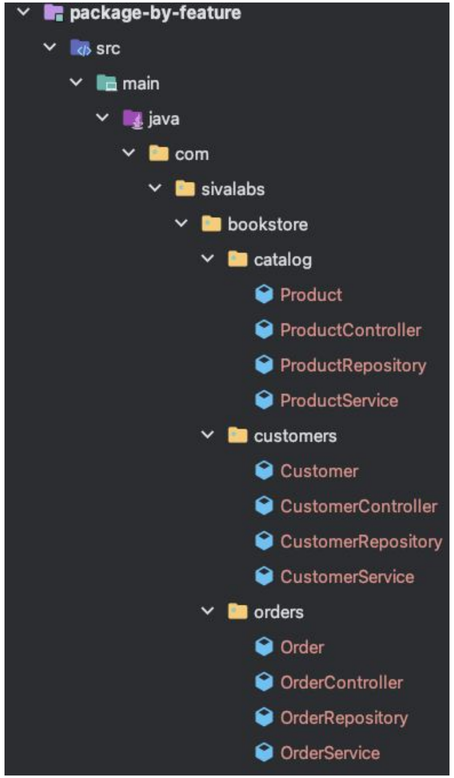
Layered Architecture

Package By Layer



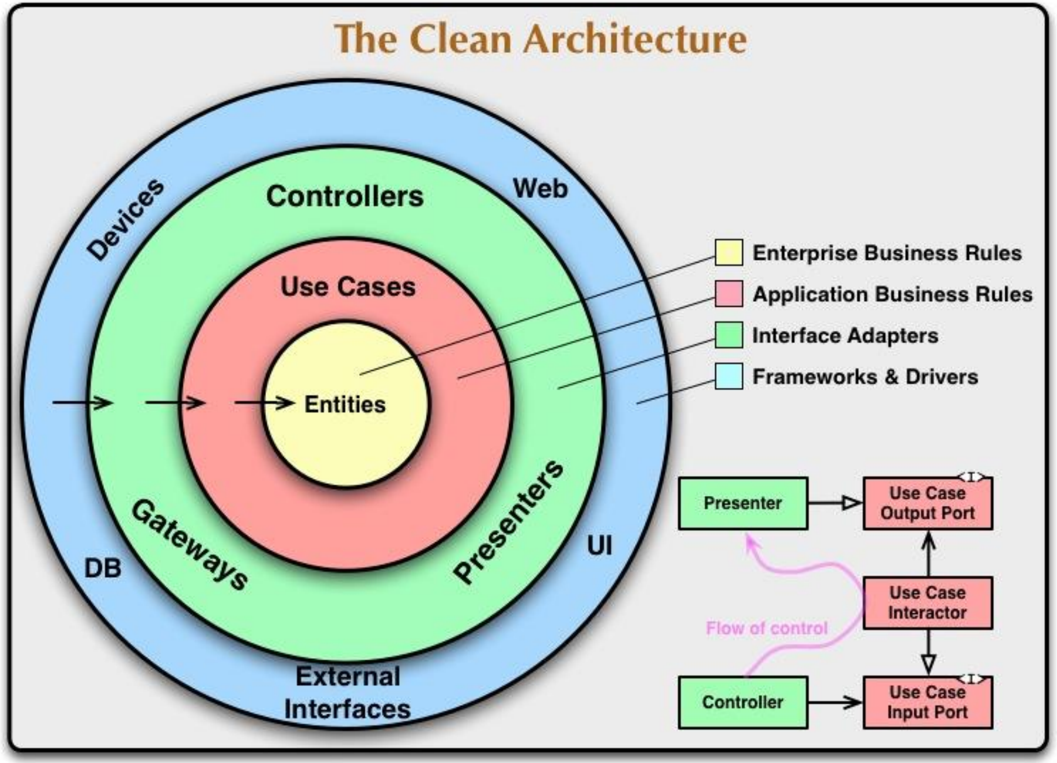
```
src
  main
    java
      com
        sivalabs
          bookstore
            controllers
              CustomerController
              OrderController
              ProductController
            entities
              Customer
              Order
              Product
            repositories
              CustomerRepository
              OrderRepository
              ProductRepository
            services
              CustomerService
              OrderService
              ProductService
```

Package By Feature



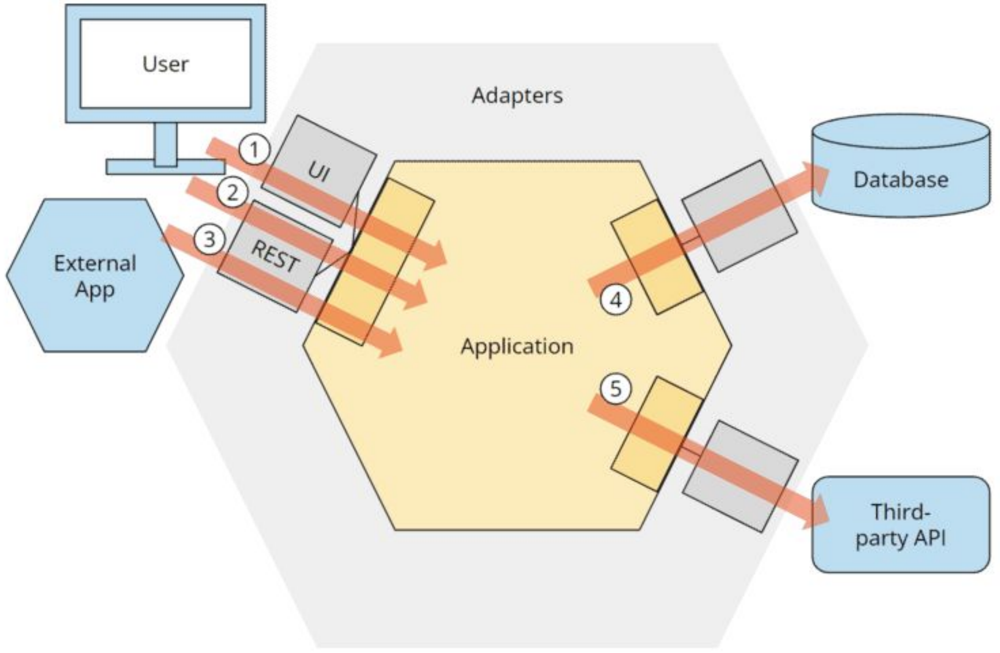
```
package-by-feature
  src
    main
      java
        com
          sivalabs
            bookstore
              catalog
                Product
                ProductController
                ProductRepository
                ProductService
              customers
                Customer
                CustomerController
                CustomerRepository
                CustomerService
              orders
                Order
                OrderController
                OrderRepository
                OrderService
```

Clean Architecture



Source: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html

Hexagonal Architecture



Hexagonal architecture with control flow

Source: https://www.happycoders.eu/software-craftsmanship/hexagonal-architecture/

Domain Driven Design (DDD)

- An approach to software design that reflects the domain in the real world
- **Talk to the domain experts** and use **Ubiquitous Language**
- Identity **Bounded Contexts**

    - **Aggregate**
    - **Entity**
    - **Value Object**
    - **Repository**
    - **Domain Event**
    - **Domain Service**
    - **Application Service**
    - **Module**

Q & A

DAY 4

Team Workflow Standardization

- Team Workflow
  - Git Branching Strategy
  - Code Review Process
  - Feedback Cycle
  - Time to Demo
- Standardization of Tools
  - IDE
  - Local Development Setup
  - Docker, Docker Compose, Skaffold

Documenting Architecture

- Architectural Decision Records
- C4 Model Diagrams

Enforcing Architecture Principles

- Coding Standards and Bug Pattern Detectors
  - CheckStyle
  - PMD, SpotBugs
  - ErrorProne
  - SonarQube
- **ArchUnit** - Enforce architecture guidelines as tests
- **Spring Modulith** - Enforce modularity in Modular Monoliths
- DevOps, DevSecOps - CI/CD Automation
- Observability - Continuous Monitoring and Time to react

Deployment Considerations

- On-Prem or Cloud
- Kubernetes
- Platform Engineering

Q & A