Architecture Style : Modulith (vs Microservices)

Pier-Jean Malandrino

Published in Level Up Coding

4 min read · Jan 28, 2024

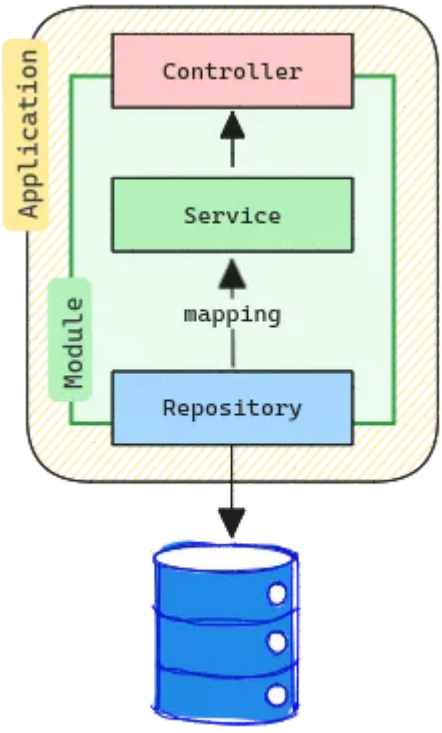
What is Modulith Architecture?

Modulith architecture is a style of software design that emphasizes modularity within a monolithic application. It aims to combine the simplicity and straightforward deployment model of a monolithic architecture with the modularity and maintainability typically associated with microservices.

In a modulith, the application is structured as a collection of loosely coupled modules, each encapsulating a specific business capability or domain. These modules interact with each other through well-defined interfaces, yet they are deployed as a single unit, similar to a traditional monolithic application.

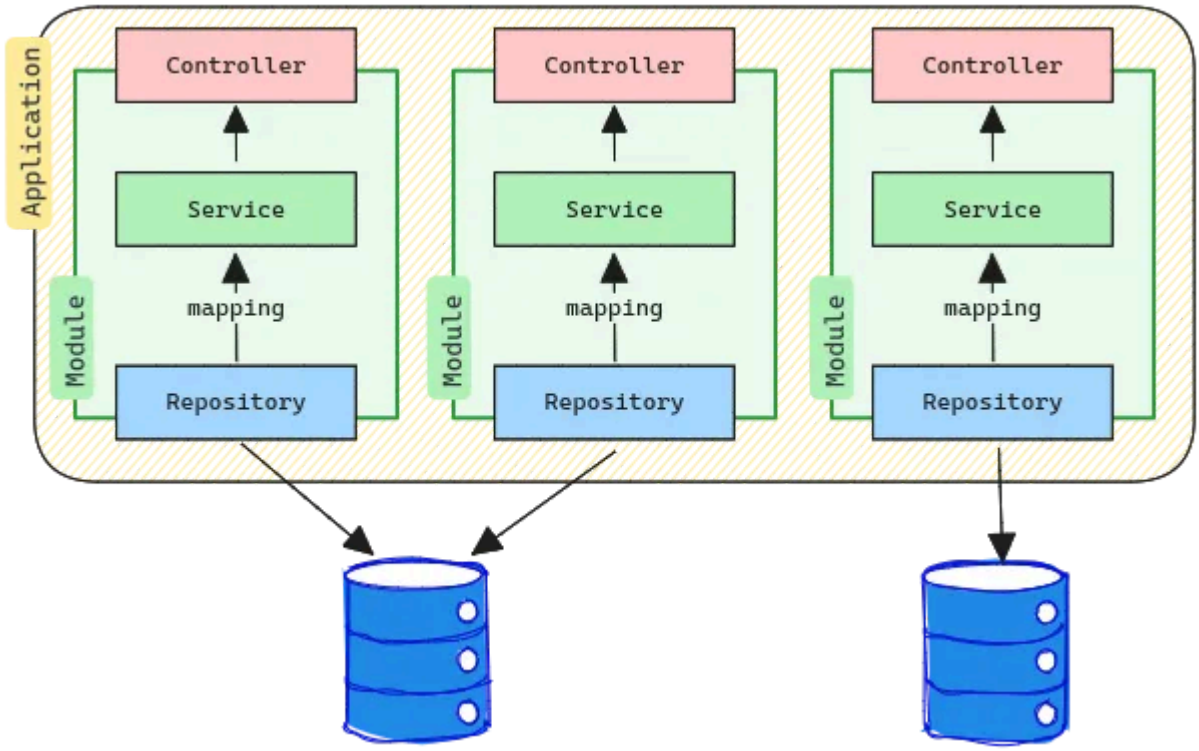
> *In a modulith, the application is structured as a collection of loosely coupled modules, each encapsulating a specific business capability or domain.*

So a monolithic application would look like this :



Monolithic application

So a modulithic application would look like this :



Modulith application

Benefits

Enhanced Modularity

Moduliths promote clean separation of concerns by organizing code into distinct modules. This separation enhances the maintainability and understandability of the codebase, making it easier for teams to manage large and complex applications.

Simplified Deployment

Unlike microservices, which require complex orchestration for deployment, moduliths are deployed as a single unit. This simplifies the deployment process and reduces the operational overhead associated with managing multiple services.

No network overhead

Moduliths operate without the additional network overhead typical in microservices. This is due to their internal module communication being in-process, eliminating the latency and complexity associated with network calls between separate services.

Fit well with a DDD approach

Modulith architecture aligns well with Domain-Driven Design (DDD). It naturally supports bounded contexts by allowing each domain model to be encapsulated within its own module, fostering a clear domain model and business logic separation.

Trade-Offs

Potential for Tight Coupling

While moduliths aim for loose coupling between modules, there's a risk of inadvertently introducing tight coupling, which can lead to challenges in module isolation and independent scaling. Even with modular separation, coupling is not guaranteed as it is with microservices.

Complexity in Scaling

Moduliths may not scale as efficiently as microservices in certain scenarios. Scaling a modulith often means scaling the entire application rather than individual components, which can be less efficient.

Technology Stack Limitations

In a modulith, the entire application typically shares a common technology stack. This can limit the flexibility to use different technologies or programming languages best suited for specific modules, as often done in a microservices architecture.

Spof

Since moduliths are deployed as a single unit, they represent a single point of failure. If one module encounters a significant issue, it can potentially impact the entire application, unlike microservices where failure is usually isolated.

Modulith or Microservices ?

When deciding between modulith and microservices architectures, the key factor is the level of coupling you're comfortable with.

More coupling simplifies maintenance but comes with trade-offs, like scalability complexity.

For instance, if part of your application faces heavy loads distinct from the rest, a microservices approach could allow for targeted scaling. Choosing different frameworks or languages also justifies using microservices.

However, for isolating domains within an app, like products and clients, a modulith can be effective. It keeps these domains together for versioning and lifecycle management, simplifying CI/CD and database maintenance, while still maintaining a manageable level of coupling through modularization.

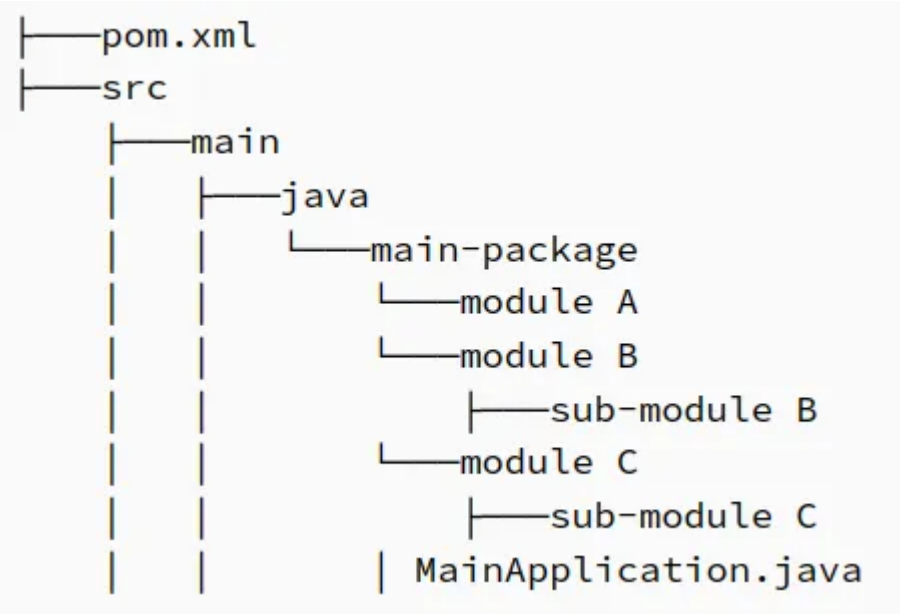
Ultimately, the best choice depends on your specific needs, and often, a combination of both approaches works well.

> *Often, a combination of both approaches works well.*

Spring Modulith Implementation

Overview

Spring Modulith is an approach for implementing the modulith architecture using the Spring framework. It is designed to help developers structure their Spring applications in a modular way, following the principles of modulith architecture.



Example from Baeldung

Key Features

Module Definition: Spring Modulith allows defining modules within a Spring application. Each module encapsulates its own business logic, data access, and Spring components.

Inter-module Communication: It provides mechanisms for modules to communicate with each other through events or shared interfaces, maintaining loose coupling. It also could be done using event between modules.

Note : If you make an interaction between modules that is not correct, the compilation will not fail, you need to implement a test that will fail to prevent that kind of usage. This is based on the ArchUnit tests.

Module Isolation: While each module is part of the same monolithic application, Spring Modulith enforces boundaries to prevent unintended dependencies and tight coupling.

Testing and Development: Spring Modulith supports testing at the module level, enabling developers to write and run tests for individual modules without the need for the entire application context.

More information here : https://www.baeldung.com/spring-modulith

Conclusion

In conclusion, the Modulith architecture offers a balanced approach to application design, blending monolithic simplicity with microservices' modularity. It suits scenarios where domain isolation within a single application is required.

While moduliths enhance modularity, reduce deployment complexity, and align well with Domain-Driven Design, they face challenges in scaling and technology flexibility.

The decision between moduliths and microservices hinges on the acceptable level of coupling, with a hybrid approach often being effective.

Spring Modulith specifically caters to the Spring framework, facilitating module definition, inter-module communication, and isolation, while supporting effective testing strategies.

Technology | Microservices | Software Development | Programming | Software Architecture

Edit profile

Written by Pier-Jean Malandrino

3.5K Followers · Writer for Level Up Coding

CTO & Head of Architecture at a digital firm, I drive technological strategy, design innovative solutions, & lead R&D projects.

More from Pier-Jean Malandrino and Level Up Coding

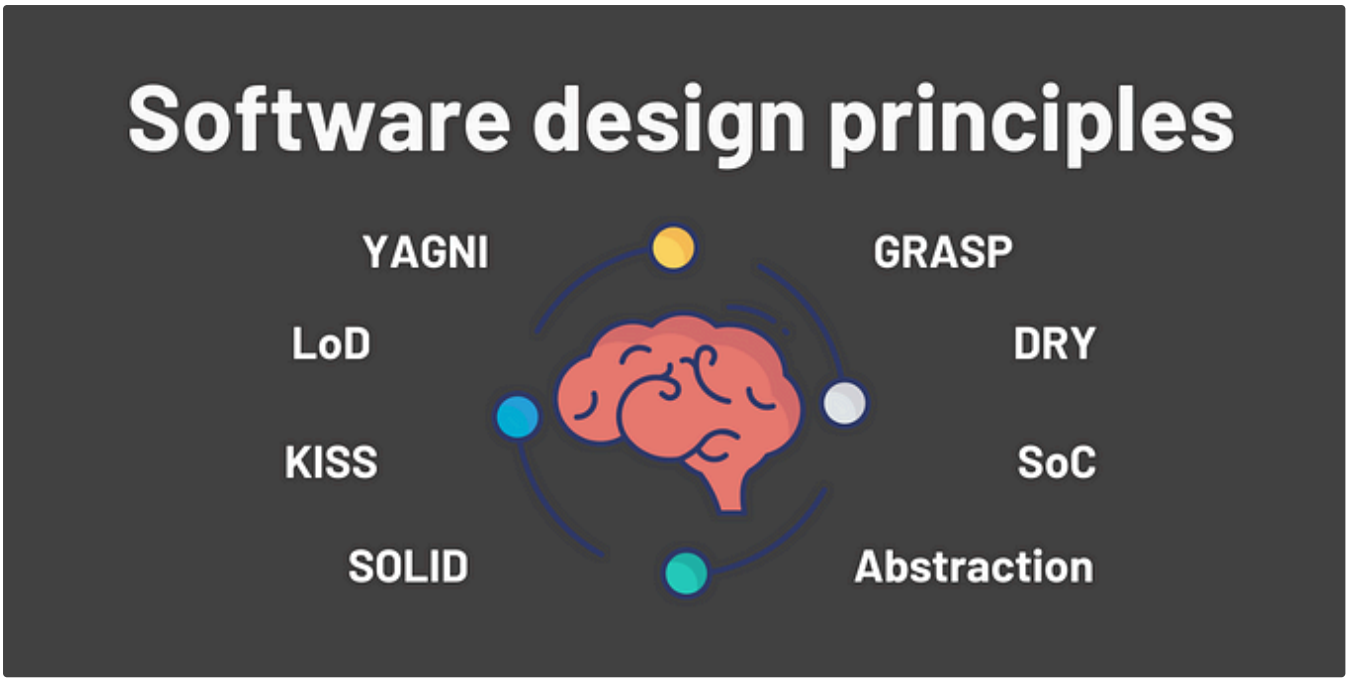
Pier-Jean Malandrino in Scub-Lab

Data Lake vs. Data Warehouse

Understanding Their Core Differences, Benefits, and Trade-offs

4 min read · Mar 11, 2024

20 1



Pavlo Kolodka in Level Up Coding

The 20 Essential Principles of Software Development: LoD, SoC, SOLID, and Beyond.

Core software development principles that every developer must know.

19 min read · Apr 14, 2024

1.4K 16

Somnath Singh in Level Up Coding

The Era of High-Paying Tech Jobs is Over

The Death of Tech Jobs.

✨ · 14 min read · Apr 1, 2024

10.7K 267

Pier-Jean Malandrino in Scub-Lab

Architecture : The cheat sheet

This paper presents a concise summary of various software architecture patterns, methods and models.

5 min read · Dec 25, 2023

1.92K 6

See all from Pier-Jean Malandrino

See all from Level Up Coding

Recommended from Medium

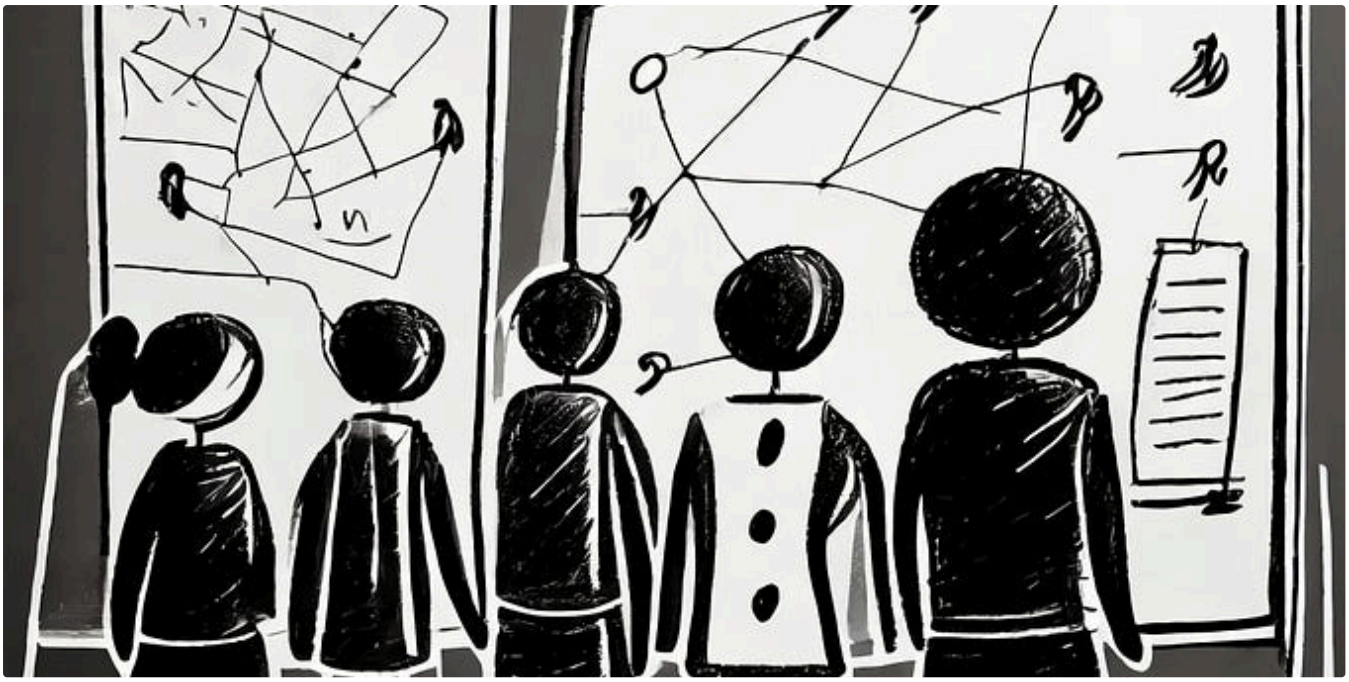
huizhou92 in Level Up Coding

If Google No Longer supports Golang

Analyzing the Potential Impact of Google's Discontinuation of Support for Go Language and Possible Future Scenarios

· 3 min read · May 7, 2024

512 17



Daniel Moldovan

How to be a staff engineer. Or team lead. Or principal engineer. Or manager. Or whatever ...

Many wonder what does it take to become a senior software engineer. Or staff engineer. Or principal software engineer. Or manager. Or ...

15 min read · Jan 11, 2024

200 1

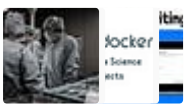
Lists

General Coding Knowledge

20 stories · 1210 saves

Stories to Help You Grow as a Software Developer

19 stories · 1053 saves



Coding & Development

11 stories · 610 saves

ChatGPT prompts

47 stories · 1547 saves

Jake Page

The guide to kubectl I never had.

What kind of engineer are you? 🤔 Can somebody guess by just looking at you? More than likely not.

17 min read · May 6, 2024

447 5

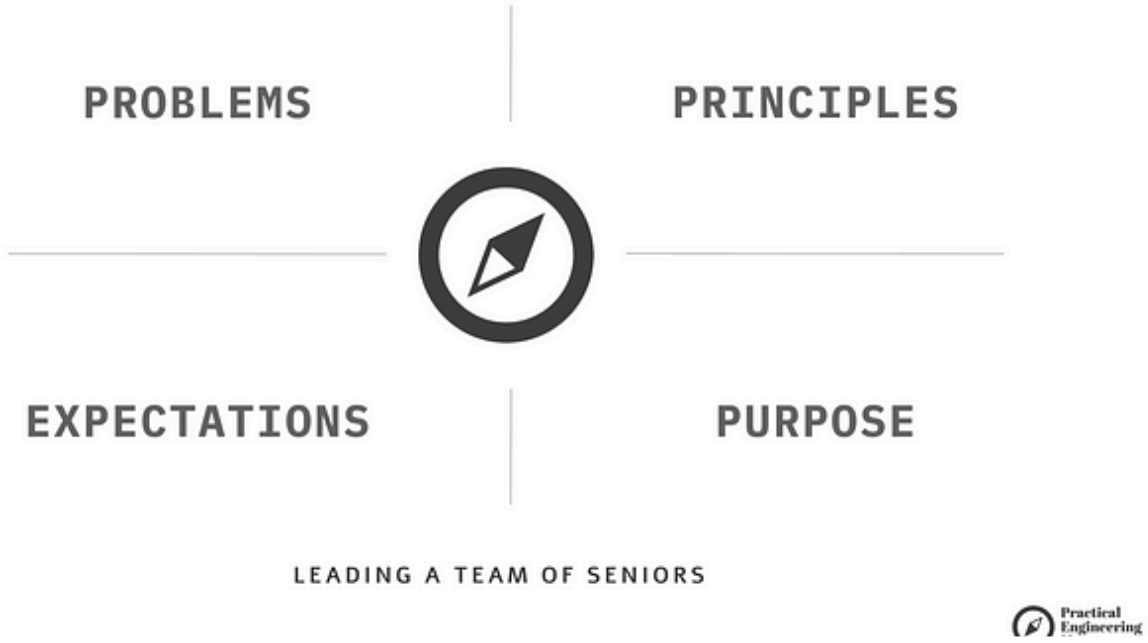
Mario Bittencourt in SSENSE-TECH

Exploring Advanced Error Handling Patterns with Event-Driven Architecture—Part I

An event-driven architecture (EDA) brings changes to the way we approach error handling. When using the more commonly adopted synchronous...

7 min read · Apr 5, 2024

340 1



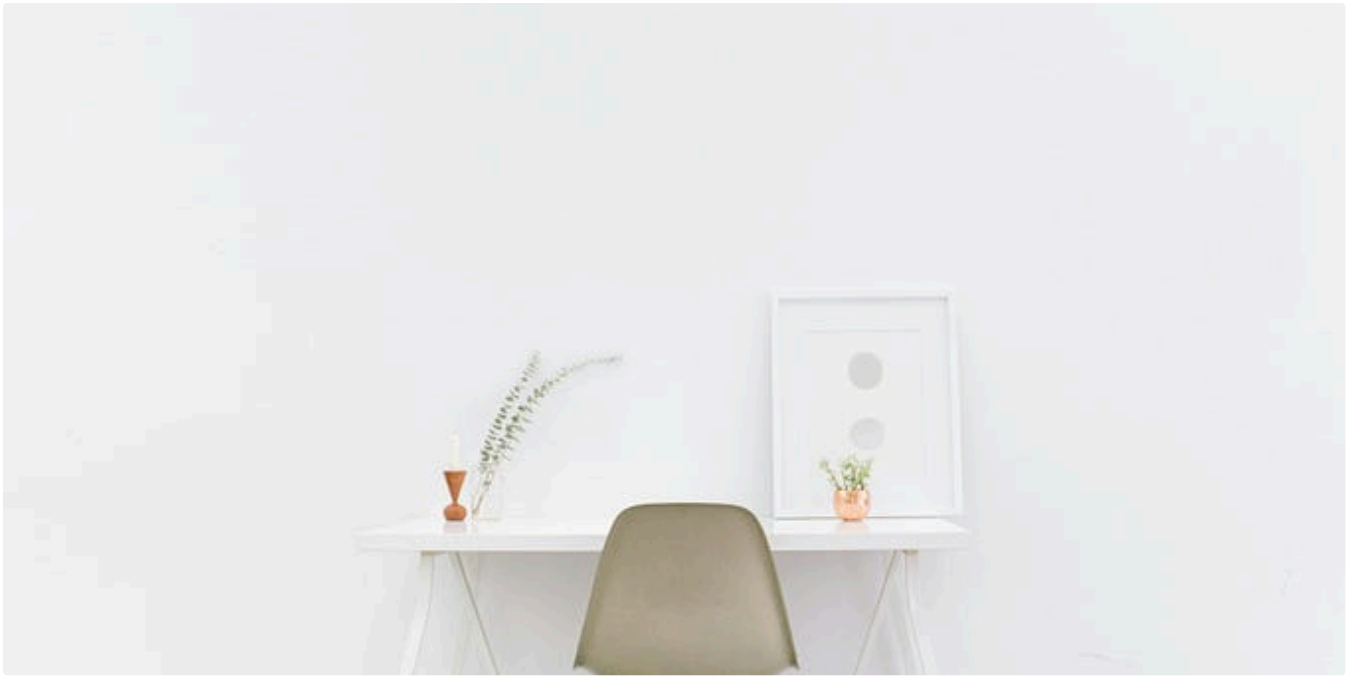
Mirek Stanek in Practical Engineering Management

How to lead a team of senior engineers

Managing people who are more skillful than you

8 min read · Jan 17, 2024

1.6K 14



Aashish Peepra

How to design clean API interfaces

This is going to be a tough read. This contains too much code to read and no pictures. It's not for people with faint hearts. So read if...

7 min read · Mar 3, 2024

525 7

See more recommendations