

hexagonal architecture vs microservices

hexagonal architecture vs microservices represents a critical comparison in the domain of modern software architecture. Both architectural patterns aim to improve software modularity, scalability, and maintainability, yet they approach these goals from different perspectives. Hexagonal architecture, also known as ports and adapters, focuses on designing systems that are independent of external frameworks, databases, and user interfaces. Microservices, on the other hand, emphasize decomposing applications into loosely coupled, independently deployable services. Understanding the distinctions, benefits, and use cases of hexagonal architecture versus microservices is essential for architects and developers aiming to build robust, flexible systems. This article explores these two paradigms in detail, highlighting their core principles, advantages, challenges, and how they can complement each other in complex software ecosystems.

- Understanding Hexagonal Architecture
- Overview of Microservices Architecture
- Comparative Analysis: Hexagonal Architecture vs Microservices
- Benefits and Challenges of Hexagonal Architecture
- Benefits and Challenges of Microservices
- When to Use Hexagonal Architecture or Microservices
- Integrating Hexagonal Architecture Within Microservices

Understanding Hexagonal Architecture

Hexagonal architecture is a design pattern that structures an application to isolate its core logic from external influences. Also known as ports and adapters architecture, it promotes a clear separation between the domain logic and external systems such as databases, user interfaces, and third-party services. The key concept is to make the application's core independent, enabling it to interact with the outside world through well-defined ports and adapters.

Core Principles of Hexagonal Architecture

The architecture revolves around the idea that the domain logic should be at the center, surrounded by ports that define input and output operations. Adapters then implement these ports to connect with external systems. This separation ensures that the core business rules remain unaffected by changes in external technologies or frameworks.

Structure and Components

Typically, the hexagonal architecture consists of three layers:

- **Domain Layer:** Contains business logic and rules.
- **Application Layer:** Defines ports that expose domain functionality.
- **Infrastructure Layer:** Implements adapters for interacting with databases, UI, and external services.

Overview of Microservices Architecture

Microservices architecture is an approach to building software systems as a collection of small, autonomous services that communicate over network protocols. Each microservice encapsulates a specific business capability and can be developed, deployed, and scaled independently. This approach contrasts with traditional monolithic architectures by breaking down applications into manageable components.

Key Characteristics of Microservices

Microservices typically possess the following traits:

- Decentralized data management with each service owning its database.
- Independent deployment pipelines for each service.
- Communication through lightweight protocols such as HTTP/REST or messaging queues.
- Focus on business capabilities aligned with bounded contexts.

Typical Microservices Architecture Components

A microservices-based system generally includes multiple independent services, API gateways to route requests, service discovery mechanisms, and centralized logging and monitoring for operational visibility.

Comparative Analysis: Hexagonal Architecture vs Microservices

While hexagonal architecture and microservices both aim to improve modularity and scalability, they operate at different levels of software design. Hexagonal architecture focuses on the internal structure of a single application or service, enhancing testability and adaptability. Microservices architecture addresses system-level decomposition by splitting functionality into discrete services.

Scope and Focus

Hexagonal architecture is primarily concerned with isolating business logic within a single application, whereas microservices deal with the distribution of functionality across multiple services. This distinction means hexagonal architecture can be implemented within a microservice or a monolith.

Modularity and Independence

Both architectures promote modularity. Hexagonal architecture achieves this by decoupling the core logic from external dependencies. Microservices foster independence by enabling services to be developed, deployed, and scaled separately.

Communication and Integration

In hexagonal architecture, communication occurs internally via ports and adapters, often within the same process. In microservices, communication happens over networks, using protocols like REST or messaging, which introduces complexities such as latency and fault tolerance.

Benefits and Challenges of Hexagonal Architecture

Implementing hexagonal architecture offers several advantages, especially for managing complexity within an application.

Benefits

- **Improved Testability:** Isolation of business logic simplifies unit testing.
- **Flexibility:** Easy to switch external systems without affecting core logic.
- **Maintainability:** Clear separation of concerns aids in code organization and maintenance.
- **Technology Agnostic:** Core domain remains independent of specific technologies or frameworks.

Challenges

- **Initial Learning Curve:** Developers must understand ports and adapters concepts.
- **Complexity in Small Projects:** May introduce unnecessary abstraction for simple applications.
- **Potential Overhead:** Additional layers can increase development effort.

Benefits and Challenges of Microservices

Microservices architecture provides significant benefits but also introduces unique challenges that need consideration.

Benefits

- **Scalability:** Services can be scaled independently based on demand.
- **Resilience:** Failure in one service does not necessarily affect others.
- **Technology Diversity:** Teams can choose different technologies per service.
- **Faster Deployment:** Independent deployment cycles accelerate delivery.

Challenges

- **Complexity:** Distributed systems require handling network communication, latency, and fault tolerance.
- **Data Consistency:** Managing transactions and consistency across services can be difficult.
- **Operational Overhead:** Requires infrastructure for service discovery, monitoring, and logging.
- **Inter-Service Communication:** Designing robust APIs and handling versioning is critical.

When to Use Hexagonal Architecture or Microservices

Choosing between hexagonal architecture and microservices depends on project requirements, team expertise, and system complexity. These architectures are not mutually exclusive and can be combined effectively.

Use Cases for Hexagonal Architecture

- Applications needing clear separation between business logic and infrastructure.
- Projects where testability and maintainability are priorities.
- Systems aiming for technology-agnostic core logic to accommodate future changes.
- Monolithic applications that require improved structure and flexibility.

Use Cases for Microservices

- Large-scale systems requiring independent scaling of components.
- Organizations with multiple teams working on different business capabilities.
- Systems demanding high availability and fault tolerance.
- Projects that benefit from technology diversity and continuous deployment.

Integrating Hexagonal Architecture Within Microservices

Hexagonal architecture can be effectively applied inside individual microservices to enhance their internal structure. By using hexagonal principles, each microservice can maintain a clean separation between domain logic and external dependencies, increasing testability and adaptability.

Advantages of Combining Both

- **Enhanced Modularity:** Clear boundaries within each microservice.
- **Improved Maintainability:** Easier to update and replace adapters without impacting core logic.
- **Better Test Coverage:** Domain logic can be tested independently from infrastructure concerns.
- **Consistency:** Uniform architectural style across services simplifies understanding and onboarding.

Incorporating hexagonal architecture within microservices is a best practice for building scalable, maintainable, and resilient distributed systems. This layered approach helps manage complexity both within and across services, aligning with modern software development goals.

Frequently Asked Questions

What is hexagonal architecture?

Hexagonal architecture, also known as Ports and Adapters, is a software design pattern that emphasizes a clear separation between the core business logic and external systems, allowing easy

adaptability and maintainability.

What are microservices?

Microservices is an architectural style that structures an application as a collection of small, autonomous services, each responsible for a specific business capability, communicating over network protocols.

How does hexagonal architecture differ from microservices?

Hexagonal architecture is a design pattern focused on structuring individual applications with clear boundaries between core logic and external interfaces, whereas microservices is an architectural approach that decomposes an entire system into multiple independent services.

Can hexagonal architecture be used within microservices?

Yes, hexagonal architecture can be applied within each microservice to ensure that the service's core domain logic is isolated from infrastructure concerns, making the microservice more maintainable and testable.

Which architecture is better for scalability: hexagonal architecture or microservices?

Microservices architecture generally offers better scalability at the system level since services can be scaled independently, while hexagonal architecture improves code maintainability and can be used inside scalable microservices.

Does hexagonal architecture replace microservices?

No, hexagonal architecture does not replace microservices; instead, it can complement microservices by providing a robust way to organize the internal structure of each microservice.

What are the main benefits of using hexagonal architecture in microservices?

Using hexagonal architecture in microservices enhances testability, maintainability, and flexibility by decoupling business logic from external dependencies within each service.

How do microservices handle communication compared to hexagonal architecture?

Microservices communicate over network protocols such as HTTP or messaging, focusing on inter-service communication, while hexagonal architecture deals with the internal interaction between the core logic and external interfaces within a single application.

Is it possible to implement microservices without hexagonal architecture?

Yes, microservices can be implemented without hexagonal architecture, but adopting hexagonal architecture can improve service design by promoting separation of concerns and easier testing.

What challenges might arise when combining hexagonal architecture with microservices?

Combining hexagonal architecture with microservices can increase complexity due to multiple layers of abstraction and requires disciplined design to ensure that each service remains cohesive and that inter-service communication does not become a bottleneck.

Additional Resources

1. Hexagonal Architecture: Designing Robust and Maintainable Software

This book offers a comprehensive introduction to hexagonal architecture, also known as the ports and

adapters pattern. It explains how this architectural style promotes loose coupling and testability by isolating the core logic from external systems. Readers will learn practical techniques to implement hexagonal architecture in various programming languages and improve software maintainability.

2. Microservices vs. Hexagonal Architecture: Choosing the Right Approach

Focused on comparing microservices and hexagonal architecture, this book helps software architects and developers understand the strengths and trade-offs of each approach. It covers scenarios where one might be more beneficial than the other and discusses how these patterns can coexist in large-scale systems. The book includes case studies and decision frameworks to guide architectural choices.

3. Building Scalable Systems with Microservices and Hexagonal Architecture

This title explores how microservices and hexagonal architecture can be combined to create scalable, resilient systems. It delves into design principles, communication patterns, and deployment strategies that leverage both approaches. Readers will gain insights into managing complexity and improving system evolution through modular design.

4. Practical Hexagonal Architecture for Microservices Developers

Targeted at developers working with microservices, this book provides practical guidance on applying hexagonal architecture principles within microservice-based applications. It covers techniques for defining clear boundaries, handling integrations, and ensuring testability in distributed systems. The content is enriched with code examples and real-world scenarios.

5. Domain-Driven Design and Hexagonal Architecture: A Perfect Match

This book bridges domain-driven design (DDD) concepts with hexagonal architecture to help developers build well-structured, domain-centric applications. It explains how hexagonal architecture supports DDD's emphasis on the domain model by decoupling it from infrastructure concerns. Readers will find strategies for aligning architecture with business requirements effectively.

6. Microservices Patterns and Hexagonal Architecture: Implementing Modern Software Solutions

Covering a broad spectrum of microservices design patterns, this book integrates hexagonal

architecture as a foundational concept for building modular services. It discusses service decomposition, event-driven communication, and testing strategies alongside hexagonal principles. The author provides practical advice for creating resilient and maintainable microservices.

7. Refactoring to Hexagonal Architecture in a Microservices World

This guide focuses on refactoring existing microservice applications to adopt hexagonal architecture. It addresses common challenges such as breaking dependencies on frameworks and external services, improving testability, and enhancing modularity. The book includes step-by-step refactoring techniques and best practices to evolve legacy codebases.

8. Hexagonal Architecture in Action: Case Studies from Microservices Projects

Through detailed case studies, this book demonstrates the application of hexagonal architecture in real-world microservices projects. It highlights successes and pitfalls, providing valuable lessons for architects and developers. The narrative helps readers understand how to adapt hexagonal principles to diverse business domains and technical contexts.

9. Comparative Architectures: Hexagonal Architecture, Microservices, and Beyond

This book offers a wider perspective on software architecture by comparing hexagonal architecture and microservices with other modern architectural styles. It evaluates each pattern's impact on scalability, maintainability, and team organization. Readers will benefit from frameworks and criteria to select the most suitable architecture for their projects.

Hexagonal Architecture Vs Microservices

Find other PDF articles:

<http://www.speargroupplc.com/gacor1-15/pdf?ID=Sxd86-4688&title=harry-wong-first-days-of-school.pdf>

hexagonal architecture vs microservices: *Hands-On Software Architecture with Java*
Giuseppe Bonocore, Arunee Singhchawla, 2022-03-16 Build robust and scalable Java applications by learning how to implement every aspect of software architecture Key Features Understand the fundamentals of software architecture and build production-grade applications in Java Make smart

architectural decisions with comprehensive coverage of various architectural approaches from SOA to microservices Gain an in-depth understanding of deployment considerations with cloud and CI/CD pipelines Book Description Well-written software architecture is the core of an efficient and scalable enterprise application. Java, the most widespread technology in current enterprises, provides complete toolkits to support the implementation of a well-designed architecture. This book starts with the fundamentals of architecture and takes you through the basic components of application architecture. You'll cover the different types of software architectural patterns and application integration patterns and learn about their most widespread implementation in Java. You'll then explore cloud-native architectures and best practices for enhancing existing applications to better suit a cloud-enabled world. Later, the book highlights some cross-cutting concerns and the importance of monitoring and tracing for planning the evolution of the software, foreseeing predictable maintenance, and troubleshooting. The book concludes with an analysis of the current status of software architectures in Java programming and offers insights into transforming your architecture to reduce technical debt. By the end of this software architecture book, you'll have acquired some of the most valuable and in-demand software architect skills to progress in your career. What you will learn Understand the importance of requirements engineering, including functional versus non-functional requirements Explore design techniques such as domain-driven design, test-driven development (TDD), and behavior-driven development Discover the mantras of selecting the right architectural patterns for modern applications Explore different integration patterns Enhance existing applications with essential cloud-native patterns and recommended practices Address cross-cutting considerations in enterprise applications regardless of architectural choices and application type Who this book is for This book is for Java software engineers who want to become software architects and learn everything a modern software architect needs to know. The book is also for software architects, technical leaders, vice presidents of software engineering, and CTOs looking to extend their knowledge and stay up to date with the latest developments in the field of software architecture.

hexagonal architecture vs microservices: Microservice APIs Jose Haro Peralta, 2023-01-10 'Microservice APIs in Python' shares successful strategies and techniques for designing Microservices systems, with a particular emphasis on creating easy-to-consume APIs. The practical guide focuses on implementation over philosophising and has just enough theory to get you started. You'll quickly go hands on designing the architecture for a microservices platform, produce standard specifications for REST and GraphQL APIs, and bake in authentication features to keep your APIs secure.

hexagonal architecture vs microservices: Java Microservices and Containers in the Cloud Binildas A. Christudas, 2024-09-28 Spring Boot helps developers create applications that simply run. When minimal configuration is required to start up an application, even novice Java developers are ready to start. But this simplicity shouldn't constrain developers in addressing more complex enterprise requirements where microservice architecture is concerned. With the need to rapidly deploy, patch, or scale applications, containers provide solutions which can accelerate development, testing as well as production cycles. The cloud helps companies to scale and adapt at speed, accelerate innovation and drive business agility, without heavy upfront IT investment. What if we can equip even a novice developer with all that is required to help enterprises achieve all of this, this book does this and more. Java Microservices and Containers in the Cloud offers a comprehensive guide to both architecture and programming aspects to Java microservices development, providing a fully hands-on experience. We not only describe various architecture patterns but also provide practical implementations of each pattern through code examples. Despite the focus on architecture, this book is designed to be accessible to novice developers with only basic programming skills, such as writing a Hello World program and using Maven to compile and run Java code. It ensures that even such readers can easily comprehend, deploy, and execute the code samples provided in the book. Regardless of your current knowledge or lack thereof in Docker, Kubernetes, and Cloud technologies, this book will empower you to develop programming skills in these areas. There is no

restriction on beginners attempting to understand serious and non-trivial architecture constraints. While mastering concurrency and scalability techniques often requires years of experience, this book promises to empower you to write microservices, as well as how to containerize and deploy them in the cloud. If you are a non-programming manager who is not afraid to read code snippets, this book will empower you to navigate the challenges posed by seasoned architects. It will equip you with the necessary understanding of specialized jargon, enabling you to engage in more meaningful discussions and break through barriers when collaborating with programmers, architects and engineers across the table. The code examples provided in the book are intentionally designed to be simple and accessible to all, regardless of your programming background. Even if you are a C# or Python programmer and not familiar with Java, you will find the code examples easy to follow and understand. You will Acquire proficiency in both RPC-style and Messaging-style inter-microservice communication Construct microservices utilizing a combination of SQL (PostgreSQL) and NoSQL (MongoDB) databases Leverage Liquibase, a database schema version control tool, and administer UI in conjunction with PostgreSQL Leverage both GraphQL and conventional REST approaches side by side Gain practical experience in implementing Hexagonal and Onion Architectures through hands-on exercises Integrate asynchronous processing into your Java applications using powerful APIs such as DeferredResult and CompletableFuture Who it's for: Developers, programmers and Architects who want to level up their Java Micoservices and Architecture knowledge as well as managers who want to brush up on their technical knowledge around the topic.

hexagonal architecture vs microservices: Microservices Patterns Chris Richardson, 2018-10-27 A comprehensive overview of the challenges teams face when moving to microservices, with industry-tested solutions to these problems. - Tim Moore, Lightbend 44 reusable patterns to develop and deploy reliable production-quality microservices-based applications, with worked examples in Java Key Features 44 design patterns for building and deploying microservices applications Drawing on decades of unique experience from author and microservice architecture pioneer Chris Richardson A pragmatic approach to the benefits and the drawbacks of microservices architecture Solve service decomposition, transaction management, and inter-service communication Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About The Book Microservices Patterns teaches you 44 reusable patterns to reliably develop and deploy production-quality microservices-based applications. This invaluable set of design patterns builds on decades of distributed system experience, adding new patterns for composing services into systems that scale and perform under real-world conditions. More than just a patterns catalog, this practical guide with worked examples offers industry-tested advice to help you design, implement, test, and deploy your microservices-based application. What You Will Learn How (and why!) to use microservices architecture Service decomposition strategies Transaction management and querying patterns Effective testing strategies Deployment patterns This Book Is Written For Written for enterprise developers familiar with standard enterprise application architecture. Examples are in Java. About The Author Chris Richardson is a Java Champion, a JavaOne rock star, author of Manning's POJOs in Action, and creator of the original CloudFoundry.com. Table of Contents Escaping monolithic hell Decomposition strategies Interprocess communication in a microservice architecture Managing transactions with sagas Designing business logic in a microservice architecture Developing business logic with event sourcing Implementing queries in a microservice architecture External API patterns Testing microservices: part 1 Testing microservices: part 2 Developing production-ready services Deploying microservices Refactoring to microservices

hexagonal architecture vs microservices: Microservices Eberhard Wolff, 2016-10-03 The Most Complete, Practical, and Actionable Guide to Microservices Going beyond mere theory and marketing hype, Eberhard Wolff presents all the knowledge you need to capture the full benefits of this emerging paradigm. He illuminates microservice concepts, architectures, and scenarios from a technology-neutral standpoint, and demonstrates how to implement them with today's leading

technologies such as Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud. The author fully explains the benefits and tradeoffs associated with microservices, and guides you through the entire project lifecycle: development, testing, deployment, operations, and more. You'll find best practices for architecting microservice-based systems, individual microservices, and nanoservices, each illuminated with pragmatic examples. The author supplements opinions based on his experience with concise essays from other experts, enriching your understanding and illuminating areas where experts disagree. Readers are challenged to experiment on their own the concepts explained in the book to gain hands-on experience. Discover what microservices are, and how they differ from other forms of modularization Modernize legacy applications and efficiently build new systems Drive more value from continuous delivery with microservices Learn how microservices differ from SOA Optimize the microservices project lifecycle Plan, visualize, manage, and evolve architecture Integrate and communicate among microservices Apply advanced architectural techniques, including CQRS and Event Sourcing Maximize resilience and stability Operate and monitor microservices in production Build a full implementation with Docker, Java, Spring Boot, the Netflix stack, and Spring Cloud Explore nanoservices with Amazon Lambda, OSGi, Java EE, Vert.x, Erlang, and Seneca Understand microservices' impact on teams, technical leaders, product owners, and stakeholders Managers will discover better ways to support microservices, and learn how adopting the method affects the entire organization. Developers will master the technical skills and concepts they need to be effective. Architects will gain a deep understanding of key issues in creating or migrating toward microservices, and exactly what it will take to transform their plans into reality.

hexagonal architecture vs microservices: Fundamentals of Software Architecture Craig Risi, 2025-05-30 DESCRIPTION With the rising complexity of modern software systems, strong, scalable software architecture has become the backbone of any successful application. This book gives you the essential knowledge to grasp the core ideas and methods of effective software design, helping you build strong, flexible systems right from the start. The book systematically navigates the critical aspects of software architecture, commencing with a clear definition of its significance and the pivotal role of the software architect. It delves into fundamental architectural properties like performance, security, and maintainability, underscoring the importance of modularity in crafting well-structured systems. You will explore various established architectural styles, including microservices and layered architecture, alongside key design patterns such as MVC and repository, gaining insights into their practical application. The book further elucidates the function of software components, the art of architecting for optimal performance and security, and essential design principles for building robust solutions. Finally, it examines the impact of modern development practices (Agile, DevOps), positions architecture within the broader engineering context, emphasizes the importance of testing at the architectural level, and offers a glimpse into current and future trends shaping the field. By the end of this book, you will have a solid understanding of the core concepts, helping you to contribute effectively to software design discussions, make informed architectural decisions, and build a strong foundation for creating high-quality, future-proof software systems. WHAT YOU WILL LEARN ● Define core architecture, architect roles, and fundamental design attributes. ● Apply modularity principles for resilient and adaptable software design. ● Design cohesive components, manage coupling, and optimize system decomposition. ● Cultivate essential soft skills for effective leadership and stakeholder management. ● Define technical requirements and understand modern development practices. WHO THIS BOOK IS FOR This book is for software developers, technical leads, and anyone involved in software creation, seeking a foundational understanding of software architecture principles and practices to enhance their design skills and project outcomes. TABLE OF CONTENTS Prologue 1. Defining Software Architecture 2. The Role of a Software Architect 3. Architectural Properties 4. The Importance of Modularity 5. Architectural Styles 6. Architectural Patterns 7. Component Architecture 8. Architecting for Performance 9. Architecting for Security 10. Design and Presentation 11. Evolutionary Architecture 12. Soft Skills for Software Architects 13. Writing Technical Requirements 14. Development Practices 15. Architecture as Engineering 16. Testing in Software Architecture 17. Current and

hexagonal architecture vs microservices: gRPC Microservices in Go Hüseyin Babal, 2024-01-09 Build super fast and super secure microservices with the gRPC high-performance messaging protocol and powerful Go language. In gRPC Microservices in Go you'll learn: Designing and implementing resilient microservice architecture Testing microservices Deploying microservices to the cloud with modern orchestration tools Monitoring and overseeing microservices The powerful gRPC Remote Procedure Call framework delivers superior speed and security over protocols like REST. When paired with Golang's low-level efficiency and flexibility, gRPC and Go become a killer combination for latency-sensitive microservices applications. gRPC Microservices in Go shows you how to utilize these powerful tools to build production-grade microservices. You'll learn to develop microservice inter-service communication patterns that are powered by gRPC, design backward compatible APIs, and apply hexagonal architecture to microservices. About the technology Go is perfect for writing fast, reliable microservices code, but that's only half the story. You also need a communications framework like gRPC to connect your services and handle load balancing, tracing, health checking, and authentication. Together, Go and gRPC accelerate the development process and eliminate many of the challenges you face when building and deploying microservices. About the book gRPC Microservices in Go teaches you how to build production-ready microservices using Go and gRPC. In it, you'll learn to create efficient APIs in Go, use gRPC for network communication, and deploy on cloud and Kubernetes. Helpful examples, including a complete eCommerce web app, make it easy to grasp each concept. You'll also get an inside look at testing, deployment, and efficient DevOps practices for microservices. What's inside Designing and implementing resilient microservice architecture Testing microservices Cloud deploying microservices with orchestration tools Monitoring and overseeing microservices About the reader For software developers who know the basics of Go. About the author Hüseyin Babal has been using Go in production since 2017 to build and maintain SaaS platforms. Table of Contents PART 1 - GRPC AND MICROSERVICES ARCHITECTURE 1 Introduction to Go gRPC microservices 2 gRPC meets microservices PART 2 - DEVELOPING, TESTING, AND DEPLOYING A GRPC MICROSERVICE APPLICATION 3 Getting up and running with gRPC and Golang 4 Microservice project setup 5 Interservice communication 6 Resilient communication 7 Testing microservices 8 Deployment PART 3 - GRPC AND MICROSERVICES ARCHITECTURE 9 Observability

hexagonal architecture vs microservices: Mastering API Architecture James Gough, Daniel Bryant, Matthew Auburn, 2021-03-19 Most organizations with a web presence build and operate APIs; the doorway for customers to interact with the company's services. Designing, building, and managing these critical programs affect everyone in the organization, from engineers and product owners to C-suite executives. But the real challenge for developers and solution architects is creating an API platform from the ground up. With this practical book, you'll learn strategies for building and testing REST APIs that use API gateways to combine offerings at the microservice level. Authors James Gough, Daniel Bryant, and Matthew Auburn demonstrate how simple additions to this infrastructure can help engineers and organizations migrate to the cloud; and open the opportunity to connect internal services using technologies like a service mesh. Learn API fundamentals and architectural patterns for building an API platform Use practical examples to understand how to design, build, and test API-based systems Deploy, operate, and configure key components of an API platform Use API gateways and service meshes appropriately, based on case studies Understand core security and common vulnerabilities in API architecture Secure data and APIs using threat modeling and technologies like OAuth2 and TLS Learn how to evolve existing systems toward API- and cloud-based architectures

hexagonal architecture vs microservices: Docker: Zero To Hero Rob Botwright, 2024 □ DOCKER: ZERO TO HERO BOOK BUNDLE □ Ready to level up your Docker skills and become a containerization pro? Look no further! Introducing the Docker: Zero to Hero book bundle, your ultimate guide to building, testing, and deploying applications fast. With four comprehensive books covering everything from Docker basics to expert-level techniques, this bundle has everything you

need to master Docker and revolutionize your development workflow. □ **BOOK 1: DOCKER DEMYSTIFIED** □ New to Docker? No problem! Dive into the world of containerization with Docker Demystified, a beginner's guide that breaks down complex concepts into easy-to-understand lessons. Learn how Docker works, create and manage containers, and discover the power of containerization for modern software development. □ **BOOK 2: MASTERING DOCKER** □ Ready to take your Docker skills to the next level? Mastering Docker is your roadmap to advanced techniques and best practices. Optimize Docker images, implement networking and storage solutions, and orchestrate multi-container applications with Docker Compose. Whether you're deploying in the cloud or on-premises, this book has you covered. □ **BOOK 3: DOCKER DEPLOYMENT STRATEGIES** □ Scaling and orchestrating containers at scale is a breeze with Docker Deployment Strategies. Explore different deployment strategies, from setting up Docker Swarm clusters to rolling updates and service scaling. Plus, learn advanced networking and security considerations for deploying Docker in production environments. □ **BOOK 4: EXPERT DOCKER** □ Ready to become a Docker expert? Expert Docker is your guide to building complex microservices architectures with confidence. Architect and deploy sophisticated, distributed systems using Docker, and design scalable, resilient, and maintainable microservices architectures that stand the test of time. With over 3000 characters of expert guidance and practical advice, the Docker: Zero to Hero book bundle is your ticket to mastering Docker and transforming your development workflow. Don't miss out on this opportunity to become a Docker hero – grab your bundle today and start building, testing, and deploying applications faster than ever before! □□

hexagonal architecture vs microservices: Fundamentals of Software Architecture Mark Richards, Neal Ford, 2025-03-12 Salary surveys worldwide regularly place software architect in the top 10 best jobs, yet no real guide exists to help developers become architects. Until now. This updated edition provides a comprehensive overview of software architecture's many aspects, with five new chapters covering the latest insights from the field. Aspiring and existing architects alike will examine architectural characteristics, architectural patterns, component determination, diagramming architecture, governance, data, generative AI, team topologies, and many other topics. Mark Richards and Neal Ford—hands-on practitioners who have taught software architecture classes professionally for years—focus on architecture principles that apply across all technology stacks. You'll explore software architecture in a modern light, taking into account all the innovations of the past decade. This book examines: Architecture styles and patterns: Microservices, modular monoliths, microkernels, layered architectures, and many more Components: Identification, coupling, cohesion, partitioning, and granularity Soft skills: Effective team management, collaboration, business engagement models, negotiation, presentations, and more Modernity: Engineering practices and operational approaches that have changed radically in the past few years, including cloud considerations and generative AI Architecture as an engineering discipline: Repeatable results, metrics, and concrete valuations that add rigor to software architecture

hexagonal architecture vs microservices: Kubernetes Patterns Bilgin Ibryam, Roland Huss, 2022-09-01 The way developers design, build, and run software has changed significantly with the evolution of microservices and containers. These modern architectures offer new distributed primitives that require a different set of practices than many developers, tech leads, and architects are accustomed to. With this focused guide, Bilgin Ibryam and Roland Huss provide common reusable patterns and principles for designing and implementing cloud native applications on Kubernetes. Each pattern includes a description of the problem and a Kubernetes-specific solution. All patterns are backed by and demonstrated with concrete code examples. This updated edition is ideal for developers and architects familiar with basic Kubernetes concepts who want to learn how to solve common cloud native challenges with proven design patterns. You'll explore: Foundational patterns covering core principles and practices for building and running container-based cloud native applications Behavioral patterns that delve into finer-grained concepts for managing various types of container and platform interactions Structural patterns for organizing containers within a Pod for addressing specific use cases Configuration patterns that provide insight into how

application configurations can be handled in Kubernetes Security patterns for hardening the access to cloud native applications running on Kubernetes Advanced patterns covering more complex topics such as operators and autoscaling

hexagonal architecture vs microservices: Software Architecture: The Hard Parts Neal Ford, Mark Richards, Pramod Sadalage, Zhamak Dehghani, 2021-09-23 There are no easy decisions in software architecture. Instead, there are many hard parts--difficult problems or issues with no best practices--that force you to choose among various compromises. With this book, you'll learn how to think critically about the trade-offs involved with distributed architectures. Architecture veterans and practicing consultants Neal Ford, Mark Richards, Pramod Sadalage, and Zhamak Dehghani discuss strategies for choosing an appropriate architecture. By interweaving a story about a fictional group of technology professionals--the Sysops Squad--they examine everything from how to determine service granularity, manage workflows and orchestration, manage and decouple contracts, and manage distributed transactions to how to optimize operational characteristics, such as scalability, elasticity, and performance. By focusing on commonly asked questions, this book provides techniques to help you discover and weigh the trade-offs as you confront the issues you face as an architect. Analyze trade-offs and effectively document your decisions Make better decisions regarding service granularity Understand the complexities of breaking apart monolithic applications Manage and decouple contracts between services Handle data in a highly distributed architecture Learn patterns to manage workflow and transactions when breaking apart applications

hexagonal architecture vs microservices: Java EE 8 Design Patterns and Best Practices Rhuan Rocha, João Purificação, 2018-08-10 Get the deep insights you need to master efficient architectural design considerations and solve common design problems in your enterprise applications. Key Features The benefits and applicability of using different design patterns in JAVA EE Learn best practices to solve common design and architectural challenges Choose the right patterns to improve the efficiency of your programs Book Description Patterns are essential design tools for Java developers. Java EE Design Patterns and Best Practices helps developers attain better code quality and progress to higher levels of architectural creativity by examining the purpose of each available pattern and demonstrating its implementation with various code examples. This book will take you through a number of patterns and their Java EE-specific implementations. In the beginning, you will learn the foundation for, and importance of, design patterns in Java EE, and then will move on to implement various patterns on the presentation tier, business tier, and integration tier. Further, you will explore the patterns involved in Aspect-Oriented Programming (AOP) and take a closer look at reactive patterns. Moving on, you will be introduced to modern architectural patterns involved in composing microservices and cloud-native applications. You will get acquainted with security patterns and operational patterns involved in scaling and monitoring, along with some patterns involved in deployment. By the end of the book, you will be able to efficiently address common problems faced when developing applications and will be comfortable working on scalable and maintainable projects of any size. What you will learn Implement presentation layers, such as the front controller pattern Understand the business tier and implement the business delegate pattern Master the implementation of AOP Get involved with asynchronous EJB methods and REST services Involve key patterns in the adoption of microservices architecture Manage performance and scalability for enterprise-level applications Who this book is for Java developers who are comfortable with programming in Java and now want to learn how to implement design patterns to create robust, reusable and easily maintainable apps.

hexagonal architecture vs microservices: Designing Hexagonal Architecture with Java Davi Vieira, 2023-09-29 Learn to build robust, resilient, and highly maintainable cloud-native Java applications with hexagonal architecture and Quarkus Key Features Use hexagonal architecture to increase maintainability and reduce technical debt Learn how to build systems that are easy to change and understand Leverage Quarkus to create modern cloud-native applications Purchase of the print or Kindle book includes a free PDF eBook Book Description We live in a fast-evolving world with new technologies emerging every day, where enterprises are constantly changing in an

unending quest to be more profitable. So, the question arises — how to develop software capable of handling a high level of unpredictability. With this question in mind, this book explores how the hexagonal architecture can help build robust, change-tolerable, maintainable, and cloud-native applications that can meet the needs of enterprises seeking to increase their profits while dealing with uncertainties. This book starts by uncovering the secrets of the hexagonal architecture's building blocks, such as entities, use cases, ports, and adapters. You'll learn how to assemble business code in the domain hexagon, create features with ports and use cases in the application hexagon, and make your software compatible with different technologies by employing adapters in the framework hexagon. In this new edition, you'll learn about the differences between a hexagonal and layered architecture and how to apply SOLID principles while developing a hexagonal system based on a real-world scenario. Finally, you'll get to grips with using Quarkus to turn your hexagonal application into a cloud-native system. By the end of this book, you'll be able to develop robust, flexible, and maintainable systems that will stand the test of time. What you will learn

- Apply SOLID principles to the hexagonal architecture
- Assemble business rules algorithms using the specified design pattern
- Combine domain-driven design techniques with hexagonal principles to create powerful domain models
- Employ adapters to enable system compatibility with various protocols such as REST, gRPC, and WebSocket
- Create a module and package structure based on hexagonal principles
- Use Java modules to enforce dependency inversion and ensure software component isolation
- Implement Quarkus DI to manage the life cycle of input and output ports

Who this book is for
This book is for software architects and Java developers looking to improve code maintainability and enhance productivity with an architecture that allows changes in technology without compromising business logic. Intermediate knowledge of the Java programming language and familiarity with Jakarta EE will help you to get the most out of this book.

hexagonal architecture vs microservices: Domain-driven Design with Java Otavio Santana, 2025-09-22

DESCRIPTION Domain-driven Design (DDD) continues to shape how modern software systems are built by bridging the gap between technical teams and business needs. Its emphasis on modeling the domain with precision and clarity is especially relevant in today's fast-paced, complex software landscape. This book begins with DDD fundamentals, including core principles, a shared language, and the distinction between strategic and tactical approaches, progressing to strategic concepts like bounded contexts, context mapping, and domain events. It explores the tactical Java implementation detailing entities, value objects, services, aggregates, and repositories. The book also explores testing strategies and architectural validation using ArchUnit/jMolecules. Further, it explores DDD across microservices, monoliths, and distributed systems, integrating with Clean Architecture and SQL/NoSQL data modeling to prevent impedance mismatch. It thoroughly covers applying DDD within Jakarta EE, Spring, Eclipse MicroProfile, and Quarkus. By the end, you will be equipped to model business logic more effectively, design systems that reflect real-world domains, and integrate DDD seamlessly into enterprise applications. You will gain clarity, confidence, and the tools needed to build software that delivers business value.

WHAT YOU WILL LEARN

- Apply DDD from strategic to tactical design.
- Model aggregates, entities, and value objects in Java.
- Use DDD in monoliths, microservices, and distributed systems.
- Integrate DDD with Spring and Jakarta EE frameworks.
- Apply Clean Architecture principles alongside DDD.
- Structure data modeling for SQL and NoSQL systems.
- Apply bounded contexts, context mapping, and domain events for architecture.
- Unit/integration testing, validate design with ArchUnit/jMolecules.
- Build responsive microservices with Quarkus extensions, reactive programming.

WHO THIS BOOK IS FOR This book is ideal for Java developers, software architects, tech leads, and backend engineers. It is especially valuable for professionals designing scalable enterprise systems or applying DDD in modern software architecture.

TABLE OF CONTENTS

1. Understanding Domain-driven Design
2. Strategic DDD Concepts
3. Tactical DDD Implementation
4. Testing and Validating DDD Applications
5. DDD in Microservices, Monoliths, and Distributed Systems
6. Integrating DDD with Clean Architecture
7. DDD and Data Modeling
8. Enterprise Java with Jakarta EE
9. Enterprise Java with Spring
10. Eclipse MicroProfile and Domain-driven Design

11. Quarkus and Domain-driven Design 12. Code Design and Best Practices for DDD 13. Final Considerations

hexagonal architecture vs microservices: A Journey Towards Bio-inspired Techniques in Software Engineering Jagannath Singh, Saurabh Bilgaiyan, Bhabani Shankar Prasad Mishra, Satchidananda Dehuri, 2020-03-11 This book covers a range of basic and advanced topics in software engineering. The field has undergone several phases of change and improvement since its invention, and there is significant ongoing research in software development, addressing aspects such as analysis, design, testing and maintenance. Rather than focusing on a single aspect of software engineering, this book provides a systematic overview of recent techniques, including requirement gathering in the form of story points in agile software, and bio-inspired techniques for estimating the effort, cost, and time required for software development. As such it is a valuable resource for new researchers interested in advances in software engineering — particularly in the area of bio-inspired techniques.

hexagonal architecture vs microservices: Software Architecture with Kotlin Jason (Tsz Shun) Chow, 2024-12-31 Develop innovative architectural styles by analyzing and merging various approaches, focusing on making trade-offs and mitigating risks to solve real-world problems Key Features Learn how to analyze and dissect various architectural styles into building blocks Combine existing ideas with your own to create custom solutions Make informed decisions by navigating trade-offs and compromises Purchase of the print or Kindle book includes a free PDF eBook Book Description Software Architecture with Kotlin explores the various styles of software architecture with a focus on using the Kotlin programming language. The author draws on their 20+ years of industry experience in developing large-scale enterprise distributed systems to help you grasp the principles, practices, and patterns that shape the architectural landscape of modern software systems. The book establishes a strong foundation in software architecture, explaining key concepts such as architectural qualities and principles, before teaching you how architectural decisions impact the quality of a system, such as scalability, reliability, and extendability. The chapters address modern architecture topics such as microservices, serverless, and event-driven architectures, providing insights into the challenges and trade-offs involved in adopting these architectural styles. You'll also discover practical tools that'll help you make informed decisions and mitigate risks. All architectural patterns in this book are demonstrated using Kotlin. By the end of this book, you'll have gained practical expertise by using real-world examples, along with a solid understanding of Kotlin, to become a more proficient and impactful software architect. What you will learn Master the fundamental principles of architecture and design Explore common architectural styles and their applicable scenarios Analyze, break down, compare, and design architectural styles to solve practical problems Reason, negotiate, and make difficult choices in the absence of ideal solutions Mitigate risks when making compromises and trade-offs Create scalable, sustainable, maintainable, and extendable software systems Use the Kotlin programming language to achieve your architectural goals Who this book is for This book is for developers with basic Kotlin knowledge seeking a deeper understanding of architecture, Kotlin Android developers who are starting to get involved in backend development, and Java developers transitioning to Kotlin. It's also ideal for software architects who are less experienced in Kotlin and want to enhance their skills, as well as those who enjoy discussing and exploring unique architectural concepts.

hexagonal architecture vs microservices: Clean Architecture with .NET Dino Esposito, 2024-03-12 Understand what to do at any point in developing a clean .NET architecture Master advanced .NET techniques with a focus on actual value delivered by working within a modular, clean architecture. Microsoft Data Platform MVP Dino Esposito explains key clean architecture concepts with a mix of pragmatism and design discipline and helps you solidify your knowledge through a real-world project. Starting with an explanation of the quest for modular software architecture continuing through the methodology of domain-driven design (DDD), Esposito emphasizes the role that modularization plays in managing complexity in software development. Breaking down the layers of an architecture that is modular and maintainable, he presents a sample project that is not

simply another to-do list, but an actual tool for the reader. Ultimately, an exploration of common dilemmas for both developers and operations brings together historical developments with real solutions for today. Microsoft Data Platform MVP Dino Esposito helps you:

- Understand the relevance of modular software architecture in the history of software
- Review domain-driven design concepts both, strategic and practical
- Apply modular analysis techniques to your development
- Make the most of layered architecture
- Make the most of layered architecture that is modular and maintainable
- Explore in detail the individual layers—presentation, application, domain and infrastructure
- Make sense of domain services to separate raw persistence from persistence-related business tasks
- Make your way through a series of C# best-practices for modeling classes from real-world entities
- Understand the benefits of microservices versus modular monoliths
- Understand the analysis of technical shortcuts and benefits of long-term technical investment
- Understand client-side, server-side and other common deployment dilemmas
- Set up your architecture, test your conclusions, and find even more help

hexagonal architecture vs microservices: Serverless Development on AWS Sheen Brisals, Luke Hedger, 2024-01-23 The adoption of serverless is on the rise, but until now, little guidance has been available for development teams that want to apply this technology on AWS. This definitive guide is packed with architectural, security, and data best practices and patterns for architects and engineers who want to build reliable enterprise-scale serverless solutions. Sheen Brisals, an AWS Serverless Hero, and Luke Hedger, an AWS Community Builder, outline the serverless adoption requirements for an enterprise, examine the development tools your team needs, and explain in depth the nuances of testing event-driven and distributed serverless services. You'll gain practical guidance for keeping up with change and learn how to build serverless solutions with sustainability in mind. Examine the serverless technology ecosystem and AWS services needed to develop serverless applications Learn the approach and preparation required for a successful serverless adoption in an enterprise Learn serverless architectures and implementation patterns Design, develop, and test distributed serverless microservices on AWS cloud Apply security best practices while building serverless solutions Identify and adapt the implementation patterns for your particular use case Incorporate the necessary measures for observable serverless applications Implement sustainable serverless applications in the cloud

hexagonal architecture vs microservices: Port for Microservices Developers William Smith, 2025-08-19 Port for Microservices Developers Port for Microservices Developers is an authoritative guide to mastering the Port and Adapter (Hexagonal) Architecture and its vital role in building robust, scalable microservices ecosystems. Designed for professionals and architects seeking clarity and best practices, this book explores the fundamental concepts and evolution of port-centric design. Readers are guided from the origins and core principles of ports and adapters through advanced comparisons with layered and clean architectures, ensuring a deep understanding of how to apply these concepts effectively in distributed, business-critical systems. The book excels in bridging theory and real-world execution by providing practical approaches for defining ports in microservices, implementing adapters with modern frameworks, and integrating enterprise patterns such as orchestration, choreography, and anti-corruption layers. Security is addressed in depth, offering thorough models and techniques for authentication, authorization, data validation, incident response, and compliance. Readers will also learn how port abstractions empower rigorous testing, continuous delivery, observability, and troubleshooting—critical needs for modern DevOps pipelines and production environments. Going beyond foundational knowledge, Port for Microservices Developers addresses the challenges of polyglot persistence, interoperability, dynamic configuration, and scaling in distributed contexts. A strategic perspective is offered for evolving legacy applications, governing cross-organizational ports, and adapting to future industry standards. Rich with examples, actionable patterns, and forward-looking insights, this book is an indispensable companion for advancing both the technical excellence and business agility of microservices teams.

Related to hexagonal architecture vs microservices

Hexagon - Wikipedia From bees' honeycombs to the Giant's Causeway, hexagonal patterns are prevalent in nature due to their efficiency. In a hexagonal grid each line is as short as it can possibly be if a large area

HEXAGONAL Definition & Meaning - Merriam-Webster The meaning of HEXAGONAL is having six angles and six sides. How to use hexagonal in a sentence

HEXAGONAL | definition in the Cambridge English Dictionary Instead, they'll see a colorful geometric shape -- hexagonal if they drive, circular if they're a rider -- surrounding a small, bit-like square

Hexagon - Math is Fun A hexagon is a 6-sided polygon (a flat shape with straight sides): Soap bubbles tend to form hexagons when they join up

HEXAGONAL Definition & Meaning | Hexagonal definition: of, relating to, or having the form of a hexagon.. See examples of HEXAGONAL used in a sentence

HEXAGONAL definition and meaning | Collins English Dictionary A hexagonal object or shape has six straight sides. The rigs will be unmanned and comprise several hexagonal platforms. Collins COBUILD Advanced Learner's Dictionary. Copyright ©

Hexagonal - definition of hexagonal by The Free Dictionary 1. Having six sides. 2. Relating to a crystal having three axes of equal length intersecting at angles of 60° in one plane, and a fourth axis of a different length that is perpendicular to this

Hexagon - Wikipedia From bees' honeycombs to the Giant's Causeway, hexagonal patterns are prevalent in nature due to their efficiency. In a hexagonal grid each line is as short as it can possibly be if a large area

HEXAGONAL Definition & Meaning - Merriam-Webster The meaning of HEXAGONAL is having six angles and six sides. How to use hexagonal in a sentence

HEXAGONAL | definition in the Cambridge English Dictionary Instead, they'll see a colorful geometric shape -- hexagonal if they drive, circular if they're a rider -- surrounding a small, bit-like square

Hexagon - Math is Fun A hexagon is a 6-sided polygon (a flat shape with straight sides): Soap bubbles tend to form hexagons when they join up

HEXAGONAL Definition & Meaning | Hexagonal definition: of, relating to, or having the form of a hexagon.. See examples of HEXAGONAL used in a sentence

HEXAGONAL definition and meaning | Collins English Dictionary A hexagonal object or shape has six straight sides. The rigs will be unmanned and comprise several hexagonal platforms. Collins COBUILD Advanced Learner's Dictionary. Copyright ©

Hexagonal - definition of hexagonal by The Free Dictionary 1. Having six sides. 2. Relating to a crystal having three axes of equal length intersecting at angles of 60° in one plane, and a fourth axis of a different length that is perpendicular to this

Hexagon - Wikipedia From bees' honeycombs to the Giant's Causeway, hexagonal patterns are prevalent in nature due to their efficiency. In a hexagonal grid each line is as short as it can possibly be if a large area

HEXAGONAL Definition & Meaning - Merriam-Webster The meaning of HEXAGONAL is having six angles and six sides. How to use hexagonal in a sentence

HEXAGONAL | definition in the Cambridge English Dictionary Instead, they'll see a colorful geometric shape -- hexagonal if they drive, circular if they're a rider -- surrounding a small, bit-like square

Hexagon - Math is Fun A hexagon is a 6-sided polygon (a flat shape with straight sides): Soap bubbles tend to form hexagons when they join up

HEXAGONAL Definition & Meaning | Hexagonal definition: of, relating to, or having the form of a hexagon.. See examples of HEXAGONAL used in a sentence

HEXAGONAL definition and meaning | Collins English Dictionary A hexagonal object or shape

has six straight sides. The rigs will be unmanned and comprise several hexagonal platforms. Collins COBUILD Advanced Learner's Dictionary. Copyright ©

Hexagonal - definition of hexagonal by The Free Dictionary 1. Having six sides. 2. Relating to a crystal having three axes of equal length intersecting at angles of 60° in one plane, and a fourth axis of a different length that is perpendicular to this

Hexagon - Wikipedia From bees' honeycombs to the Giant's Causeway, hexagonal patterns are prevalent in nature due to their efficiency. In a hexagonal grid each line is as short as it can possibly be if a large area

HEXAGONAL Definition & Meaning - Merriam-Webster The meaning of HEXAGONAL is having six angles and six sides. How to use hexagonal in a sentence

HEXAGONAL | definition in the Cambridge English Dictionary Instead, they'll see a colorful geometric shape -- hexagonal if they drive, circular if they're a rider -- surrounding a small, bit-like square

Hexagon - Math is Fun A hexagon is a 6-sided polygon (a flat shape with straight sides): Soap bubbles tend to form hexagons when they join up

HEXAGONAL Definition & Meaning | Hexagonal definition: of, relating to, or having the form of a hexagon.. See examples of HEXAGONAL used in a sentence

HEXAGONAL definition and meaning | Collins English Dictionary A hexagonal object or shape has six straight sides. The rigs will be unmanned and comprise several hexagonal platforms. Collins COBUILD Advanced Learner's Dictionary. Copyright ©

Hexagonal - definition of hexagonal by The Free Dictionary 1. Having six sides. 2. Relating to a crystal having three axes of equal length intersecting at angles of 60° in one plane, and a fourth axis of a different length that is perpendicular to this

Hexagon - Wikipedia From bees' honeycombs to the Giant's Causeway, hexagonal patterns are prevalent in nature due to their efficiency. In a hexagonal grid each line is as short as it can possibly be if a large area

HEXAGONAL Definition & Meaning - Merriam-Webster The meaning of HEXAGONAL is having six angles and six sides. How to use hexagonal in a sentence

HEXAGONAL | definition in the Cambridge English Dictionary Instead, they'll see a colorful geometric shape -- hexagonal if they drive, circular if they're a rider -- surrounding a small, bit-like square

Hexagon - Math is Fun A hexagon is a 6-sided polygon (a flat shape with straight sides): Soap bubbles tend to form hexagons when they join up

HEXAGONAL Definition & Meaning | Hexagonal definition: of, relating to, or having the form of a hexagon.. See examples of HEXAGONAL used in a sentence

HEXAGONAL definition and meaning | Collins English Dictionary A hexagonal object or shape has six straight sides. The rigs will be unmanned and comprise several hexagonal platforms. Collins COBUILD Advanced Learner's Dictionary. Copyright ©

Hexagonal - definition of hexagonal by The Free Dictionary 1. Having six sides. 2. Relating to a crystal having three axes of equal length intersecting at angles of 60° in one plane, and a fourth axis of a different length that is perpendicular to this

Related to hexagonal architecture vs microservices

From Service-Oriented Architecture to Microservices (CMS Wire5y) Designing our applications as small independent units is the first step towards building a modern infrastructure that is nimble, agile and scalable. Legacy systems still form the backbone of many

From Service-Oriented Architecture to Microservices (CMS Wire5y) Designing our applications as small independent units is the first step towards building a modern infrastructure that is nimble, agile and scalable. Legacy systems still form the backbone of many

Beyond The Architecture Cage Match: How The Microservices Vs. Monoliths Debate Is

Damaging Your Business (Forbes4mon) In the red corner, weighing in with independent scalability and distributed complexity: microservices! In the blue corner, the reigning legacy champion, with its infamous deployment challenges: the

Beyond The Architecture Cage Match: How The Microservices Vs. Monoliths Debate Is Damaging Your Business (Forbes4mon) In the red corner, weighing in with independent scalability and distributed complexity: microservices! In the blue corner, the reigning legacy champion, with its infamous deployment challenges: the

Microservices 101: A guide to microservice architecture (ZDNet5y) The idea behind microservices and a microservices architecture is relatively simple: hide all the complexities of hardware, operating systems, and different development toolkits behind a standard

Microservices 101: A guide to microservice architecture (ZDNet5y) The idea behind microservices and a microservices architecture is relatively simple: hide all the complexities of hardware, operating systems, and different development toolkits behind a standard

The downsides of microservices architecture (InfoWorld2y) Microservices came in with a great deal of momentum a few years ago, but now we're seeing their drawbacks for applications on cloud platforms. A microservices architecture for application development

The downsides of microservices architecture (InfoWorld2y) Microservices came in with a great deal of momentum a few years ago, but now we're seeing their drawbacks for applications on cloud platforms. A microservices architecture for application development

Why microservices need event-driven architecture (ZDNet3y) Microservices promise to help break down monolithic applications and enable the consistent delivery of services. But they can't do the job without help. This is where event-driven architecture (EDA)

Why microservices need event-driven architecture (ZDNet3y) Microservices promise to help break down monolithic applications and enable the consistent delivery of services. But they can't do the job without help. This is where event-driven architecture (EDA)

What are Microservices and Can this Architecture Improve Your Application Development? (CMS Wire7y) Big brands like Amazon and Netflix are embracing microservice architecture, but what is it, and how does it fit into application and software development? The software products we use every day for

What are Microservices and Can this Architecture Improve Your Application Development? (CMS Wire7y) Big brands like Amazon and Netflix are embracing microservice architecture, but what is it, and how does it fit into application and software development? The software products we use every day for

Back to Home: <http://www.speargroupllc.com>