microservices sequence diagram example

Microservices Sequence Diagram Example: Visualizing Interactions in Modern Architectures

microservices sequence diagram example offers an insightful way to visualize the dynamic interactions between various components in a microservices architecture. As businesses increasingly adopt microservices for their flexibility and scalability, understanding the flow of communication between services becomes essential. Sequence diagrams, a type of UML (Unified Modeling Language) diagram, provide a clear, step-by-step view of how microservices collaborate to fulfill user requests or internal processes.

In this article, we'll explore what a microservices sequence diagram is, why it's crucial in designing and maintaining microservices, and dive into a practical microservices sequence diagram example that highlights common patterns and challenges. Whether you're a developer, architect, or product manager, grasping this concept can enhance your ability to build resilient, efficient microservices systems.

Understanding Microservices and Sequence Diagrams

Before jumping into the example, it helps to clarify the building blocks. Microservices architecture breaks down a large application into smaller, independent services that communicate through APIs. Each microservice handles a specific business capability, which allows for easier updates, scalability, and deployment.

Sequence diagrams, on the other hand, are interaction diagrams used to showcase how different components or objects interact in a time-ordered sequence. They visually represent the exchange of messages, method calls, and responses between entities, making them perfect for mapping out microservices communication.

Why Use Sequence Diagrams for Microservices?

Microservices often involve complex communication patterns, including synchronous calls, asynchronous messaging, event-driven interactions, and more. Sequence diagrams help:

- Clarify the flow of requests and responses between services.
- Identify potential bottlenecks or points of failure.
- Document workflows for onboarding and collaboration.
- Aid in debugging by visualizing where issues might arise.

- Support design decisions around service boundaries and responsibilities.

By leveraging sequence diagrams, teams gain a shared understanding of how microservices collaborate, which is invaluable in distributed system design.

Microservices Sequence Diagram Example: An E-commerce Order Processing Flow

To make things tangible, let’s consider a common scenario in an e-commerce platform: processing an order. This workflow typically involves several microservices such as the User Service, Order Service, Inventory Service, Payment Service, and Notification Service. Below is a detailed microservices sequence diagram example illustrating the interaction steps.

Step-by-Step Interaction

1. **User Places Order**
The process begins when the User Service receives an order request from the customer interface.

2. **Order Service Validates and Creates Order**
The User Service forwards the order details to the Order Service, which validates the data and creates an order record.

3. **Inventory Check**
The Order Service calls the Inventory Service to verify if the requested items are in stock.

4. **Inventory Confirmation or Rejection**
The Inventory Service confirms availability, reserving the stock if possible, or rejects the request if items are out of stock.

5. **Payment Processing**
Upon successful inventory confirmation, the Order Service triggers the Payment Service to process the payment.

6. **Payment Confirmation**
The Payment Service authorizes the payment and sends a confirmation back to the Order Service.

7. **Order Status Update**
The Order Service updates the order status to ‘Confirmed’ and triggers the Notification Service.

8. **User Notification**
The Notification Service sends an email or SMS to the customer confirming the

order.

Visualizing the Sequence Diagram

In a microservices sequence diagram depicting this flow, you'd see vertical lifelines representing each service (User, Order, Inventory, Payment, Notification). Horizontal arrows indicate the messages or API calls flowing between them in chronological order. This visualization helps identify synchronous calls (e.g., Order Service waiting for inventory confirmation) and asynchronous events (like notifications sent after order confirmation).

Best Practices for Creating Effective Microservices Sequence Diagrams

Creating detailed and clear sequence diagrams for microservices can be challenging due to the complexity and scale of interactions. Here are some tips to keep in mind:

1. Focus on Specific Use Cases

Avoid trying to capture the entire system in one diagram. Instead, narrow down on a particular business process or user interaction. This keeps the diagram manageable and meaningful.

2. Distinguish Between Synchronous and Asynchronous Communication

Use solid arrows for synchronous calls and dashed arrows for asynchronous messages or events. This distinction clarifies the communication pattern and potential latency points.

3. Represent Error Handling and Timeouts

Include alternative flows showing what happens if a service fails or times out. This enhances the diagram's usefulness in understanding system resilience.

4. Use Consistent Naming and Notation

Maintain clear labels for services, methods, and messages to avoid confusion. Consistency helps when diagrams are shared across teams.

5. Integrate with Other UML Diagrams

Sequence diagrams complement component diagrams, deployment diagrams, and data flow diagrams. Use them in tandem for a holistic view of your microservices architecture.

Tools for Designing Microservices Sequence Diagrams

Several tools can assist in creating visually appealing and accurate sequence diagrams tailored for microservices:

- **PlantUML:** A text-based UML tool that supports sequence diagrams and can integrate into CI/CD pipelines.
- **Lucidchart:** Offers drag-and-drop functionality with templates for microservices architecture diagrams.
- **Draw.io (diagrams.net):** Free and versatile, suitable for quick diagramming.
- **Microsoft Visio:** A professional tool with robust UML support.
- **SequenceDiagram.org:** A simple online tool focused solely on sequence diagrams.

Choosing a tool often depends on your team’s workflow, collaboration needs, and integration with other documentation practices.

Challenges When Modeling Microservices Interactions

While sequence diagrams are helpful, modeling microservices interactions comes with its own set of challenges:

- **Handling Asynchronous Messaging:** Many microservices communicate via message queues or event streams, which can be tricky to represent sequentially.
- **Scaling Complexity:** As the number of services grows, diagrams can become cluttered and hard to read.
- **Dynamic Service Discovery:** Microservices may dynamically discover and call other services, which can be difficult to capture statically.
- **Versioning and Evolution:** Services evolve independently, requiring diagrams to be updated constantly to stay relevant.

To overcome these, consider creating layered diagrams that show high-level flows first, then zoom into specific interactions. Incorporate notes or annotations to explain complex parts.

Integrating Microservices Sequence Diagrams in Development Workflows

Incorporating sequence diagrams into your development lifecycle can improve communication and reduce misunderstandings. Here’s how:

- **Design Phase:** Use diagrams to plan service interactions before coding begins.
- **Code Reviews:** Reference diagrams to ensure implementation matches design.
- **Onboarding:** Help new team members understand system workflows quickly.
- **Incident Analysis:** Map out failed interactions during troubleshooting.
- **Documentation:** Maintain diagrams as living documents alongside code.

By embedding sequence diagrams into these stages, teams foster a culture of clarity and shared understanding.

Microservices sequence diagram example scenarios like the e-commerce order processing flow bring to life the invisible choreography happening behind user actions. They help bridge the gap between abstract architecture and concrete implementation details. Whether you’re designing new microservices or maintaining existing ones, investing time in creating and refining sequence diagrams will pay dividends in smoother development, easier debugging, and better collaboration.

Frequently Asked Questions

What is a microservices sequence diagram example?

A microservices sequence diagram example illustrates the interaction between multiple microservices over time, showing the sequence of messages exchanged to complete a business process or transaction.

Why are sequence diagrams important in microservices architecture?

Sequence diagrams help visualize the communication flow between microservices, making it easier to understand dependencies, identify bottlenecks, and design robust and scalable distributed systems.

Can you provide a simple microservices sequence

diagram example for an e-commerce checkout process?

Yes. In the sequence diagram, the client sends a checkout request to the Order Service, which communicates with Inventory Service to check stock, then with Payment Service to process payment, and finally confirms the order back to the client.

How do microservices sequence diagrams handle asynchronous communication?

Asynchronous communication in microservices sequence diagrams is represented using asynchronous message arrows, often with return messages shown separately or omitted, highlighting the non-blocking nature of the interaction.

What tools can be used to create microservices sequence diagrams?

Popular tools include UML modeling software like Lucidchart, Microsoft Visio, PlantUML, Draw.io, and online diagramming platforms that support sequence diagram creation.

How detailed should a microservices sequence diagram example be?

The level of detail depends on the audience; for high-level understanding, focus on main service interactions, while for technical design, include message payloads, error handling, and asynchronous flows.

Can microservices sequence diagrams help in debugging and monitoring?

Yes, they provide a clear representation of service interactions, which can assist in tracing issues, understanding failure points, and improving monitoring strategies in complex microservices environments.

Are there best practices for creating microservices sequence diagrams?

Best practices include keeping diagrams focused on specific use cases, clearly labeling services and messages, differentiating synchronous and asynchronous calls, and regularly updating diagrams to reflect system changes.

Additional Resources

Microservices Sequence Diagram Example: A Detailed Exploration of Inter-Service Communication

microservices sequence diagram example serves as a crucial tool for architects, developers, and system analysts aiming to visualize and comprehend the complex interactions within distributed systems. As microservices architecture gains widespread adoption due to its scalability and modularity, understanding the flow of communication between independent services becomes imperative. Sequence diagrams, a subset of UML (Unified Modeling Language) diagrams, provide a timeline-based representation of how various components within a microservices ecosystem collaborate to fulfill a business process.

This article delves into the practical application of microservices sequence diagrams, illustrating how these diagrams map inter-service communication patterns, facilitate debugging, and enhance system documentation. By examining a concrete microservices sequence diagram example, we aim to clarify the nuances of asynchronous messaging, service orchestration, and fault tolerance inherent in microservices architectures.

Understanding Microservices and Their Communication Complexities

Microservices architecture decomposes a monolithic application into a collection of loosely coupled services, each responsible for a specific business capability. While this modular approach offers flexibility and independent deployment cycles, it introduces challenges in managing communication and data consistency across services.

Unlike monolithic systems where function calls happen within a single process, microservices often communicate over the network using HTTP/REST, gRPC, message queues, or event-driven mechanisms. This distributed nature leads to intricate interaction patterns, making it difficult to trace the sequence of calls and responses during a transaction lifecycle without proper visualization tools.

The Role of Sequence Diagrams in Microservices Design

Sequence diagrams graphically represent the order of message exchanges between services over time. They visualize the lifelines of interacting services and depict synchronous or asynchronous calls, responses, and exceptions.

By employing a microservices sequence diagram example, developers can:

- Clarify service dependencies and interaction flows
- Identify potential bottlenecks or points of failure
- Document and communicate service contracts and APIs
- Support debugging by tracing real-time message flows
- Facilitate design reviews and architectural decisions

The diagram acts as a blueprint for both implementation and operational monitoring, bridging the gap between conceptual design and practical execution.

Microservices Sequence Diagram Example: An E-Commerce Order Processing Flow

To concretize the concept, consider an e-commerce platform where a customer places an order. The order processing involves multiple microservices:

1. API Gateway
2. Order Service
3. Inventory Service
4. Payment Service
5. Notification Service

Below is a breakdown of the interaction sequence:

1. The client sends an order request to the API Gateway.
2. The API Gateway forwards the request to the Order Service.
3. The Order Service validates the order and checks inventory by calling the Inventory Service.
4. The Inventory Service confirms product availability.

5. The Order Service then initiates payment processing via the Payment Service.
6. The Payment Service processes the payment and returns the result.
7. Upon successful payment, the Order Service updates the order status.
8. The Order Service triggers a notification through the Notification Service to inform the customer.

This sequence can be represented in a microservices sequence diagram example, capturing synchronous calls such as the Inventory Service check and asynchronous events like notifications.

Key Features Highlighted in the Diagram

- **Lifelines:** Each microservice is represented by a vertical lifeline showing its active period.
- **Messages:** Arrows indicate requests and responses, distinguishing synchronous (solid lines) and asynchronous (dashed lines) communication.
- **Activation Bars:** Highlight the time during which a service is active in processing a request.
- **Conditional Flows:** Represent checks such as inventory availability and payment success.
- **Exception Handling:** Optionally denotes error scenarios, e.g., payment failure triggering compensating actions.

The diagram's visual clarity aids in understanding how responsibilities are divided and how services collaborate to complete the order processing workflow.

Comparing Synchronous vs. Asynchronous Communication in Sequence Diagrams

Microservices often leverage a mix of synchronous and asynchronous communication, each with trade-offs that the sequence diagram must accurately depict.

Synchronous Communication

Synchronous calls imply a request-response pattern where the caller waits for the callee to process and return a result before proceeding. In the example, the Order Service’s calls to Inventory and Payment Services are synchronous, as each step depends on the previous operation's success.

Pros:

- Simple to implement and understand
- Ensures immediate feedback for dependent services

Cons:

- Can lead to tight coupling and reduced fault tolerance
- Potential latency if downstream services are slow

Asynchronous Communication

Asynchronous messaging allows services to communicate via events or message queues, decoupling sender and receiver temporally. The Notification Service receiving events after order completion exemplifies this pattern.

Pros:

- Improves scalability and resilience
- Enables event-driven architectures

Cons:

- Increased complexity in ensuring message delivery and ordering
- Harder to trace and debug without adequate tooling

A detailed microservices sequence diagram example must differentiate these communication types, often using distinct arrow styles or annotations, to provide an accurate system portrayal.

Tools and Best Practices for Creating Microservices Sequence Diagrams

Several tools facilitate the creation of sequence diagrams tailored to microservices environments:

- **PlantUML:** Text-based diagram generation supporting integration with CI/CD pipelines.
- **Lucidchart:** Collaborative online diagramming with templates for microservices.
- **SequenceDiagram.org:** Lightweight, web-based sequence diagram editor.
- **Enterprise Architect:** Comprehensive UML modeling with support for large-scale systems.

Best practices include:

- Maintain simplicity by focusing on critical interactions rather than exhaustive detail.
- Use consistent notation to distinguish synchronous vs. asynchronous calls.
- Incorporate error flows and retries to reflect real-world scenarios.
- Keep diagrams updated alongside evolving service APIs and workflows.
- Leverage color coding or annotations for clarity and emphasis.

Adhering to these guidelines ensures that the microservices sequence diagram example remains a valuable asset throughout the software development lifecycle.

Challenges in Modeling Microservices with Sequence Diagrams

Despite their utility, sequence diagrams for microservices face inherent challenges:

- **Scalability:** Large-scale systems with dozens of microservices can result in overly complex diagrams that are difficult to interpret.
- **Dynamic Behavior:** Microservices may exhibit dynamic routing or circuit breaker patterns, complicating static sequence representations.
- **Asynchronous Messaging:** Event-driven interactions may span long periods, making linear sequence diagrams less suitable.
- **Versioning and Evolution:** Frequent changes in service interfaces require continuous updates to diagrams, risking outdated documentation.

Mitigating these issues often involves creating focused diagrams for specific use cases or workflows instead of attempting to model the entire system in a single diagram.

Integrating Sequence Diagrams with Modern Microservices Practices

The rise of DevOps, continuous integration/continuous deployment (CI/CD), and observability tools influences how sequence diagrams are used in microservices contexts. Automated generation of sequence diagrams from distributed tracing data, for example, offers a dynamic and accurate reflection of runtime interactions, complementing manually crafted diagrams.

Moreover, combining sequence diagrams with architecture decision records (ADRs) and API specifications (e.g., OpenAPI) enhances communication across cross-functional teams. This integrated approach supports faster onboarding, better alignment, and proactive identification of architectural risks.

In essence, a microservices sequence diagram example is not merely a static illustration but a dynamic instrument that encapsulates the intricate choreography of services within distributed systems. When thoughtfully constructed and maintained, these diagrams illuminate the pathways of inter-service communication, foster shared understanding, and guide the evolution of resilient, scalable microservices architectures.

Microservices Sequence Diagram Example

microservices sequence diagram example: Microservices by Example Biswa Pujarini Mohapatra, Baishakhi Banerjee, Gaurav Aroraa, 2019-01-24 A book with lot of practical and architectural styles for Microservices using .NET Core DESCRIPTION This book predominately covers Microservices architecture with real-world example which can help professionals on case adoption of this technology. Following the trend of modularity in real world, the idea behind Microservice by Examples is to allow developers to build their applications from various independent components which can be easily changed, removed or upgraded. Also, it is relevant now because of enterprises are moving towards DevOps/ Modernisation, this book will emphasise on containers and Dockers as well. Ê KEY FEATURES ¥ Ê Ê Ê Understand core concept of Microservices ¥ Ê Ê Ê Understand various Microservices design patterns ¥ Ê Ê Ê Build microservices application using real-world examples ¥ Ê Ê Ê Deployment of microservices using Docker ¥ Ê Ê Ê Microservices Orchestration using Azure Service Fabric ¥ Ê Ê Ê Azure DevOps (CI/CD) using MSBuild ¥ Ê Ê Ê Understand the concept of API Management ¥ Ê Ê Ê Authentication/Authorization using JWT token for Microservices ¥ Ê Ê Ê Integrating Microservices in Angular 6.0 Single Page Application. ¥ Ê Ê Ê Dos and donÕts during integration ¥ Ê Ê Ê Ensuring End to end testing Ê WHAT WILL YOU LEARN ¥Ê Ê Ê Microservices and its ArchitectureÊ ¥Ê Ê Ê Designing the microservice application layerÊ ¥Ê Ê Ê Hands on Micro services development of Online Hotel Booking AppÊ ¥Ê Ê Ê Deployment of Microservices for App-Modernization at Scale with DockerÊ ¥Ê Ê Ê Service Orchestration of Microservices using Azure Service FabricÊ ¥Ê Ê Ê Integrating various componentsÊÊ ¥Ê Ê Ê Hands on Integration with API Management ¥Ê Ê Ê Testing MicroservicesÊ Ê WHO THIS BOOK IS FOR This book is for .NET Core developers who are new to microservices and want to learn, understand the microservices architecture. Ê Table of Contents 1. Ê Ê An introduction to MicroservicesÊ 2. Ê Ê Micro services ArchitectureÊ 3. Ê Ê Designing the microservice application layerÊ 4. Ê Ê Hands on Micro services development of Online Hotel Booking AppÊ 5. Ê Ê Deployment of Microservices for App-Modernization at Scale with DockerÊ 6. Ê Ê Service Orchestration of Microservices using Azure Service FabricÊ 7. Ê Ê Integrating various componentsÊÊ 8. Ê Ê Hands on Integration with API Management 9. Ê Ê Testing MicroservicesÊ 10. Ê Extending application with loggingÊ 11. Ê What is next?

microservices sequence diagram example: Microservices for the Enterprise Kasun Indrasiri, Prabath Siriwardena, 2018-11-14 Understand the key challenges and solutions around building microservices in the enterprise application environment. This book provides a comprehensive understanding of microservices architectural principles and how to use microservices in real-world scenarios. Architectural challenges using microservices with service integration and API management are presented and you learn how to eliminate the use of centralized integration products such as the enterprise service bus (ESB) through the use of composite/integration microservices. Concepts in the book are supported with use cases, and emphasis is put on the reality that most of you are implementing in a “brownfield” environment in which you must implement microservices alongside legacy applications with minimal disruption to your business. Microservices for the Enterprise covers state-of-the-art techniques around microservices messaging, service development and description, service discovery, governance, and data management technologies and guides you through the microservices design process. Also included is the importance of organizing services as core versus atomic, composite versus integration, and API versus edge, and how such organization helps to eliminate the use of a central ESB and expose services through an API gateway. What You'll Learn Design and develop microservices architectures with confidence Put into practice the most modern techniques around messaging technologies Apply the Service Mesh pattern to overcome inter-service communication challenges Apply battle-tested microservices security patterns to address real-world scenarios Handle API management, decentralized data management, and observability Who This Book Is For Developers and DevOps engineers responsible for implementing applications around a microservices architecture, and architects and analysts who are designing such systems

microservices sequence diagram example: Microservices: Up and Running Ronnie Mitra, Irakli Nadareishvili, 2020-11-25 Microservices architectures offer faster change speeds, better scalability, and cleaner, evolvable system designs. But implementing your first microservices architecture is difficult. How do you make myriad choices, educate your team on all the technical details, and navigate the organization to a successful execution to maximize your chance of success? With this book, authors Ronnie Mitra and Irakli Nadareishvili provide step-by-step guidance for building an effective microservices architecture. Architects and engineers will follow an implementation journey based on techniques and architectures that have proven to work for microservices systems. You'll build an operating model, a microservices design, an infrastructure foundation, and two working microservices, then put those pieces together as a single implementation. For anyone tasked with building microservices or a microservices architecture, this guide is invaluable. Learn an effective and explicit end-to-end microservices system design Define teams, their responsibilities, and guidelines for working together Understand how to slice a big application into a collection of microservices Examine how to isolate and embed data into corresponding microservices Build a simple yet powerful CI/CD pipeline for infrastructure changes Write code for sample microservices Deploy a working microservices application on Amazon Web Services

microservices sequence diagram example: Testing Java Microservices Jason Porter, Alex Soto, Andrew Gumbrecht, 2018-08-03 Summary Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. You'll learn how to increase your test coverage and productivity, and gain confidence that your system will work as you expect. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Microservice applications present special testing challenges. Even simple services need to handle unpredictable loads, and distributed message-based designs pose unique security and performance concerns. These challenges increase when you throw in asynchronous communication and containers. About the Book Testing Java Microservices teaches you to implement unit and integration tests for microservice systems running on the JVM. You'll work with a microservice environment built using Java EE, WildFly Swarm, and Docker. You'll advance from writing simple unit tests for individual services to more-advanced practices like chaos or integration tests. As you move towards a continuous-delivery pipeline, you'll also master live system testing using technologies like the Arquillian, Wiremock, and Mockito frameworks, along with techniques like contract testing and over-the-wire service virtualization. Master these microservice-specific practices and tools and you'll greatly increase your test coverage and productivity, and gain confidence that your system will work as you expect. What's Inside Test automation Integration testing microservice systems Testing container-centric systems Service virtualization About the Reader Written for Java developers familiar with Java EE, EE4J, Spring, or Spring Boot. About the Authors Alex Soto Bueno and Jason Porter are Arquillian team members. Andy Gumbrecht is an Apache TomEE developer and PMC. They all have extensive enterprise-testing experience. Table of Contents An introduction to microservices Application under test Unit-testing microservices Component-testing microservices Integration-testing microservices Contract tests End-to-end testing Docker and testing Service virtualization Continuous delivery in microservices

microservices sequence diagram example: Ultimate Microservices with RabbitMQ Peter Morlion, 2024-05-28 TAGLINE Learn the ins and outs of a RabbitMQ-enabled microservices system. KEY FEATURES ● Discover the fundamental principles of microservices and their organizational impact. ● Develop a deep understanding of messaging and RabbitMQ within microservices architecture. ● Acquire the expertise to seamlessly integrate and maximize the benefits of microservices with RabbitMQ for creating of robust and high-performing systems. DESCRIPTION Embark on a transformative journey into the world of microservices architecture with 'Ultimate Microservices with RabbitMQ' This comprehensive resource equips you with the knowledge and

tools needed to design, deploy, and manage scalable microservices architectures using RabbitMQ as a messaging backbone. From laying the groundwork with foundational concepts to implementing advanced techniques, this book covers everything you need to know to build resilient and high-performing microservices-based systems. It dives into the intricacies of RabbitMQ messaging to leverage its capabilities to facilitate communication and data exchange across distributed systems. You will discover best practices for designing scalable and fault-tolerant microservices architectures, and gain insights into effective integration strategies. The book will help you learn to combine microservices and RabbitMQ for designing, building and maintaining robust architectures that leverage the strengths of both paradigms. By the end of this book, you will be primed to navigate the complexities of modern software development with confidence, driving innovation and efficiency in professional endeavors. WHAT WILL YOU LEARN ● Gain a solid understanding of microservices fundamentals and their organizational impact. ● Explore various messaging paradigms and their application within RabbitMQ. ● Implement RabbitMQ as a message broker within your microservices architecture. ● Understand the prerequisites for maintaining a resilient microservices setup with RabbitMQ. ● Explore upcoming trends in message-driven microservices architectures. WHO IS THIS BOOK FOR? This book is tailored for software developers, architects, and engineering managers intrigued by microservices and messaging with RabbitMQ. Whether you're an entry-level developer or a seasoned architect, this book offers valuable insights and guidance to help you grasp the fundamental concepts and practical considerations essential for navigating the complexities of microservices with RabbitMQ. TABLE OF CONTENTS 1. An Introduction to Microservices 2. A Deeper Look Into Microservices 3. An Introduction to RabbitMQ 4. RabbitMQ Use Cases 5. Designing a Microservices Architecture With RabbitMQ 6. Running A Microservices Architecture With RabbitMQ 7. The Future of Microservices 8. The Future of RabbitMQ Index

microservices sequence diagram example: Spring 5.0 Microservices Rajesh R V, 2017-07-13 A practical, comprehensive, and user-friendly approach to building microservices in Spring About This Book Update existing applications to integrate reactive streams released as a part of Spring 5.0 Learn how to use Docker and Mesos to push the boundaries and build successful microservices Upgrade the capability model to implement scalable microservices Who This Book Is For This book is ideal for Spring developers who want to build cloud-ready, Internet-scale applications, and simple RESTful services to meet modern business demands. What You Will Learn Familiarize yourself with the microservices architecture and its benefits Find out how to avoid common challenges and pitfalls while developing microservices Use Spring Boot and Spring Cloud to develop microservices Handle logging and monitoring microservices Leverage Reactive Programming in Spring 5.0 to build modern cloud native applications Manage internet-scale microservices using Docker, Mesos, and Marathon Gain insights into the latest inclusion of Reactive Streams in Spring and make applications more resilient and scalable In Detail The Spring Framework is an application framework and inversion of the control container for the Java platform. The framework's core features can be used by any Java application, but there are extensions to build web applications on top of the Java EE platform. This book will help you implement the microservice architecture in Spring Framework, Spring Boot, and Spring Cloud. Written to the latest specifications of Spring that focuses on Reactive Programming, you'll be able to build modern, internet-scale Java applications in no time. The book starts off with guidelines to implement responsive microservices at scale. Next, you will understand how Spring Boot is used to deploy serverless autonomous services by removing the need to have a heavyweight application server. Later, you'll learn how to go further by deploying your microservices to Docker and managing them with Mesos. By the end of the book, you will have gained more clarity on the implementation of microservices using Spring Framework and will be able to use them in internet-scale deployments through real-world examples. Style and approach The book takes a step-by-step approach on developing microservices using Spring Framework, Spring Boot, and a set of Spring Cloud components that will help you scale your applications.

microservices sequence diagram example: Microservices in Action Morgan Bruce, Paulo A

Pereira, 2018-10-03 The one [and only] book on implementing microservices with a real-world, cover-to-cover example you can relate to. - Christian Bach, Swiss Re Microservices in Action is a practical book about building and deploying microservice-based applications. Written for developers and architects with a solid grasp of service-oriented development, it tackles the challenge of putting microservices into production. Purchase of the print book includes a free eBook in PDF, Kindle, and ePub formats from Manning Publications. About the Technology Invest your time in designing great applications, improving infrastructure, and making the most out of your dev teams. Microservices are easier to write, scale, and maintain than traditional enterprise applications because they're built as a system of independent components. Master a few important new patterns and processes, and you'll be ready to develop, deploy, and run production-quality microservices. About the Book Microservices in Action teaches you how to write and maintain microservice-based applications. Created with day-to-day development in mind, this informative guide immerses you in real-world use cases from design to deployment. You'll discover how microservices enable an efficient continuous delivery pipeline, and explore examples using Kubernetes, Docker, and Google Container Engine. What's inside An overview of microservice architecture Building a delivery pipeline Best practices for designing multi-service transactions and queries Deploying with containers Monitoring your microservices About the Reader Written for intermediate developers familiar with enterprise architecture and cloud platforms like AWS and GCP. About the Author Morgan Bruce and Paulo A. Pereira are experienced engineering leaders. They work daily with microservices in a production environment, using the techniques detailed in this book. Table of Contents Designing and running microservices Microservices at SimpleBank Architecture of a microservice application Designing new features Transactions and queries in microservices Designing reliable services Building a reusable microservice framework Deploying microservices Deployment with containers and schedulers Building a delivery pipeline for microservices Building a monitoring system Using logs and traces to understand behavior Building microservice teams PART 1 - The lay of the land PART 2 - Design PART 3 - Deployment PART 4 - Observability and ownership

microservices sequence diagram example: Information Science and Applications 2018 Kuinam J. Kim, Nakhoon Baek, 2018-07-23 This book contains selected papers from the 9th International Conference on Information Science and Applications (ICISA 2018) and provides a snapshot of the latest issues encountered in technical convergence and convergences of security technology. It explores how information science is core to most current research, industrial and commercial activities and consists of contributions covering topics including Ubiquitous Computing, Networks and Information Systems, Multimedia and Visualization, Middleware and Operating Systems, Security and Privacy, Data Mining and Artificial Intelligence, Software Engineering, and Web Technology. The proceedings introduce the most recent information technology and ideas, applications and problems related to technology convergence, illustrated through case studies, and reviews converging existing security techniques. Through this volume, readers will gain an understanding of the current state-of-the-art information strategies and technologies of convergence security. The intended readership includes researchers in academia, industry and other research institutes focusing on information science and technology.

microservices sequence diagram example: *Evolve the Monolith to Microservices with Java and Node* Sandro De Santis, Luis Florez, Duy V Nguyen, Eduardo Rosa, IBM Redbooks, 2016-12-05 Microservices is an architectural style in which large, complex software applications are composed of one or more smaller services. Each of these microservices focuses on completing one task that represents a small business capability. These microservices can be developed in any programming language. This IBM® Redbooks® publication shows how to break out a traditional Java EE application into separate microservices and provides a set of code projects that illustrate the various steps along the way. These code projects use the IBM WebSphere® Application Server Liberty, IBM API ConnectTM, IBM Bluemix®, and other Open Source Frameworks in the microservices ecosystem. The sample projects highlight the evolution of monoliths to microservices with Java and Node.

microservices sequence diagram example: Spring System Design in Practice Rodrigo Santiago, 2025-05-05 Transform raw requirements into scalable, resilient web applications with Spring, and build secure, high-performance systems from the ground up using expert guidance and best practices Key Features Establish yourself as the go-to person for strategic decision-making and solutions design with proven strategies Write clean, modular, and testable code with the power of dependency injection Optimize your development efforts by artfully connecting diverse use cases with the right Spring components Purchase of the print or Kindle book includes a free PDF eBook Book DescriptionSoftware system design goes beyond just writing code—it requires a structured approach to translating real-world requirements into scalable, maintainable solutions. With Rodrigo Santiago's hands-on mentoring style and Java Spring expertise, he makes system design accessible to developers at all levels. Spring System Design in Practice guides you through building robust software architectures with Spring. From breaking down complex business needs into actionable use cases to implementing services using Spring Boot, this book equips you with the tools and best practices needed for developing secure, high-performance applications. You'll explore inter-service communication, security, and aspect-oriented programming to streamline development. Covering microservices architecture, the book demonstrates how to create self-configuring, resilient, and event-driven services that integrate seamlessly into the cloud. Through hands-on experience, you'll apply best practices to enhance reliability and scalability while tackling complex challenges such as state management, resilience patterns, concurrency issues, and distributed transactions—including bottlenecks related to asynchronous and reactive programming.By the end of this book, you'll have the confidence to analyze system requirements and design well-structured, scalable architectures.What you will learn Implement microservices for scalable, resilient web systems Break down business goals into well-structured product requirements Weigh tradeoffs between writing asynchronous vs. synchronous services and SQL vs. NoSQL storage Accelerate service development and reliability through the adoption of test-driven development Identify and eliminate hidden performance bottlenecks to maximize speed and efficiency Achieve real-time processing and responsiveness in distributed environments Who this book is for If you're an entry-level IT professional with junior to mid-level Java and Spring experience, this pragmatic guide will fast-track your journey to mastering the Spring ecosystem. Designed to accelerate your career path toward becoming a senior software engineer, system architect, technical lead, or aspiring CTO, it provides clear explanations of the Spring ecosystem and its intricate patterns. For experienced developers or architects, this book offers essential insights and hands-on knowledge to deepen your architectural skills and design resilient web systems.

microservices sequence diagram example: Design It! Michael Keeling, 2017-10-18 Don't engineer by coincidence-design it like you mean it! Filled with practical techniques, Design It! is the perfect introduction to software architecture for programmers who are ready to grow their design skills. Lead your team as a software architect, ask the right stakeholders the right questions, explore design options, and help your team implement a system that promotes the right -ilities. Share your design decisions, facilitate collaborative design workshops that are fast, effective, and fun-and develop more awesome software! With dozens of design methods, examples, and practical know-how, Design It! shows you how to become a software architect. Walk through the core concepts every architect must know, discover how to apply them, and learn a variety of skills that will make you a better programmer, leader, and designer. Uncover the big ideas behind software architecture and gain confidence working on projects big and small. Plan, design, implement, and evaluate software architectures and collaborate with your team, stakeholders, and other architects. Identify the right stakeholders and understand their needs, dig for architecturally significant requirements, write amazing quality attribute scenarios, and make confident decisions. Choose technologies based on their architectural impact, facilitate architecture-centric design workshops, and evaluate architectures using lightweight, effective methods. Write lean architecture descriptions people love to read. Run an architecture design studio, implement the architecture you've designed, and grow your team's architectural knowledge. Good design requires good communication. Talk

about your software architecture with stakeholders using whiteboards, documents, and code, and apply architecture-focused design methods in your day-to-day practice. Hands-on exercises, real-world scenarios, and practical team-based decision-making tools will get everyone on board and give you the experience you need to become a confident software architect.

microservices sequence diagram example: Transforming Your Business with AWS Philippe Abdoulaye, 2021-10-06 Expert guidance on how to use Amazon Web Services to supercharge your digital services business In Transforming Your Business with AWS: Getting the Most Out of Using AWS to Modernize and Innovate Your Digital Services, renowned international consultant and sought-after speaker Philippe Abdoulaye delivers a practical and accessible guide to using Amazon Web Services to modernize your business and the digital services you offer. This book provides you with a concrete action plan to build a team capable of creating world-class digital services and long-term competitive advantages. You'll discover what separates merely average digital service organizations from the truly outstanding, as well as how moving to the cloud will enable your business to deliver your services faster, better, and more efficiently. This book also includes: A comprehensive overview of building industry-leading digital service delivery capabilities, including discussions of the development lifecycle, best practices, and AWS-based development infrastructure Explanations of how to implement a digital business transformation strategy An exploration of key roles like DevOps Continuous Delivery, Continuous Deployment, Continuous Integration, Automation, and DevSecOps Hands-on treatments of AWS application management tools, including Elastic Beanstalk, CodeDeploy, and CodePipeline Perfect for executives, managers, and other business leaders attempting to clarify and implement their organization's digital vision and strategy, Transforming Your Business with AWS is a must-read reference that answers the why and, most importantly, the how, of digital transformation with Amazon Web Services.

microservices sequence diagram example: Microservice Patterns and Best Practices Vinicius Feitosa Pacheco, 2018-01-31 Explore the concepts and tools you need to discover the world of microservices with various design patterns Key Features Get to grips with the microservice architecture and build enterprise-ready microservice applications Learn design patterns and the best practices while building a microservice application Obtain hands-on techniques and tools to create high-performing microservices resilient to possible fails Book Description Microservices are a hot trend in the development world right now. Many enterprises have adopted this approach to achieve agility and the continuous delivery of applications to gain a competitive advantage. This book will take you through different design patterns at different stages of the microservice application development along with their best practices. Microservice Patterns and Best Practices starts with the learning of microservices key concepts and showing how to make the right choices while designing microservices. You will then move onto internal microservices application patterns, such as caching strategy, asynchronism, CQRS and event sourcing, circuit breaker, and bulkheads. As you progress, you'll learn the design patterns of microservices. The book will guide you on where to use the perfect design pattern at the application development stage and how to break monolithic application into microservices. You will also be taken through the best practices and patterns involved while testing, securing, and deploying your microservice application. At the end of the book, you will easily be able to create interoperable microservices, which are testable and prepared for optimum performance. What you will learn How to break monolithic application into microservices Implement caching strategies, CQRS and event sourcing, and circuit breaker patterns Incorporate different microservice design patterns, such as shared data, aggregator, proxy, and chained Utilize consolidate testing patterns such as integration, signature, and monkey tests Secure microservices with JWT, API gateway, and single sign on Deploy microservices with continuous integration or delivery, Blue-Green deployment Who this book is for This book is for architects and senior developers who would like implement microservice design patterns in their enterprise application development. The book assumes some prior programming knowledge.

microservices sequence diagram example: Practical Microservices with Dapr and .NET Davide Bedin, Mark Russinovich, 2022-11-11 Use the innovative, highly portable event-driven

distributed application runtime to simplify building resilient and scalable microservices for cloud and edge applications. Purchase of the print or Kindle book includes a free eBook in the PDF format. Key FeaturesBuild resilient, stateless, and stateful microservice applications that run on the cloud and edgeOvercome common issues in distributed systems, such as low latency and scaling, using any language and frameworkLearn how to expose and operate Dapr applications with multiple optionsBook Description This second edition will help you get to grips with microservice architectures and how to manage application complexities with Dapr in no time. You'll understand how Dapr simplifies development while allowing you to work with multiple languages and platforms. Following a C# sample, you'll understand how Dapr's runtime, building blocks, and software development kits (SDKs) help you to simplify the creation of resilient and portable microservices. Dapr provides an event-driven runtime that supports the essential features you need for building microservices, including service invocation, state management, and publish/subscribe messaging. You'll explore all of those in addition to various other advanced features with this practical guide to learning Dapr. With a focus on deploying the Dapr sample application to an Azure Kubernetes Service cluster and to the Azure Container Apps serverless platform, you'll see how to expose the Dapr application with NGINX, YARP, and Azure API Management. By the end of this book, you'll be able to write microservices easily by implementing industry best practices to solve problems related to distributed systems. What you will learnUse Dapr to create services, invoking them directly and via pub/subDiscover best practices for working with microservice architecturesLeverage the actor model to orchestrate data and behaviorExpose API built with Dapr applications via NGINX and Azure API ManagementUse Azure Kubernetes Service to deploy a sample applicationMonitor Dapr applications using Zipkin, Prometheus, and GrafanaScale and load test Dapr applications on KubernetesGet to grips with Azure Container Apps as you combine Dapr with a serverless platformWho this book is for This book is for developers looking to explore and implement microservices architectures in Dapr applications using .NET examples. Whether you are new to microservices or have knowledge of this architectural approach and want to get hands-on experience using Dapr, you'll find this book useful. Familiarity with .NET will help you to understand the C# samples and code snippets used in the book.

microservices sequence diagram example: Dependable Software Engineering. Theories, Tools, and Applications Xinyu Feng, Markus Müller-Olm, Zijiang Yang, 2018-08-25 This book constitutes the proceedings of the Third International Symposium on Dependable Software Engineering: Theories, Tools, and Applications, SETTA 2018, held in Beijing, China, in September 2018. The 9 full papers presented together with 3 short papers were carefully reviewed and selected from 22 submissions. The purpose of SETTA is to provide an international forum for researchers and practitioners to share cutting-edge advancements and strengthen collaborations in the field of formal methods and its interoperability with software engineering for building reliable, safe, secure, and smart systems.

microservices sequence diagram example: *Building Micro-Frontends* Luca Mezzalira, 2021-11-17 What's the answer to today's increasingly complex web applications? Micro-frontends. Inspired by the microservices model, this approach lets you break interfaces into separate features managed by different teams of developers. With this practical guide, Luca Mezzalira shows software architects, tech leads, and software developers how to build and deliver artifacts atomically rather than use a big bang deployment. You'll learn how micro-frontends enable your team to choose any library or framework. This gives your organization technical flexibility and allows you to hire and retain a broad spectrum of talent. Micro-frontends also support distributed or colocated teams more efficiently. Pick up this book and learn how to get started with this technological breakthrough right away. Explore available frontend development architectures Learn how microservice principles apply to frontend development Understand the four pillars for creating a successful micro-frontend architecture Examine the benefits and pitfalls of existing micro-frontend architectures Learn principles and best practices for creating successful automation strategies Discover patterns for integrating micro-frontend architectures using microservices or a monolith API layer

microservices sequence diagram example: Microservices Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Patricia Lago, Manuel Mazzara, Victor Rivera, Andrey Sadovykh, 2019-12-11 This book describes in contributions by scientists and practitioners the development of scientific concepts, technologies, engineering techniques and tools for a service-based society. The focus is on microservices, i.e cohesive, independent processes deployed in isolation and equipped with dedicated memory persistence tools, which interact via messages. The book is structured in six parts. Part 1 “Opening” analyzes the new (and old) challenges including service design and specification, data integrity, and consistency management and provides the introductory information needed to successfully digest the remaining parts. Part 2 “Migration” discusses the issue of migration from monoliths to microservices and their loosely coupled architecture. Part 3 “Modeling” introduces a catalog and a taxonomy of the most common microservices anti-patterns and identifies common problems. It also explains the concept of RESTful conversations and presents insights from studying and developing two further modeling approaches. Next , Part 4 is dedicated to various aspects of “Development and Deployment”. Part 5 then covers “Applications” of microservices, presenting case studies from Industry 4.0, Netflix, and customized SaaS examples. Eventually, Part 6 focuses on “Education” and reports on experiences made in special programs, both at academic level as a master program course and for practitioners in an industrial training. As only a joint effort between academia and industry can lead to the release of modern paradigm-based programming languages, and subsequently to the deployment of robust and scalable software systems, the book mainly targets researchers in academia and industry who develop tools and applications for microservices.

microservices sequence diagram example: Communicating Process Architectures 2017 & 2018 J. Bækgaard Pedersen, K. Chalmers, J.F. Broenink, 2019-03-26 Concurrent and parallel systems are intrinsic to the technology which underpins almost every aspect of our lives today. This book presents the combined post-proceedings for two important conferences on concurrent and parallel systems: Communicating Process Architectures 2017, held in Sliema, Malta, in August 2017, and Communicating Process Architectures 2018, held in Dresden, Germany, in August 2018. CPA 2017: Fifteen papers were accepted for presentation and publication, they cover topics including mathematical theory, programming languages, design and support tools, verification, and multicore infrastructure and applications ranging from supercomputing to embedded. A workshop on domain-specific concurrency skeletons and the abstracts of eight fringe presentations reporting on new ideas, work in progress or interesting thoughts associated with concurrency are also included in these proceedings. CPA 2018: Eighteen papers were accepted for presentation and publication, they cover topics including mathematical theory, design and programming language and support tools, verification, multicore run-time infrastructure, and applications at all levels from supercomputing to embedded. A workshop on translating CSP-based languages to common programming languages and the abstracts of four fringe presentations on work in progress, new ideas, as well as demonstrations and concerns that certain common practices in concurrency are harmful are also included in these proceedings. The book will be of interest to all those whose work involves concurrent and parallel systems.

microservices sequence diagram example: Solutions Architect's Handbook Saurabh Shrivastava, Neelanjali Srivastav, 2024-03-29 From fundamentals and design patterns to the latest techniques such as generative AI, machine learning and cloud native architecture, gain all you need to be a pro Solutions Architect crafting secure and reliable AWS architecture. Get With Your Book: PDF Copy, AI Assistant, and Next-Gen Reader Free Key Features Hits all the key areas -Rajesh Sheth, VP, Elastic Block Store, AWS Offers the knowledge you need to succeed in the evolving landscape of tech architecture - Luis Lopez Soria, Senior Specialist Solutions Architect, Google A valuable resource for enterprise strategists looking to build resilient applications - Cher Simon, Principal Solutions Architect, AWS Book DescriptionBuild a strong foundation in solution architecture and excel in your career with the Solutions Architect’s Handbook. Authored by seasoned AWS technology leaders Saurabh Shrivastav and Neelanjali Srivastav, this book goes

beyond traditional certification guides, offering in-depth insights and advanced techniques to meet the specific needs and challenges of solutions architects today. This edition introduces exciting new features that keep you at the forefront of this evolving field. From large language models and generative AI to deep learning innovations, these cutting-edge advancements are shaping the future of technology. Key topics such as cloud-native architecture, data engineering architecture, cloud optimization, mainframe modernization, and building cost-efficient, secure architectures remain essential today. This book covers both emerging and foundational technologies, guiding you through solution architecture design with key principles and providing the knowledge you need to succeed as a Solutions Architect. It also sharpens your soft skills, providing career-accelerating techniques to stay ahead. By the end of this book, you will be able to harness cutting-edge technologies, apply practical insights from real-world scenarios, and enhance your solution architecture skills with the Solutions Architect's Handbook.What you will learn Explore various roles of a solutions architect in the enterprise Apply design principles for high-performance, cost-effective solutions Choose the best strategies to secure your architectures and boost availability Develop a DevOps and CloudOps mindset for collaboration, operational efficiency, and streamlined production Apply machine learning, data engineering, LLMs, and generative AI for improved security and performance Modernize legacy systems into cloud-native architectures with proven real-world strategies Master key solutions architect soft skills Who this book is for This book is for software developers, system engineers, DevOps engineers, architects, and team leaders who already work in the IT industry and aspire to become solutions architect professionals. Solutions architects who want to expand their skillset or get a better understanding of new technologies will also learn valuable new skills. To get started, you'll need a good understanding of the real-world software development process and some awareness of cloud technology.

microservices sequence diagram example: Embracing Microservices Design Ovais Mehboob Ahmed Khan, Nabil Siddiqui, Timothy Oleson, Mark Fussell, 2021-10-29 Develop microservice-based enterprise applications with expert guidance to avoid failures and technological debt with the help of real-world examples Key FeaturesImplement the right microservices adoption strategy to transition from monoliths to microservicesExplore real-world use cases that explain anti-patterns and alternative practices in microservices developmentDiscover proven recommendations for avoiding architectural mistakes when designing microservicesBook Description Microservices have been widely adopted for designing distributed enterprise apps that are flexible, robust, and fine-grained into services that are independent of each other. There has been a paradigm shift where organizations are now either building new apps on microservices or transforming existing monolithic apps into microservices-based architecture. This book explores the importance of anti-patterns and the need to address flaws in them with alternative practices and patterns. You'll identify common mistakes caused by a lack of understanding when implementing microservices and cover topics such as organizational readiness to adopt microservices, domain-driven design, and resiliency and scalability of microservices. The book further demonstrates the anti-patterns involved in re-platforming brownfield apps and designing distributed data architecture. You'll also focus on how to avoid communication and deployment pitfalls and understand cross-cutting concerns such as logging, monitoring, and security. Finally, you'll explore testing pitfalls and establish a framework to address isolation, autonomy, and standardization. By the end of this book, you'll have understood critical mistakes to avoid while building microservices and the right practices to adopt early in the product life cycle to ensure the success of a microservices initiative. What you will learnDiscover the responsibilities of different individuals involved in a microservices initiativeAvoid the common mistakes in architecting microservices for scalability and resiliencyUnderstand the importance of domain-driven design when developing microservicesIdentify the common pitfalls involved in migrating monolithic applications to microservicesExplore communication strategies, along with their potential drawbacks and alternativesDiscover the importance of adopting governance, security, and monitoringUnderstand the role of CI/CD and testingWho this book is for This practical microservices book is for software

architects, solution architects, and developers involved in designing microservices architecture and its development, who want to gain insights into avoiding pitfalls and drawbacks in distributed applications, and save time and money that might otherwise get wasted if microservices designs fail. Working knowledge of microservices is assumed to get the most out of this book.

Related to microservices sequence diagram example

What are microservices? The microservices pattern language is your guide when designing an architecture: service collaboration, testing, deployment, common crosscutting concerns and more
Microservices Pattern: Microservice Architecture pattern I appreciate how you highlighted the benefits of microservices, such as increased scalability and faster development cycles, while also addressing the potential challenges involved. Your
A pattern language for microservices Hi there! I'm looking to learn how to build multi-tenant microservices. Can you recommend some resources?
Pattern: Transactional outbox - Microservices Using this pattern and thinking in microservices, once each service has its own database, should I have an Outbox table in each service database or should I have a common (centralized)
Pattern: Saga - Microservices In a portfolio of hundreds of microservices with dozens of key biz transactions each with 3-5 participating services, how to make sure sagas are implemented properly across all service
Pattern: API Gateway / Backends for Frontends - Microservices The granularity of APIs provided by microservices is often different than what a client needs. Microservices typically provide fine-grained APIs, which means that clients need to interact
Pattern: Event sourcing - Microservices Related to microservices, and depending on the case, you can "concentrate" the SAGA in one service, or the SAGA process (its steps) can be distributed in different services that react to the
Pattern: Shared database - Microservices Several microservices maintaining connection pools to the same database creates contention. Multiple independent processes accessing the database via some ORM layer can cause hard
Pattern: Messaging - Microservices The Domain-specific protocol pattern is an alternative pattern The RPI pattern is an alternative pattern See also My book Microservices patterns describes inter-communication in depth
Essential characteristics of the microservice architecture: loosely Avoid the pitfalls of adopting microservices and learn essential topics, such as service decomposition and design and how to refactor a monolith to microservices
What are microservices? The microservices pattern language is your guide when designing an architecture: service collaboration, testing, deployment, common crosscutting concerns and more
Microservices Pattern: Microservice Architecture pattern I appreciate how you highlighted the benefits of microservices, such as increased scalability and faster development cycles, while also addressing the potential challenges involved. Your
A pattern language for microservices Hi there! I'm looking to learn how to build multi-tenant microservices. Can you recommend some resources?
Pattern: Transactional outbox - Microservices Using this pattern and thinking in microservices, once each service has its own database, should I have an Outbox table in each service database or should I have a common (centralized)
Pattern: Saga - Microservices In a portfolio of hundreds of microservices with dozens of key biz transactions each with 3-5 participating services, how to make sure sagas are implemented properly across all service
Pattern: API Gateway / Backends for Frontends - Microservices The granularity of APIs provided by microservices is often different than what a client needs. Microservices typically provide fine-grained APIs, which means that clients need to interact
Pattern: Event sourcing - Microservices Related to microservices, and depending on the case,

you can "concentrate" the SAGA in one service, or the SAGA process (its steps) can be distributed in different services that react to

Pattern: Shared database - Microservices Several microservices maintaining connection pools to the same database creates contention. Multiple independent processes accessing the database via some ORM layer can cause hard

Pattern: Messaging - Microservices The Domain-specific protocol pattern is an alternative pattern The RPI pattern is an alternative pattern See also My book Microservices patterns describes inter-communication in depth

Essential characteristics of the microservice architecture: loosely Avoid the pitfalls of adopting microservices and learn essential topics, such as service decomposition and design and how to refactor a monolith to microservices

Back to Home: https://old.rga.ca