



MICROSERVICE DESIGN

A GUIDE FOR EXECUTIVES

Whitepaper By

Pegasus One
1440 N Harbor Blvd #900
(714) 485-8104
info@pegasusone.com

CONTENTS

- 01 **PREFACE**
MICROSERVICE DESIGN
- 02 **WHAT ARE MICROSERVICES**
FEATURES AND BENEFITS
- 04 **DESIGNING MICROSERVICE ARCHITECTURES**
THINGS TO DESIGN FOR
- 05 **BEST PRACTICES FOR DESIGN**
5 BEST PRACTICES TO KEEP IN MIND WHILE
DESIGNING MICROSERVICES
- 10 **IMPLEMENTATION TYPES**
6 PRIMARY IMPLEMENTATION TYPES, AND
THEIR KEY CHARACTERISTICS
- 17 **BOOST YOUR OPERATIONAL EFFICIENCY**
HOW WE CAN HELP YOU BUILD YOUR
MICROSERVICE ARCHITECTURE
- 18 **TALK TO US**

MICROSERVICE DESIGN

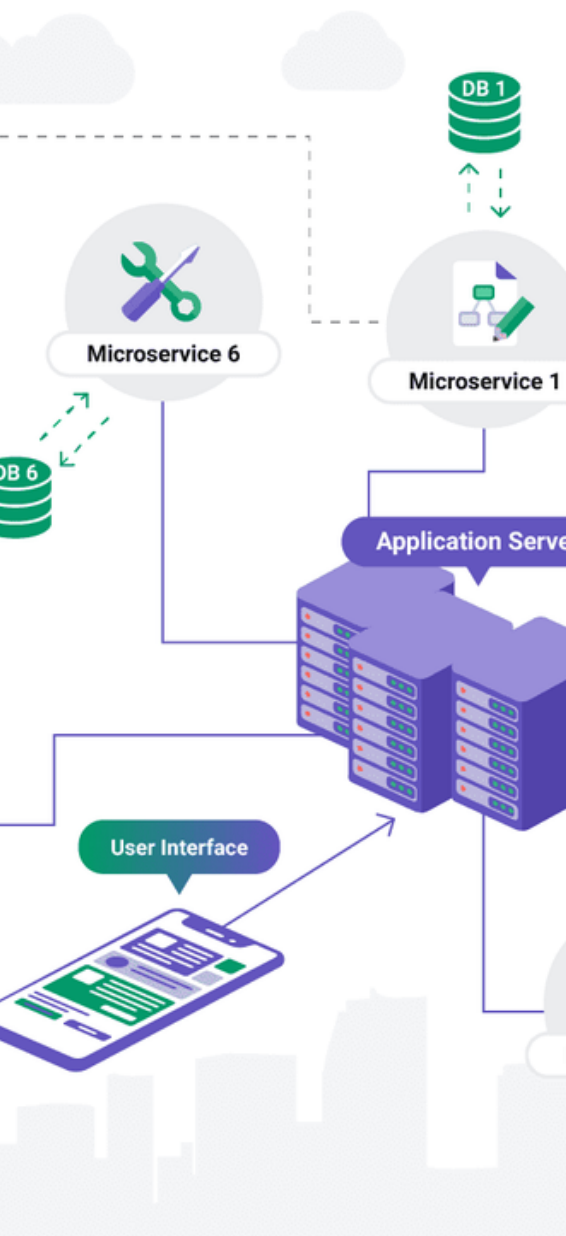


PREFACE

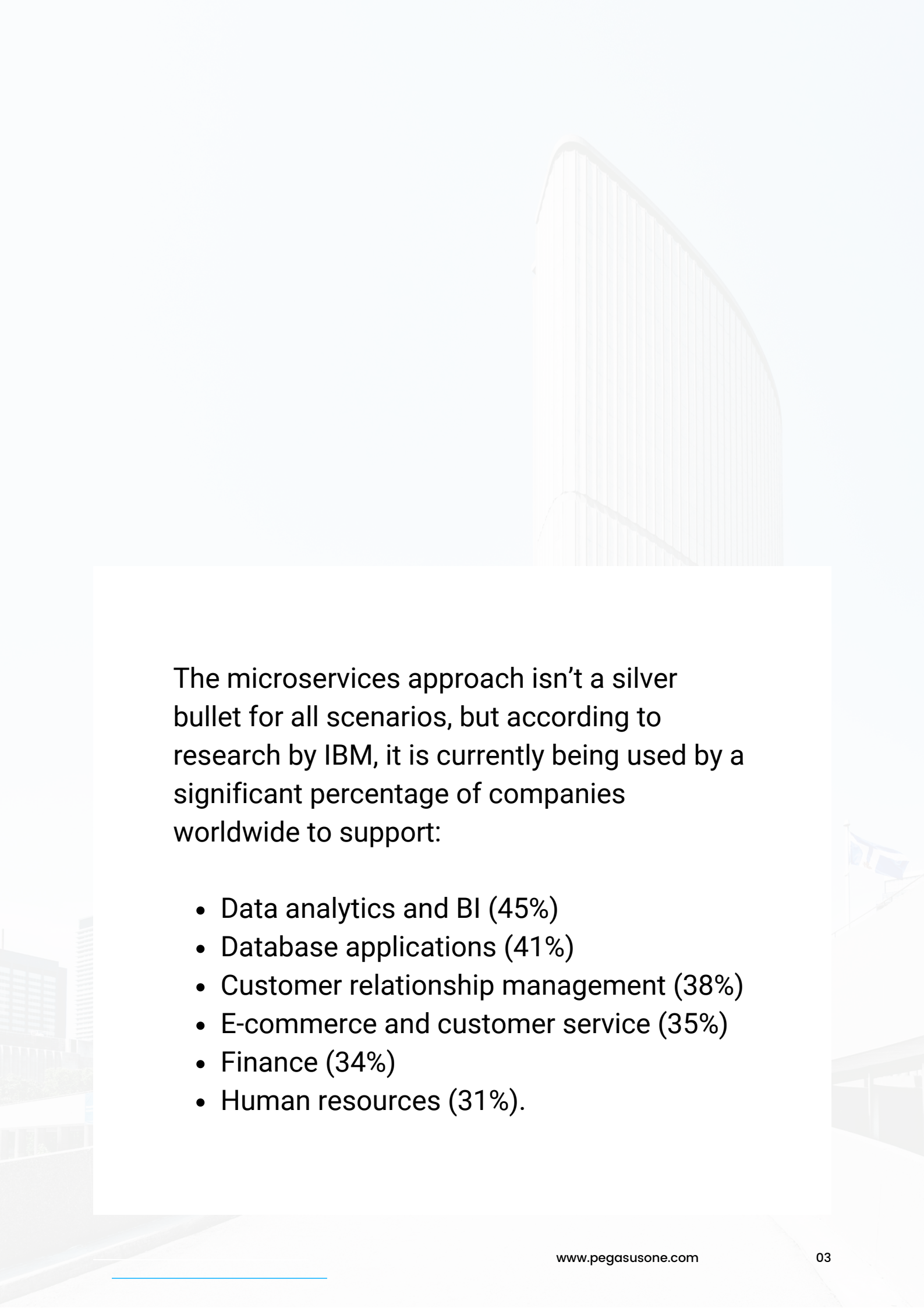
In today's business landscape, the competitive advantage often comes down to how quickly you can deliver on your initiatives, and how easily you adapt to changes and opportunities. Utilizing microservices may be the solution to scale up your delivery and improve your operational agility. In this white paper, we'll take a deep dive into microservices, providing insight into how they are designed, defining the six primary microservice architecture implementation types, and explaining the key characteristics of each one.

WHAT ARE MICROSERVICES

Microservices (or microservice architecture) is an architectural style that structures applications as a collection of modular services. These services are much easier to maintain and offer other features and benefits, including:



- **Loose coupling** to avoid cascading impacts to other services and systems when a microservice is changed
- **Independent deployment** to circumvent testing and deployment conflicts that can lead to application failure
- **Business capability** focus to enable microservices to deliver on strategic outcomes
- **Small team** ownership to support autonomy and loose coupling and reduce complexity and overhead
- **Scalability** to operate more efficiently, perform better when handling multiple tasks and requests, and be able to scale as needs evolve
- **Fault tolerance** to enable a service to continue operating in cases of component failure
- **Availability** with all hosts mapping to the same storage to provide redundancy and deliver uninterrupted services



The microservices approach isn't a silver bullet for all scenarios, but according to research by IBM, it is currently being used by a significant percentage of companies worldwide to support:

- Data analytics and BI (45%)
- Database applications (41%)
- Customer relationship management (38%)
- E-commerce and customer service (35%)
- Finance (34%)
- Human resources (31%).

DESIGNING MICROSERVICE ARCHITECTURES

To get the most from a migration to microservice architecture, you must design for:

- **Independent and autonomous service**
- **Scalability**
- **Decentralization**
- **Resilience**
- **Real-time load balancing**
- **High availability**
- **Continuous delivery**
- **API integration**
- **Isolation from failures**
- **Auto-provisioning**

TOP 5 BEST PRACTICES FOR MICROSERVICE DESIGN

GIVE EACH MICROSERVICE A SEPARATE DATA STORE

Avoid using the same backend datastore across your microservices so that each team can select the database that's the best fit. While it may seem more efficient to simply create and share a single database amongst microservices, it can create problems. If a team has a need to update the database structure they're using, all the other microservices attached to that database will have to be changed as well.

1

TOP 5 BEST PRACTICES FOR MICROSERVICE DESIGN

CREATE A NEW MICROSERVICE WHEN ADDING OR CHANGING CODE

Adding or re-writing code in a deployed microservice? Avoid editing the code in the existing microservice and instead create a new microservice so the stability of the deployed service is not at risk. Build, deploy, and test the new microservice until it's stable, and then merge it with the original microservice.

2

TOP 5 BEST PRACTICES FOR MICROSERVICE DESIGN

BUILD EACH MICROSERVICE SEPARATELY

Build out each microservice separately, enabling it to access repository files that are service-appropriate, making introducing a new service easier. In some cases, multiple microservices may access similar files of differing revision levels. This may require those administering the services to verify their use before decommissioning old files during a clean-up.

TOP 5 BEST PRACTICES FOR MICROSERVICE DESIGN

USE CONTAINERS FOR DEPLOYMENT

Using containers like Docker makes the deployment process much easier, enabling teams to use a single tool to deploy every service within a given container. Once a service is placed into a container, the tool intuitively knows how to deploy it.

TOP 5 BEST PRACTICES FOR MICROSERVICE DESIGN

REGARD SERVERS AS STATELESS

Because many servers (especially those running customer-facing code) perform the same core functions, they can be treated as interchangeable. Running systems in which individual servers perform unique, specialized tasks lacks efficiency and prohibits essential backup. With interchangeable servers, a server automatically takes over if another server should fail.

5

IMPLEMENTATION TYPES

Now we'll look at the 6 primary implementation types and their key characteristics. These include fine-grained SOA, layered APIs over fine-grained SOA, message-oriented state management over layered APIs, event-driven state management over layered APIs, isolating state in layered APIs, and replicating state in layered APIs.

- **Fine-grained services-oriented architecture (SOA)**
- **Layered APIs over fine-grained SOA**
- **Message-oriented state management over layered APIs**
- **Event-driven state management over layered APIs**
- **Isolating state in layered APIs**
- **Replicating state in layered APIs (event sourcing)**

FINE-GRAINED SOA

WHAT IT IS

Architecture focused on a single purpose, with services that support a common application while functioning independently of each other.

CHARACTERISTICS

- Used for larger, more modular services
- Divides infrastructure into granular pieces
- Services provide connectivity to external systems
- Tight dependencies reduce the speed of change

LAYERED APIS OVER FINE-GRAINED SOA

WHAT IT IS

Levels up fine-grained SOA to expose applications, using System APIS, Process APIs, and Experience APIS to connect data to them.

CHARACTERISTICS

- Creates structure within the architecture
- Provides easier insight into the purpose of individual microservices
- Enables easier management

MESSAGE-ORIENTED STATE MANAGEMENT OVER LAYERED APIS

WHAT IT IS

Replicates the state of business data between microservices or data stores.

CHARACTERISTICS

- Uses a message queue to deliver consistent external views
- Converges events
- Transmits states to disparate locations
- Queries states through other microservices

EVENT-DRIVEN STATE MANAGEMENT OVER LAYERED APIS

WHAT IT IS

Utilizes standards-enforced queues to control time-stamped actions (events) passing over the queue.

CHARACTERISTICS

- Provides real-time data updates
- Used for fraud detection, workflow notifications, news feeds, etc.

ISOLATING STATE IN LAYERED APIS

WHAT IT IS

An alternative approach to event-driven microservices.

CHARACTERISTICS

- Adds persistence to each microservice
- Provides consistency at the time of query rather than within the interchange
- Allows each microservice to contain its own state

REPLICATING STATE IN LAYERED APIS (EVENT SOURCING)

WHAT IT IS

Offers a single place for storage of state mutations that enable isolated microservices to rebuild internal states.

CHARACTERISTICS

- Consistent design
- Reduces risk of failure in microservices
- Improves speed-of-change
- Delivers faster time-to-value



Need to boost your operational efficiency?

Find out what microservices can do for you. At Pegasus One, our expert teams have helped companies just like yours restructure their applications. With a microservices infrastructure, you can scale applications up, ensure fault tolerance and provide high availability. Contact us for a free consultation to learn more about how to apply microservices to your architecture.

TALK TO US

Call us at (714) 485-8104

Write to us at info@pegasusone.com

Or visit us online at: www.pegasusone.com

1440 N HARBOR BLVD #900, FULLERTON, CA

