MICRO SERVICES

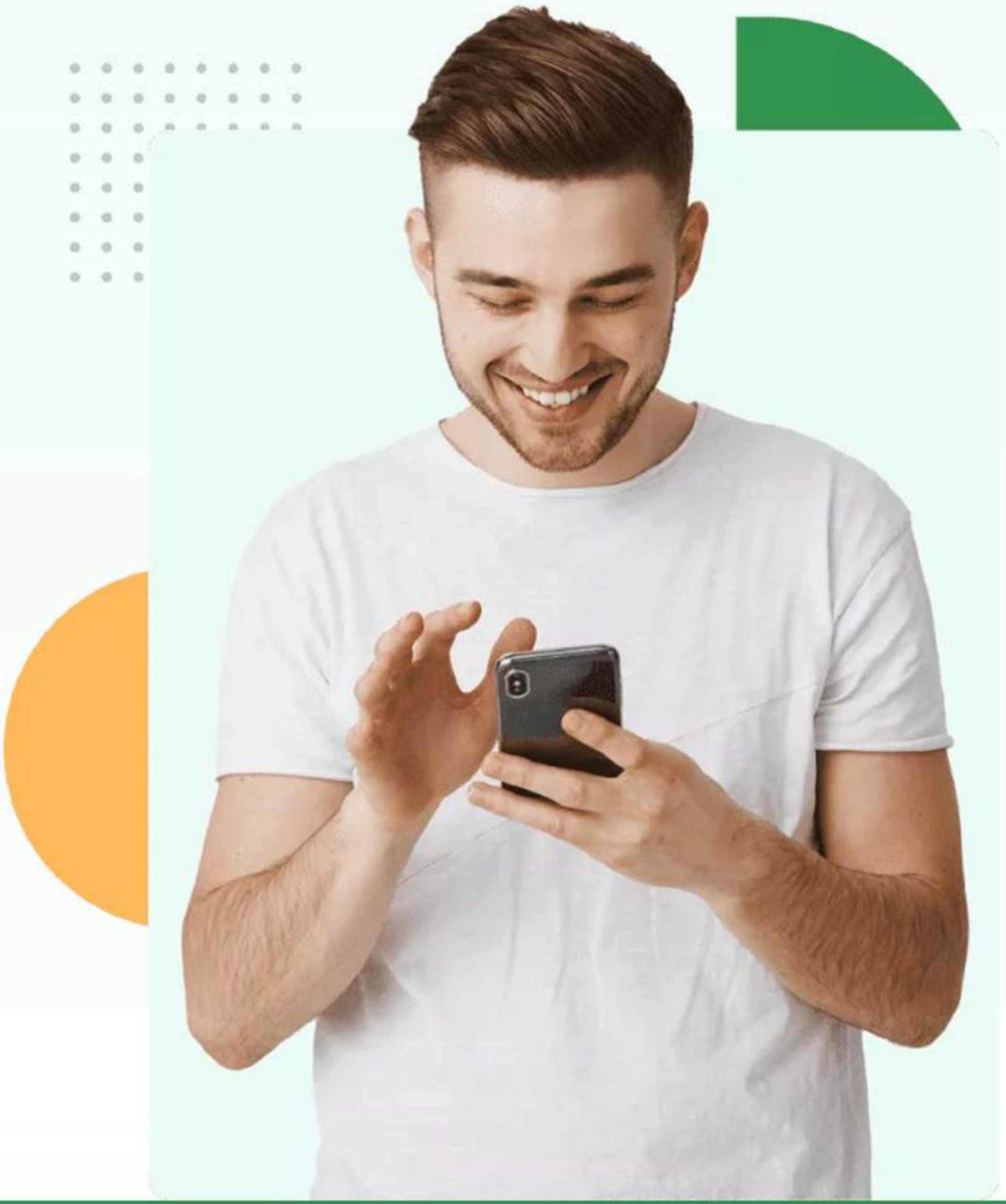


Table of Content

Overview 01

Types of Microservices 02

How does Microservices Architecture work 03

Microservices Advantage 04

Microservices Best Practices 05

Conclusion 06

References 07

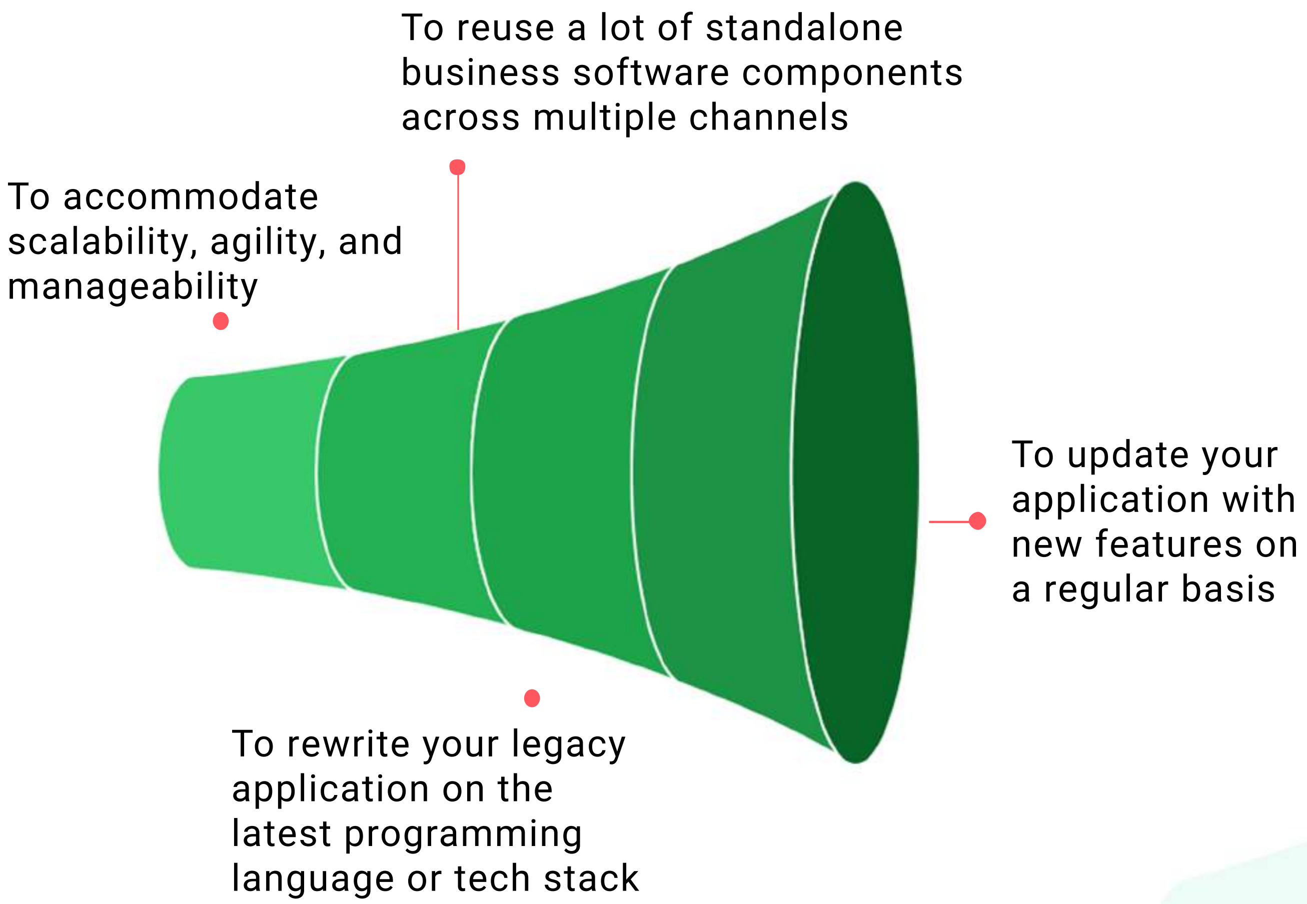
lendfoundry

Overview

Microservices-based architectures offer greater agility and help growing organizations accelerate innovation in their digital experiences. This approach provides immense flexibility and serves as a business-critical component to omnichannel applications through API integrations. With time, an enterprise application becomes complex, demands scalability, and needs high responsiveness, microservices help organizations quickly fulfill their growing business needs. Even though the microservices approach involves more complexity and monitoring systems, the benefits you can achieve are worth the effort.

According to a recent survey by Statista, 85 percent of respondents from large organizations (5,000+ employees) said they are currently using microservices.

When to use Microservice Architecture?


To reuse a lot of standalone business software components across multiple channels
To accommodate scalability, agility, and manageability
To update your application with new features on a regular basis
To rewrite your legacy application on the latest programming language or tech stack

Types of Microservices

Microservices can be broadly categorized into two “stateless” and “stateful”.

Stateful microservices

Such type of microservices possesses saved data in a database and is a good candidate as the building blocks of a distributed system. Well-behaved stateful microservices don’t tend to share databases with other microservices to make decoupling seamless and offer well-defined interfaces. Whenever a stateful service terminates it has to save its state.

Stateless microservices

Such type of microservices doesn’t save anything. Stateless microservices take in a request, process it, and send a response back without persisting any state information. Once the request is completed they forget it, which implies stateless microservices don’t keep any handy permanent notes to remind them where they got to. Whenever a stateless service terminates it has nothing to save.

Since microservices allow a large application to be separated into smaller independent parts, with each part having its own realm of responsibility, microservices architecture can be further categorized into four main types:

Single-Tier Microservice Architecture

The single-tier microservice architecture is used mainly to develop, deploy, and manage a distributed system at ease. In this architecture, a collection of individual services work together to provide a complete solution and are typically written in a specific language or framework. They are designed to be independent and self-sufficient in nature.

Two-Tier Microservice Architecture

The two-tier architecture divides each microservice into two parts i.e. a front-end and a back-end, which enables the front-end microservice to handle the user interaction and the back-end microservice to handle the business logic. Even though this segregation makes the two-tier architecture easier to scale and operate, it makes it harder to debug a problem in the back-end microservice, as the back-end microservice remains isolated from the rest.

Three-Tier Microservice Architecture

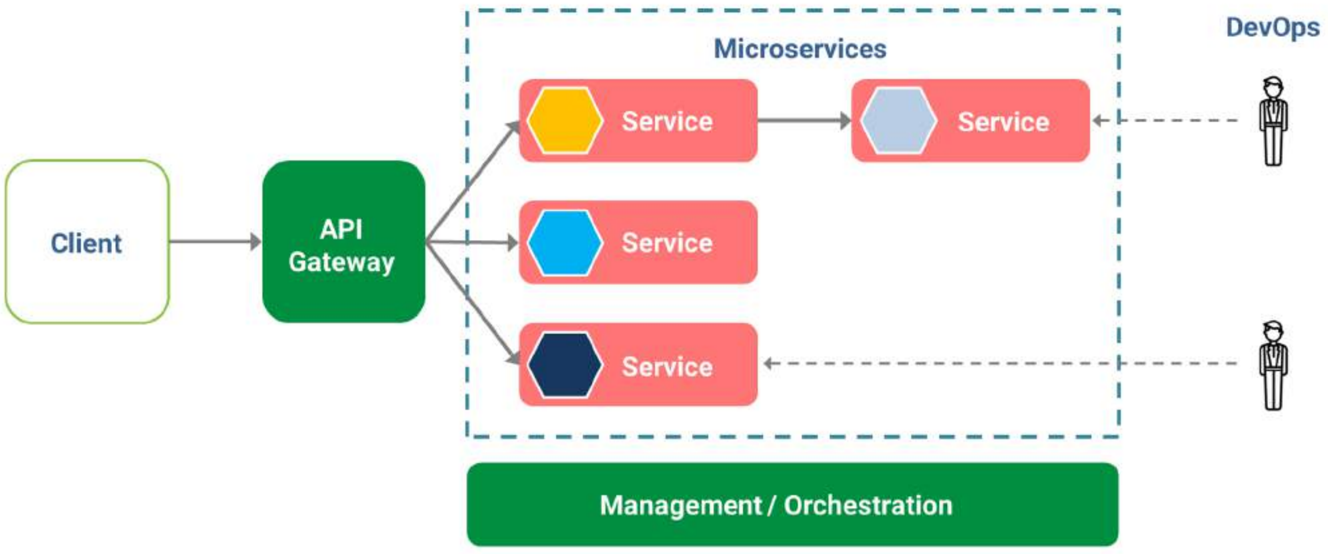
Such types of microservice architectures are used to organize distributed systems. In the typical three-tier architecture, you will find a client, a middleware layer, and a back-end or application layer. The client uses the middleware layer to access the back-end or application layer and the middleware layer provides a common interface to the back-end or application layer to manage communication between

Microservice Architecture with a Cloud-Based Back-End

The best advantage of using this microservice architecture is that the application can be easily deployed and managed on a cloud-based platform, such as Amazon Web Services (AWS) which helps the application to scale up or down during the deployment of new features or changes. Apart from this, microservice architecture with a cloud-based back-end also improves the accessibility on a wide range of devices enabling the application to be used by a large number of users, regardless of their location.

How does Microservices Architecture work?

The microservices architecture focuses on classifying bulky applications. Each one of the microservice addresses an application’s single concern, such as a data search, logging function, or web service function. All these individual microservices come together to form one efficient application. This intuitive, functional division of an application provides an even greater ability to quickly update the code of a single function, without refactoring or even redeploying the rest of the microservices architecture, in an overall service or larger end-user application. This makes failure points more independent of each other and helps in creating a more stable overall application architecture.



Using microservices architecture also generates an opportunity for other microservices to become self-healing. With orchestration tools like Kubernetes, self-healing can occur without human intervention automatically and is transparent to the end user as well.

Microservices Advantage

Even though microservices are decentralized and run on different servers still they work together for an application. Each microservice serves a single function and enables simple routing between services with API communication. Here are some of the other benefits:

Accelerated Development

The Microservices approach offers your team the ability to move faster when it comes to adding new features by breaking major application functionalities into independent components. The approach helps developers to quickly adapt to new technologies,
user complaints, and market changes.

Improves scalability

Scaling an application for millions of customer bases is impossible and massively expensive if you are using a monolithic setup. Microservices are an efficient use of resources and it leads to lower costs of running each instance of application development.

Enables faster Knowledge Transfer

The Microservices approach enables your existing and new talents to learn a single system for business growth. Over time new hires can continue contributing to the entire application up-gradation without any hurdle.

Offers Reusability

Microservices applications are self-contained and you can do whatever you want with them without affecting the rest of the system. Using microservices allows developers to create new capabilities without writing code from the scratch.

Easier Maintenance

Managing a microservices application is actually easier than a monolithic one. If you're dealing with thousands of discrete components and transactions, the microservices approach often has better tools and methodologies that handle a large number of components effectively.

Improves Quality

The Microservices approach impacts application development massively and enables businesses to enhance an application's real-time performance and availability.

lendfoundry

Microservices Best Practices

Based on our extensive experience with microservices-based digital lending technology, we have listed the 9 best practices and hands-on-approach that will ensure a secured and scalable FinTech ecosystem with microservices for your business.

Enhance efficiency with Domain-Driven Design (DDD):

Using a DDD-oriented approach for microservices helps with loosely coupled services and enables consistent high-level functionality. The strategic phase of the Domain-Driven Design model will ensure design architecture that can encapsulate business capabilities and its tactical phase, on the other, will allow the creation of a domain model using different design patterns for your business.

Make use of the Single Responsibility Principle (SRP):

Using SRP as a microservice design principle for FinTechs is healthy because if something goes wrong, only one program's functionality will take the hit, not the entire application, which is unfortunately true for the monolith.

Facilitate service autonomy with independent microservices:

In case you are planning to take the service isolation a step further then you should facilitate independent microservices. With this approach, a microservice can be deployed and scaled as needed because they work together and communicate through well-defined APIs or similar mechanisms that don't expose the internal workings of the microservices.

Use asynchronous communications between services:

Being a non-blocking communication protocol, asynchronous communication follows event-driven architecture which reduces the coupling between services during the execution of user requests and provides better resilience between microservices.

Ensure Distributed database for microservices:

Microservices are loosely coupled but still, they retrieve data from the same data store with a shared database, and thus to deal with multiple data queries, latency issues, improves security and resilience; FinTechs must use distributed databases for microservices.

Containerize microservices for scalable and distributed systems:

With containerized microservices, FinTechs can create a massively scalable and distributed system to avoid the bottlenecks of a central database. They also facilitate rapid rollouts & rollbacks and continuous integration & continuous delivery (CI/CD) pipelines for applications and

Implement microservices security best practices:

Since microservices communicate with external platforms or services, it’s crucial for FinTechs to tackle security issues and implement microservices security best practices by adapting to the DevOps model which can ensure the security of your entire microservices framework.

Use immutable APIs for simplified parallel programming:

Using microservice architecture makes parallel programming much easier and secured with immutable containers. Immutable APIs enable you to execute multiple threads in parallel and improve the efficiency of programming.

Adopt a DevOps culture to boost delivery speeds:

Adopting a DevOps culture for your organization will enable a cohesive strategy, efficient collaboration, increased agility, scalability, and flexibility for both development and operations.

Conclusion

Using microservices-based digital lending technology for your business can bring various benefits like speedier deployment and scalability, reduced downtime, and overall improvement. Following the microservices best practices will not only help your business with a seamless transition from monolithic to microservice-based applications but will also help in:

1. planning & organizing whether the microservices architecture is a good fit based on your requirements.
2. designing your services to be loosely coupled, have high cohesion, and cover a single bounded context.
3. developing an environment that enables developers to adapt the framework and get started quickly.
4. data storage & management of each microservice.
5. deploying & hosting your microservices with containerization and DevOps model.

lendfoundry

References

https://www.statista.com/statistics/1236823/microservices-usage-per-organization-size/

https://www.emizentech.com/blog/microservices-architecture.html

https://middleware.io/blog/microservices-architecture/

https://aws.amazon.com/microservices/

https://www.tetrain.com/component/blogfactory/post/43/what-are-the-types-of-microservice-architecture.html