Going Micro to Solve Macro Challenges

Mauricio Bonifacino Cooke

July 12, 2023

Microservices Architecture | 6 mins

Leveraging Microservices-Based Architecture to Meet the Challenges of Modernizing Payments Platforms

Complete microservices-based architecture has several key characteristics that are fundamental to resolving payment platforms challenges, like migration, scope flexibility, coexistence with legacy solutions, and migration risk mitigation. **But**...all software architecture paradigms have a dark side, and microservices are no exception.

Microservices architecture solutions can provide unique adaptability, flexibility, and integration characteristics. The challenge is taking advantage of them without getting into the "microservices trap." The right microservices design principles are the key to a successful solution. For a Payments Platform, what does that mean?

Navigating the Future Landscape of Payment Platforms

Payment Platforms need to evolve to support the adoption of new payment methods, integrate with new players, and provide flexible features that customers expect. Legacy monolithic solutions are hindering the expected industry evolution, making it imperative that modernization initiatives become a priority on the agenda.

Legacy

Successful modernization initiatives should include a phases-based migration approach and adoption, empowering the legacy solution while, at the same time, mitigating risks by avoiding a traditional rip-and-replace

microservices to include in each phase and facilitate the natural adoption evolution, finishing with the full platform replacement, step by step, phase by phase.

This approach focuses on empowering legacy solutions during the initial phase with the benefits of the new microservices platform; coexistence is the key to ensuring the best adoption process. The choice is not about replacing legacy with crazy replacement projects but about coexistence, a phased approach, and mitigating migration risk.

Avoiding the Microservices Adoption Trap:

But (there always seems to be a "but") all software architecture paradigms have advantages and disadvantages. Assuming that microservices are better than monolithic systems is a big mistake. The key to a successful solution architecture is to consider both **Solution requirements** and the optimal architecture **design**.

The Importance of Situation-Specific Design:

Each solution has specific needs. For example, consider the case of Amazon Prime: They implemented a comprehensive microservices solution with a high level of granularity, but when transactions began to scale, they faced problems being unable to keep up with the demand. The internal orchestration and communication overhead between microservices consumed a whopping amount of processing resources, making the overall

platform extremely expensive. Defying conventional trends, Amazon quickly rearchitected the solution, and reduced the required orchestration and inter-microservices communications by creating a “bigger-monolithic” process, and by doing that, improved overall performance while obtaining a 90% reduction in processing cost. Link source: https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

What Amazon experienced is not an exception; others are shifting back to monolithic architecture, which makes sense in some specific cases: for instance, when you have mostly closed solutions with no heavy integration requirements, big data volumes, and real-time processing features. In such a case, there may be better ideas than a highly segmented microservices architecture, and, as Amazon Prime did, keeping the best of a monolithic system is the right solution.

Building Robustness with the Right Architectural Design:

If the situation indicates the need for a Microservices architecture, like in the case of Payment Platforms, where phase adoption migrations and conditions for completely open and flexible integrations exist, what should one consider for sound microservices architecture design?

The answer is clear: A microservices-based should respect the correct design patterns, with proper design practices, and an overall critical view of complexity vs. simplicity. Easy to say, not that easy to do it. 😊 But these elements will preserve the right balance and ensure that the solution brings true microservices advantages.

To strike a balance between complexity and simplicity in microservice design, follow some general design principles to guide your decisions and help you avoid pitfalls. Single responsibility is one such principle. It says each microservice should have a clear responsibility corresponding to a business capability or subdomain. This will help keep the microservice simple, focused, and easy to understand and maintain. Another principle is loose coupling, which says each microservice should interact with other

microservices through well-defined interfaces, avoiding direct dependencies or shared states.

We should be cautious and avoid just following the architectural trend of the moment. There is a cyclical process in the architecture model trends. Still, the important points are

Payment platforms are a perfect example where the situation presents a clear set of needs that specifies that a full microservices solution is the best approach…but properly done.
1: Use the appropriate architecture paradigm for each solution situation needs, and
2: implement it correctly, with special care in terms of complexity vs. simplicity and good design patterns and practices

Payment platforms are a perfect example where the situation presents a clear set of needs that specifies that a full microservices solution is the best approach…but properly done.

We created the Ren platform to be 100% microservices-based, following best-in-class design patterns, practices, and the right level of complexity vs simplicity; the right balance is the key.
We approach most of our projects with phased adoption strategies, taking full advantage of the integration flexibility and risk mitigation that this approach brings to our customers.

Upgrade to Next-Gen Payments Technology

The rapid evolution of payment technology demands forward-thinking strategies. Our guide offers detailed insights to help you modernize your systems and keep pace with industry advancements. Prepare for the future—download your guide today!

Read More About:

Card Issuing
Switching
ATM Management
Payment Platform
Dynamic Currency Conversion
Payment Hub
Real-Time Payments

Comments

First name

Last name

Email

Comments

protected by reCAPTCHA

Submit

Mauricio Bonifacino Cooke

Contact me if you need help to modernize, evolve and an overall #RENasence of your #PaymentPlatforms within your acquirer, issuer, processor, payment facilitator, fintech, retail and other payments organizations. I'm an information technology senior executive with 25+ years' experience in the Payments industry and related technologies. With solid experience

in leading global software engineering projects and teams, lead the development of top global processing solutions, delivering payment solutions including switching, processing, and last mile software &hardware for customers across Latin America, Western Europe and Middle East

© Copyright Euronet 2026

Euronet, Euronet Worldwide, Ren, epay, Dandelion Payments, Ria Money Transfer, XE.com are all Euronet product/solution names are trademarks or registered trademarks of Euronet, Inc., or one of its subsidiaries, in the United States, other countries or both.

Other parties' trademarks referenced are the property of their respective owners.

Terms and Services

Privacy Policy

Powered by HubSnacks