

Composable commerce is all the rage, promising flexibility, faster innovation, and better alignment between business units and technology teams. But amidst the buzz, one question keeps coming up in meetings and vendor calls: “Do we really need microservices everywhere for composable commerce to work?”

As someone who’s led replatforms and headless rollouts for mid-market and enterprise retail teams, I’ve seen this question morph from theoretical to painfully real. In my experience working alongside firms like **Netguru**, **DEPT**, and **Codal**, the key isn’t blindly chasing a microservices architecture but adopting a selective, controlled modular approach that balances cost, ownership, and long-term replaceability.

Microservices Ecommerce: The Promise vs. The Reality

fingerlakes1.com

Microservices are often touted as the holy grail for ecommerce scalability and flexibility. The idea is straightforward: break down the monolith into independently deployable, replaceable services that communicate via APIs. This *API-first architecture* promises developers agility and the business rapid response to market demands.

However, the reality is often messier. Deploying microservices everywhere can dramatically increase complexity, operational overhead, and cost. It’s not uncommon to discover hidden costs — from maintaining numerous integration points to managing distributed data consistency — months or even quarters after launch. Companies like **Netguru**, which focus on efficient software delivery, emphasize that not all components require the same modularity level.

Why Does Everyone Assume Microservices Everywhere?

- **Marketing hype:** Vendors and agencies like **DEPT** often promise microservices as a magic bullet without clarifying trade-offs.
- **Misconception of ease:** The ‘we can do anything’ attitude glosses over complexity, leading to under-scoped projects.
- **Technical fascination:** Engineering teams are excited by cutting-edge stack diagrams but may ignore operational realities.

The key is to recognize where microservices add value and where they introduce unnecessary friction.

Selective Modular Adoption: Control Costs & Complexity

Composable commerce doesn’t mandate all-or-nothing microservices. Instead, it thrives on **modular scope discipline**. This means carefully selecting which parts of your ecommerce platform should be modularized as microservices and which can remain as integrated components or operate within a headless storefront.

Both **Codal** and **Netguru** advocate for a selective approach:

- Identify high-change areas—like promotions, product recommendations, or checkout flows—that benefit most from independent evolution.
- Keep stable, core domains—the product catalog, user authentication, and inventory—in less granular modules or bounded contexts.
- Use **API-driven integrations** to connect services, ensuring clear interfaces rather than tightly coupled components.

This strategy reduces the multiplication of integration points and distributed data management challenges, which are significant contributors to hidden costs I track religiously post-launch.

Benefits of Modular Scope Discipline

Focus Area Benefit Cost control Prevents unnecessary microservice sprawl and keeps operational overhead manageable Simplicity Reduces points of failure and debugging complexity Faster delivery Enables targeted rollouts and quicker iteration on high-value modules

Long-Term Ownership vs. One-Off Delivery

One critical lens I bring to every new architecture discussion is the question: *“Who owns this in year two?”* Too often, projects are scoped as one-off deliveries that push complexity downstream. The microservices pattern particularly demands sustained engineering rigor, deployment automation, monitoring, and support.

DEPT has emphasized the importance of this mindset, especially in enterprise settings. Modular services only pay off if an organization can maintain clear ownership boundaries and invest in ongoing monitoring.

Without long-term ownership, microservices become a tangled web of brittle APIs and accumulating tech debt rather than a dynamic ecosystem of replaceable components.



Questions to Ask Before Committing to Microservices

1. Does the team have the capacity for ongoing maintenance and monitoring?
2. Are there clear boundaries and responsibilities documented for each service?
3. Is there a platform or tooling strategy that supports rapid deployment and rollback?
4. Have we considered a phased evolution, starting with a hybrid monolith-modular architecture?

Clear System Boundaries and Replaceability

Composable commerce hinges on well-defined system boundaries—sometimes called bounded contexts—and the ability to replace modules independently without disrupting the entire ecosystem. This is where an **API-first stack** becomes indispensable.

API-first doesn't just mean having APIs; it means designing the APIs as primary contracts that enable controlled evolution. The API is the boundary guardrail that supports:

- Replacing legacy or underperforming services with minimal consumer impact
- Facilitating multi-vendor strategies where teams like **Codal** can deliver specific modules independent of core platform teams
- Consistent documentation, security, and versioning to prevent fragmentation

The headless storefront is often built atop these APIs, decoupling presentation from business logic and service orchestration. This separation improves developer velocity and customer experience agility but only works when boundaries are clear.

API-First Architecture & Controlled Evolution

To sum it up, the true power of composable commerce is unlocked through **API-first architecture and controlled evolution**, not reckless microservices proliferation. This means:

- Planning integration points upfront with defined SLAs and fallback mechanisms
- Ensuring interoperability standards across modules from multiple vendors
- Incrementally evolving the architecture with the ability to sunset or replace individual services
- Using headless storefronts as a flexible layer that composes these API-driven modules for optimal customer experiences

Finding the balance here is hard but necessary. Vendors from consulting heavyweights like **DEPT** to development-focused firms like **Netguru** and **Codal** succeed when their clients avoid the temptation of “microservices everywhere” and instead focus on measurable business value through selective modular adoption.



Final Thoughts: Practical Recommendations

If you're tasked with evaluating or implementing composable commerce, here's my blunt advice:

- **Don't buy into 'microservices everywhere' hype.** Start with your highest impact, highest change areas, not the entire catalog or checkout flows simultaneously.
- **Focus on API-first design from day one.** Your future replaceability depends on it.
- **Establish clear ownership.** If you're not asking "Who owns this in year two?" in every vendor meeting, you're setting yourself up for surprise tech debt.
- **Integrate headless storefronts wisely.** They're amazing for composability but don't solve backend complexity or data consistency problems.
- **Track hidden costs continuously.** After launch, keep a running list and communicate trade-offs openly with business stakeholders.

Composable commerce's success isn't measured by how many microservices you deploy but by how well you leverage modular components to deliver adaptable, scalable, and maintainable ecommerce solutions. Selectivity, discipline, and long-term ownership are your best friends in this journey.