Vue Storefront

MACH APPROACH TO ECOMMERCE

Microservices based, **A**PI-first, **C**loud-native SaaS and **H**eadless

powered by MACH ALLIANCE

We are on the verge of a significant transformation, as MACH is becoming a new paradigm for building future-proof software.

The "MACH" standing for Microservices-based, API-first, Cloud-native, and Headless was inspired by the urgent need to provide IT solutions that will be an answer for customer expectations that are changing at breakneck speed.

The rapid dynamic is visible in eCommerce in particular, where MACH-based solutions can be adopted relatively quickly, bringing measurable benefits; however, they should not be considered commerce-only solutions. MACH is more of a new way of thinking about building software than a sector-related tool.

Patrick Friday, CEO at Vue Storefront

What's inside

Microservices for Modern Commerce: Defining the "M" in the MACH Architecture for Enterprise Commerce

Kelly Goetsch, President of the MACH Alliance
and CPO of commercetools

We have entered a new era in commerce – one where consumers demand seamless transactions everywhere, all the time. Reacting quickly to these changes is a must for enterprise organizations to stay ahead. Unlike traditional, rigid monolithic platforms that come with lots of restrictions, microservices are independently developed, deployed, and managed by a small team of people from different disciplines – providing the flexibility to build new prototypes and deliver new features to the market quickly.

What are Microservices?

Microservices are small, standalone applications fronted by APIs that can be designed and deployed individually like Lego bricks. With each microservice you can add new functionality like a shopping cart, payment and search functions. Microservices are modular: Each microservice can be developed and released independently, and are loosely coupled with no sharing of data between microservices. Since microservices work independently of one another, they can be replaced individually without impacting the rest of the application. As Rebecca Parsons, CTO at Thoughtworks, explains in this video, microservices come with smart endpoints and dumb pipes. As a design principle, this approach favors basic, time-tested asynchronous communication mechanisms (that take the form of APIs) over complex integration platforms. APIs allow the focus to be put on the microservices by dedicated

teams. Product owners, marketers and process managers manage their own pieces. Thus, the workload shifts from IT administrators to end users. After initial development and implementation, all IT has to do is to keep things running smoothly. That brings both challenges and advantages.

The Advantages of Microservices for Modern Commerce

Facing changing customer demands, the rise of the mobile web and ever shorter innovation cycles, merchants need to make sure to have both an organizational and a technical structure that allows for agility and speed. Today, speed is a leading competitive differentiator. Monolithic e-commerce applications, which still drive many sites of big retailers worldwide, have become bottlenecks for innovation. And while the monolith can be deployed at once and tested and monitored more easily, this comes with complexity and inflexibility.

In comparison, microservices offer the following six benefits:

1 Reduced software complexity

The scope of a single microservice's functionality is limited. That makes maintaining and updating so much easier. You only have to care about messages from other microservices that you subscribe to (inputs) and your API that can be called (outputs).

2 Specialization

Microservices allow for choosing the right tool for the right task. Each microservice can use its own language, framework or ancillary services best suited and preferred by the team using it.

3 Decentralized responsibility

Dedicated teams take full responsibility for "their" microservice. This leads to smaller codebases, which help developers to focus and have a closer connection with end-users. This leads to better motivation and more clarity.

4 Faster time to market

Building a microservice requires a cross-functional development team that works independently on their project. This reduces synchronization efforts between teams and allows significantly faster deployment.

5 Increased resilience

A business application made of an array of microservices has no single point of failure. If one service no longer responds, this does not automatically break the whole application.

6 Highly scalable

Microservices are small and work independently. This means it's easier to scale them vertically and increase the overall performance of the whole business application. Without having to scale the entire application, you can scale up a single function or service. You can deploy business-critical services on multiple servers for increased availability and performance.

Experimentation drives innovation

Microservices offer the perfect environment for responsive online commerce. Testing a new service, offer or special promotion can be done quickly and more frequently thanks to smaller, dedicated sets of functionality. This is commerce-as-a-service in the truest sense. Microservices provide customers and their tech teams with the foundation to do exactly what they love: Create something new. This leads to better customer experiences and, as a result, retailers benefit by being able to stay ahead of customer demand – and the competition.

Microservices empower teams to take ownership

The decision to switch from a monolithic commerce platform to flexible microservices not only affects an organization's digital output, but it also requires structural changes to

teams and hierarchies. While large monolithic applications often require around 100 or more backend developers, microservices require small teams between 2-15 people to develop, deploy, and manage a single microservice through its entire life cycle. Each team should have a wide mix of skills including business, operations, and developers. A couple advantages of creating these small teams includes fewer communication pathways (less chance for error or confusion), more autonomy when it comes to choosing the technology best-suited for solving each business problem, and the right architectural and implementation decisions that apply solely to each team's microservice.

Microservices Challenges

Instead of a classic, horizontal structure, microservices require a cross-functional structure with vertical teams that work independently. Organizations will need to have the right infrastructure and tools in place to orchestrate and monitor their microservices architecture. The following challenges come with choosing for microservices:

- **Decentralized data:** Each microservice has its own data store. Multiple databases and transaction management require extra attention.
- **Testing:** When testing a microservices-based application, you'll need to confirm and test each dependent service. This makes integration testing and end-to-end testing more difficult and also more important — since one failure can cause something a few hops away as well.
- **Deployment:** Deployment requires more careful attention, especially at initial set up. You'll need to think about how services are rolled out and in what order, andiInvestment in automation will be necessary to save a lot of time in the deployment process.
- **Monitoring:** A centralized view of the whole system is critical to pinpoint bugs. Remote bugging is not an option across dozens or hundreds of services.

And with microservices relying heavily on eventing, certain new challenges will arise. Without using automation and advanced methodologies such as Agile, communication can be hard. This calls for DevOps tools such as CI/CD servers, configuration management platforms APM tools for network management. You need a common container orchestration system, multiple tiers of load balancing, and service discovery to make sure that the services are deployed correctly and to monitor whether they work together as planned. Companies already using these tools will find starting with microservices is easy. If these extra requirements need to be adopted however, this can be a challenge for smaller organizations.

So why then, would you go with a commerce solution that uses microservices instead of sticking with the old monolith approach? While more planning and consideration is certainly required at the beginning, particularly if migrating over from an old legacy e-commerce solution, the initial effort pays off by making commerce for your business more flexible, maintainable and simpler to run in the long term. The replaceable nature of microservices also provides for better futureproofing with easy upgradability. It's exactly why companies like Amazon, Apple and Netflix have fully embraced the microservices approach.

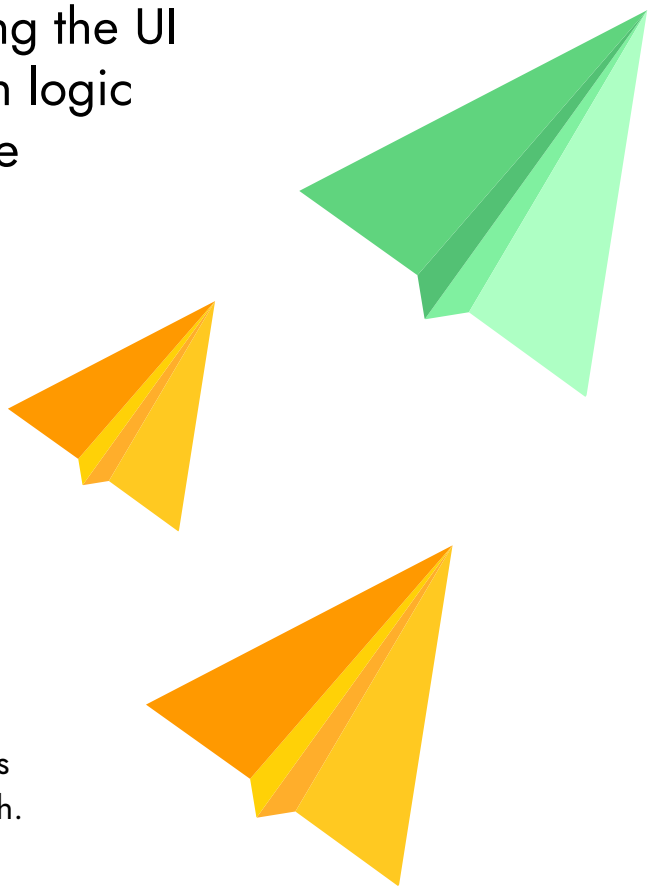
Your Microservices Starting Point

To capitalize on microservices, your organization needs to be capable enough to organize and manage it. Required tooling and infrastructure are offered by commercetools, a leading e-commerce solution that brings microservices, API, and cloud together. Their cloud-based microservices solution contains 300+ commerce APIs that can be used individually. With ready-made commerce blocks you can easily create or supplement your own infrastructure. That enables true business agility and e-commerce success.

What is headless architecture, and why is it the future of eCommerce solutions?

Patrick Friday, CEO at Vue Storefront

This relatively new concept in software development is becoming one of the most significant trends that will soon change the face of eCommerce. Based on the idea of separating the UI (frontend) from the application logic (backend), it provides even the most mature businesses with the maneuverability and flexibility of start-ups.



"Headless architecture" is an approach that fits into broader trends, such as microservices architecture, which have been a lifeline for the systems of Netflix, Amazon, Uber, and others. All of them made a significant shift from all-in-one solutions to microservices because they realized that sticking with the applications they started with was jeopardizing their growth.

The giants already moved to microservices

Netflix was one of the first businesses to conclude that a monolithic architecture is not an optimal solution for a complex application, because the components in a monolithic app are tightly connected and a single mistake can lead to several days of downtime. In the VOD business, it is a dealbreaker to users, and so the risk was too high to bear.

Amazon, now the largest eCommerce platform in the world (which sells literally everything), started out as a modest bookstore. A two-tier all-one-platform suited perfectly this, at the time, relatively small entity. However, when Amazon started to grow, it faced a pressing problem with the system's scalability. Typical bottlenecks such as long deployments, hard to handle vast databases, difficulties with adding new features, and fluctuating website traffic, delayed the company's growth.

Uber was, at first, a straightforward app available only in San Francisco and consisting of a few features like connecting drivers and users, billing, and payments. The coupled architecture of the IT system was just fine for a local enterprise but, similar to Amazon, became a massive roadblock when Uber began to scale.

Coca-Cola Company, with over three thousand products worldwide and subsidiaries in every country of the world, faces the significant challenge of connecting entities on different continents. Coca-Cola's Global IT group decided to leverage microservices and APIs to support its international growth and gradually replace their legacy software.

Etsy decided to switch to microservices after struggling with serious performance problems. Their IT team needed to decrease server processing time, while simultaneously

supporting the development of new features required to increased extensibility from the platform. To overcome the challenge, Etsy decided to redesign their platform and set up an API as a key component accessible for developers.

Before it broke the barrier of 75+ million active users monthly, **Spotify** was looking for a solution that would scale to millions of users, support multiple platforms, and handle complex business rules. Decoupling the IT system allowed the company to build flexible structures that are easily scalable, resolve real-world bottlenecks in a short time, test different solutions safely, and become less susceptible to large failures.

Zalando hit a similar wall in 2010. It had already transformed from a modest store that primarily sold flip flops to a widely known fashion brand—and their Magento-based eCommerce system couldn't handle so much more load. Switching to microservices gave Zalando the possibility to speed up the integration of new innovations and A/B testing which one would bring the best conversion. No wonder, in the fast-growing eCommerce market, there is no place for laggards—and the fashion industry is particularly ruthless with those who deal in last year's trends.

In fact, it's the same story told many times over in different major companies...

Microservices are, however, not limited to the big, globally-operating brands. They are very democratic, simply because they don't force a bloody revolution in the system, by killing off the whole class of existing functionalities and restarting the architecture from the very beginning. On the contrary, they are very agile and enable the addition of new business functionalities to existing monoliths, and integrate with standard UI with almost no technical support. In practice, it means that several different platforms (e.g., CRM, CMS, OMS, loyalty programs) can be 'covered' under one umbrella (i.e., frontend) and managed by a person without deep technical knowledge.

What can businesses gain while implementing microservices?

Microservices are a key component of a modern architecture approach that empowers the technology and business functions across an enterprise to own their tech roadmap. As part of the broader adoption of API-led solutions, microservices support the transition from platform-driven to technology/software-driven architectures.

In the past, technology ecosystems were built around a single software, or a few software providers, with platforms offering a suite of functions. It was difficult to introduce a new capability, such as a new channel or product, without impacting the rest of the platform.

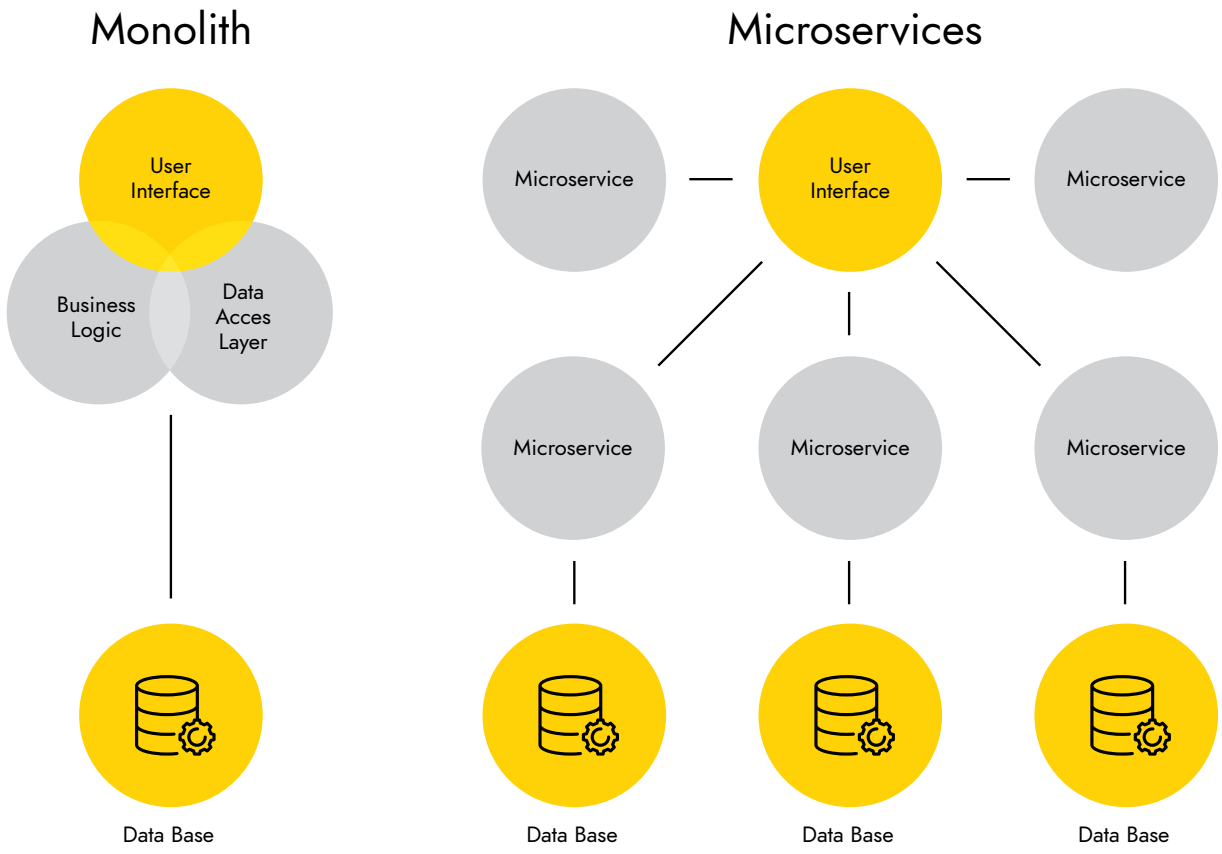
By changing one element or adding to it, another part of the platform was affected. Alternatively, the business could build a bespoke service or capability, which added complexity to the environment and ultimately resulted in higher costs and complicated releases. Using a microservice approach driven by APIs will largely remove these risks.

Coupling microservices with an API-led and cloud-native solution strategy enables companies to gain more agility, giving them the ability to innovate faster and prepare for future market disruptions. Microservices increase complexity over "monolithic" platforms but, when executed in the right way, companies can benefit from greater flexibility, capability, and the longer-term benefits that follow."

Matt Bradbeer, Director, Client Partner, EPAM

Headless is about to change the eCommerce landscape

A traditional architecture provides all the components needed for managing and publishing content on the web with a single codebase. Fifteen years ago, it was the only and the most convenient option for eCommerce businesses. The challenges have just started to appear. Traditional, all-in-one solutions were designed to give full control of the system but, as it turns out, control must go hand-in-hand with flexibility. Without the latter, the systems are simply incapable of keeping up with the pace of business changes.



Staying up to date with the latest market trends is difficult

Fast-paced changes in customer habits make the eCommerce market a highly competitive environment. The sellers must be agile enough to implement modern solutions and meet

trends and expectations quickly, because user loyalty is very fragile and depends directly upon the experience they are given. The "suits" make it challenging to deliver great experiences when all elements are tightly connected; moving to modern frameworks that deliver, for instance, better web performance can be risky, as can changing the UI. Since the frontend and the backend are one unit, developers can't just alter the frontend layer to adjust it to the new brand identity or marketing goals without interfering with the underlying database code.

Every change needs a lot of care

The updates made in the "all-in-one" systems need to be thoroughly tested to make sure nothing goes wrong because one single mistake can cause the whole system to collapse. With microservices, every piece of the software can be upgraded independently, without affecting the whole system before the last stage.

One service provider limits the business possibilities

The all-in-one approach, as we mentioned above, seems comfortable at first. There is no need to dig the internet in search of additional solutions; everything is simple to develop, test, and implement—especially since the providers often deliver easy-to-use tools to handle all these processes. The problems start during growth, when default features turn out to be not the best ones available on the market, and the dependency on one IT provider reveals itself as a significant drawback.

Microservices architecture as a way to rescue old systems

Traditional architecture predetermines the usage of specific schemes. Most of the systems are powerful and do have multiple plugins and features but—especially in online businesses—it is usually necessary to stray from the beaten track to ahead of the competition. Or at least that was the accepted norm… And then the microservices appeared. They suddenly allowed for the construction of the systems by putting together loosely coupled elements, almost like working with digital LEGO.

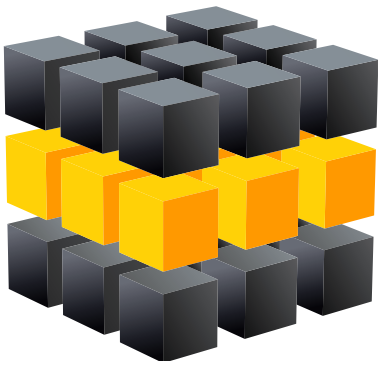
What are the main technical benefits of microservices in eCommerce?

Today, there are hundreds of proven SaaS vendors that solve specific business problems (content, commerce, personalization, etc.) much better than the basic functionality offered in suites.

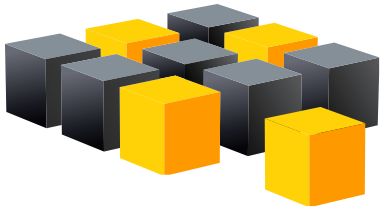
Due to the near ubiquity of support for MACH principles and how modern SaaS software works, it's easier than ever to discover, evaluate, purchase, and integrate these vendors. As needs evolve, vendors can be easily added or swapped.

We as members of the MACH Alliance represent the present and future of enterprise software and services and aim to be the catalyst for even more change.

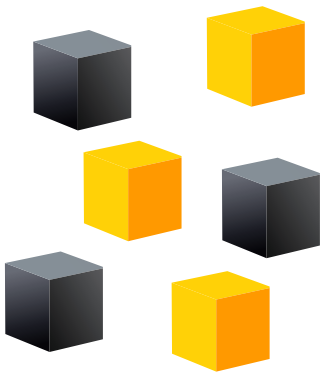
Kelly Goetsch,
President of the MACH Alliance and CPO of commercetools



Monolithic (Pre SOA)
(Tightly coupled)



SOA
(Loosely coupled)



Microservices
(Decoupled)

Microservices allow the whole system to develop at different speeds: eg. at the front with fast-paced design iteration and a field to explore new features, and at the back with taking care of security and stability. The two sides of the system communicate via API, so they remain separate entities while the data flow is undisturbed.

That model clarifies not only the structure of code, making it easier for developers to navigate it, but also simplifies the organization's structure, which can mirror it in compliance with Conway's law.

> (...) organizations which design systems (...) are constrained to produce designs which are copies of the communication structures of these organizations.
>
> **Melvin E. Conway**

API (e.g., RESTful API or Storefront API) is a bridge for data transferred between the back-end and the frontend, which provides a presenting layer in a consistent and standard format across multiple channels.

This approach has many benefits:

- Every feature can be added and tested separately without interfering with the whole system
- Every bug can be fixed independently and added only when it works well

- No single change impacts the system's integrity
- Each team can develop, deploy, and scale their services independently of others
- The entire system but divided into parts, making a new developer's onboarding process easier
- Time-to-market of a new application or feature is shorter and that makes the entire business more flexible
- We can test/add/remove various third-party platforms

Let's see some examples of these benefits.

One of the crucial obstacles in finalizing purchases is poor user experience. Users make snap decisions to abandon the cart if they have to wait for a page to load or are forced to figure out how to find the right information. Yet, they also expect that the content applies to their personal preferences that differ depending on the country in which they live, the devices they used, their age, and other factors. That means sellers need to meet user expectations by offering precisely what they want. Nothing less, nothing more—because overwhelming people with features or content is just as dangerous as offering too little.

With an all-in-one system, every frontend design change forces a change in the backend, and vice versa, which makes every decision becomes more consequential. And this issue becomes more complicated as the business grows, e.g., expanding to new countries or offering new products. Every single change on the presentation layer, every customization of templates, site structure, or design, jeopardizes the whole system, including the database.

Microservices solve that problem. Their general idea assumes that technology is used to serve the business, not the other way around. All of the units that an application consists of are focused around business goals and can be deployed or removed as needed.

Why is delivering premium CX easier with headless architecture?

Firstly, you can implement one process at a time. That means you can fix your biggest pain points right away. In the case of order management, that's often inventory visibility and accuracy—so you get a Return on Investment (ROI) earlier. You can then use the first project to help plan additional rollouts like shipping from stores, setting up drop-shipping vendors, and so on. This incremental approach also means far less high-stakes change management, more seamless employee adoption, and, ultimately, a lot less business risk than you get with a 'big bang' implementation.

Secondly, a headless platform enhances your existing customer touchpoints. Whether it's your eCommerce site, mobile, text messages, email, social, clienteling apps, or call center, it augments them all with better processes, more accurate data, and greater scalability.

That means you can provide better customer experiences and extract more value from those systems. And as your business evolves, and customer preferences change over time—think same-day delivery, social selling, sustainable delivery—it's much easier to add new systems and processes to enhance your CX.

Thirdly, a headless platform gives the ability to react quickly. If there's one thing we've learned recently, it's that the speed at which you can adapt to changes makes or breaks your business. How quickly can you roll out new processes? Bring locations offline or offline? Reroute inventory based on shifts in the market?

While no one can predict the future, a headless architecture can help you prepare for it by providing the flexibility you need to move fast. That way you can feel confident, no matter what the future may bring.

Graham Jackson,
CEO, Fluentcommerce

Microservices are the best answer to modern challenges in the eCommerce industry

Filip Rakowski, CTO at Vue Storefront

eCommerce, which was growing at a double-digit pace before COVID-19, has recently accelerated even more. The pandemic, a new study from Kantar suggests, not only sped up the shift to online retail; it also raised the bar for those who were used to think of eCommerce as just a shiny addition to traditional sales channels. The time has come to put eCommerce in the spotlight and there is an urgent need to switch to technology able to make it possible with no bloodbath.

Headless in the omnichannel eCommerce

The most popular all-in-one solutions are undoubtedly robust and offer a number of excellent features. However, their great power comes at great cost with regard to performance. As one extra-large unit with the frontend tightly coupled to and dependent upon the backend, they can be very slow. All of the out-of-the-box features mean lots of code—and the more code you have, the longer it takes to process. Headless architecture is based on a decoupled frontend integrated with content management tools via an API, so there is no need to render so much "default"; as a result, everything runs significantly faster.

Why the API-first approach in building software enables ecommerce businesses to be more customer-centric?

Ecommerce was growing rapidly in 2019 — but the events of 2020 lit an accelerant. This increased market share is great news for eCommerce brands. However, as barriers to creating stores have lessened, competition is also increasing. To stand out in a crowd of online competitors, companies need to be laser-focused on providing what customers want.

From personalization and a frictionless buyer's journey to real-time customer support and plenty of payment methods to choose from, the consumer's implicit demands keep the eCommerce landscape in a constant state of evolution. Businesses must build for the future, whatever it brings. They need technology that can deliver customers an unbeatable digital experience and a smooth path through checkout and beyond.

Leveraging an API-centric tech stack can pave the way for these customer-focused experiences. In decades past, all-in-one platforms were the choice for most eCommerce sites. However, these monoliths prevent businesses from choosing best-in-breed solutions — or building custom solutions — that can achieve a customer-centric site. By using a SaaS platform that is open to integrations and supported by a huge ecosystem of partners, a business can build a tech stack that reflects the needs of their customers and connect all the necessary pieces together through API calls.

Equally importantly, an API-first approach is flexible and able to change with the changing times. 2020 has been a

> transformative year, to say the least, and customers' needs and shopping behaviors have changed in ways that were impossible to predict a year ago. Because individual pieces can be adjusted without making changes to the entire system, an API-first approach means it's easier to adapt to a retail landscape in flux, respond to what customers are demanding and maintain a competitive edge.
>
> **Jimmy Duvall**, Chief Product Officer at BigCommerce

The advantages of headless CMS, such as Contentful, Amplience, Contentstack or Storyblok are, however, not limited to performance. A decoupled frontend makes the UI layer more flexible and that offers numerous benefits. Product Designers and Frontend Developers have much more freedom when it comes to UX/UI design. They can prepare various template options that can be integrated with CMS and then give the content team the ability to make changes without IT teams being involved. The ultimate, though not only, result of that approach is accelerating the way the whole organization operates on the market.

The second important advantage is that API-based connection to the backend also enables businesses to serve the content not only via one default channel or but to spread it to a whole variety of electronic tools such as desktops, smartphones, wearables, and any "smart" devices. Nowadays, omnichannel is the foundation of the marketing channels and is essential for business relevancy. For example, Netflix, with its headless architecture, could reach the users anywhere within the Internet of Things (IoT) with its familiar UI and near-instant performance speeds. That is its base for future growth.

Why is headless CMS a must-have in the omnichannel world?

The aphorism "publish or perish" applies to modern business, as much today as it did to academia decades ago. A traditional, monolithic CMS is an all-in-one system created for delivering content to a single channel, the web. But today's marketer must go beyond that. Mobile-friendly is a must-have, and adapting to new channels is a matter of necessity. But that's not possible with a traditional CMS.

Because the backend content and the presentation layer are bound together, it's hard to integrate with microservices and third-party tools and distribute content to new channels. A traditional CMS isn't sustainable or scalable for omnichannel delivery.

A headless CMS can be described as an API-first content management system. An application programming interface (API) lets companies manage and distribute content to any digital publishing channel, such as a website, mobile app, tablets, smartwatches, personal assistants, AR, VR, AI, etc.

A headless CMS separates the backend content from the frontend presentation layer. It is a future-proof system that can send formatted content to whatever channel requests it.

A headless CMS acts as a content hub for integrations, responding to data from any system such as Marketo, Google Analytics, Salesforce, Shopify, or commercetools.

These integrations can give content editors insight into user interactions within different channels for personalization, which helps deliver highly-targeted messaging and interactions.

The Internet of Things (IoT) era of smart devices creates new opportunities for businesses but adds challenges for a traditional CMS. As a result, companies looking for a competitive edge are turning to headless CMS solutions to address the ever-emerging omnichannel ecosystem.

For example, companies like Photobox, Freeletics, Icelandair, Ellie Mae, and Miami HEAT moved to the headless Contentstack CMS to handle omnichannel delivery. All of these companies are publishing content to more channels faster and at a significantly lower cost.

While the headless CMS may initially seem to be a technologist's choice, it actually empowers business users to independently optimize the digital customer experience across all touchpoints. Modern enterprise-grade headless CMS caters to the needs of content and marketing teams by bringing together adjacent tooling into one user interface.

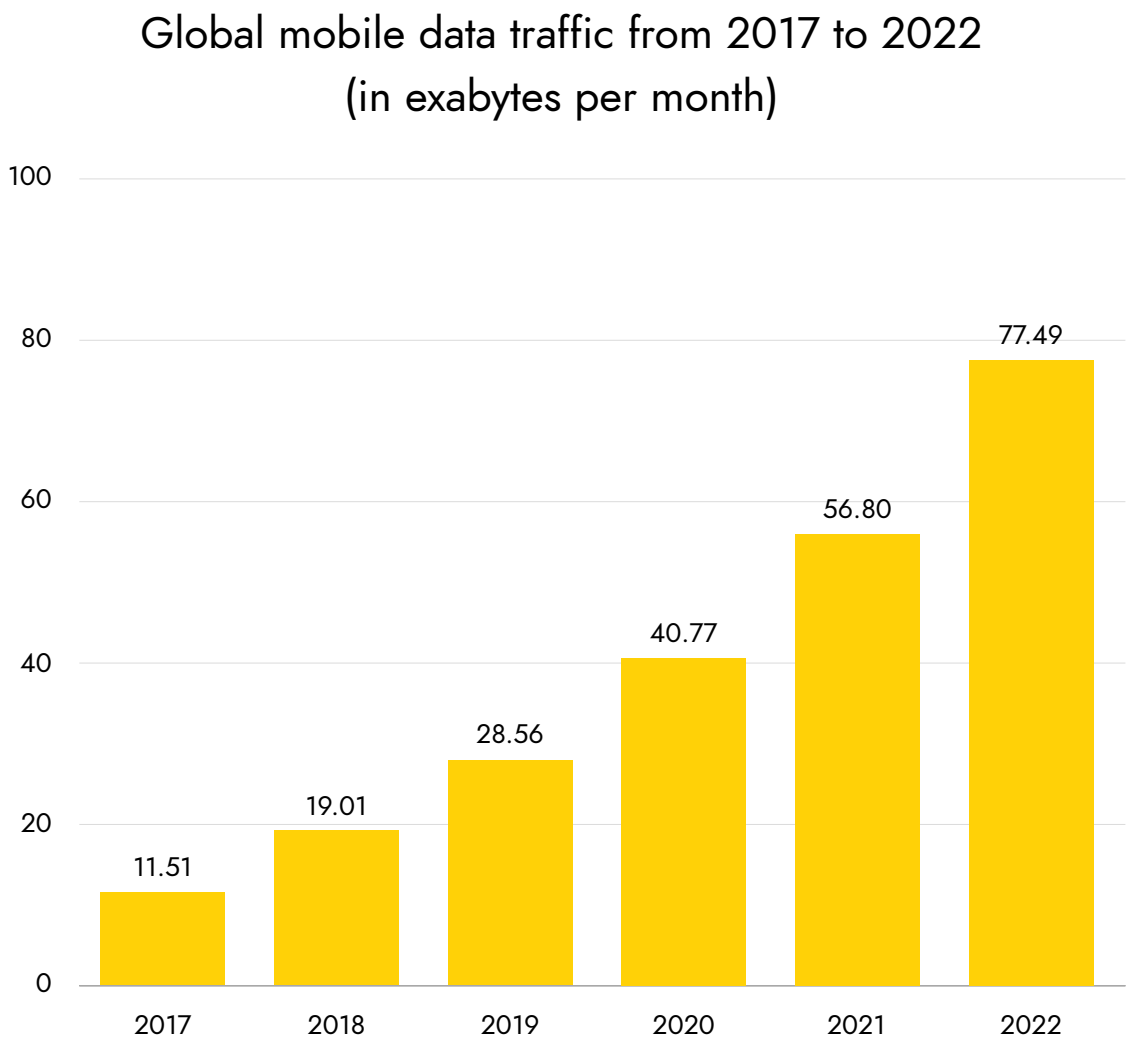
An agile CMS is, therefore, not only for content, it is an 'experience hub' for customer engagement.

Sonja Kotrotsos, Head of Product Marketing, Contentstack

Thanks to the headless CMS, it is possible to provide as many different experiences for users as needed in the omnichannel world, with no coding. It gives the freedom to adapt the content for specific channels and devices in order to deliver the best possible UX. And since the better UX, the bigger the chance of conversion, it is a solution worth considering.

The state of omnichannel eCommerce

Over 2.5 billion smartphones are used worldwide and 17 exabytes of data are downloaded by mobile devices each month. There is no doubt left: the internet has been mobile for quite some time.



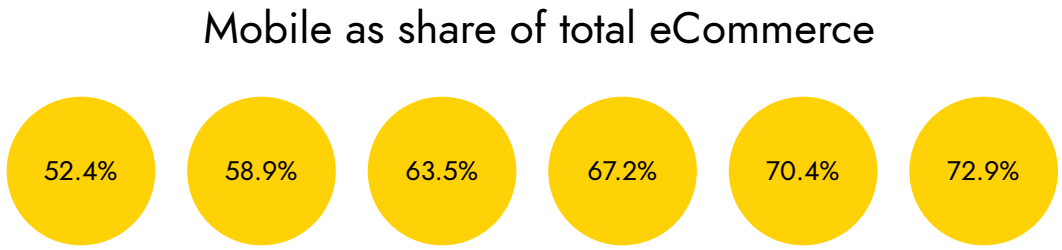
Source: Statista

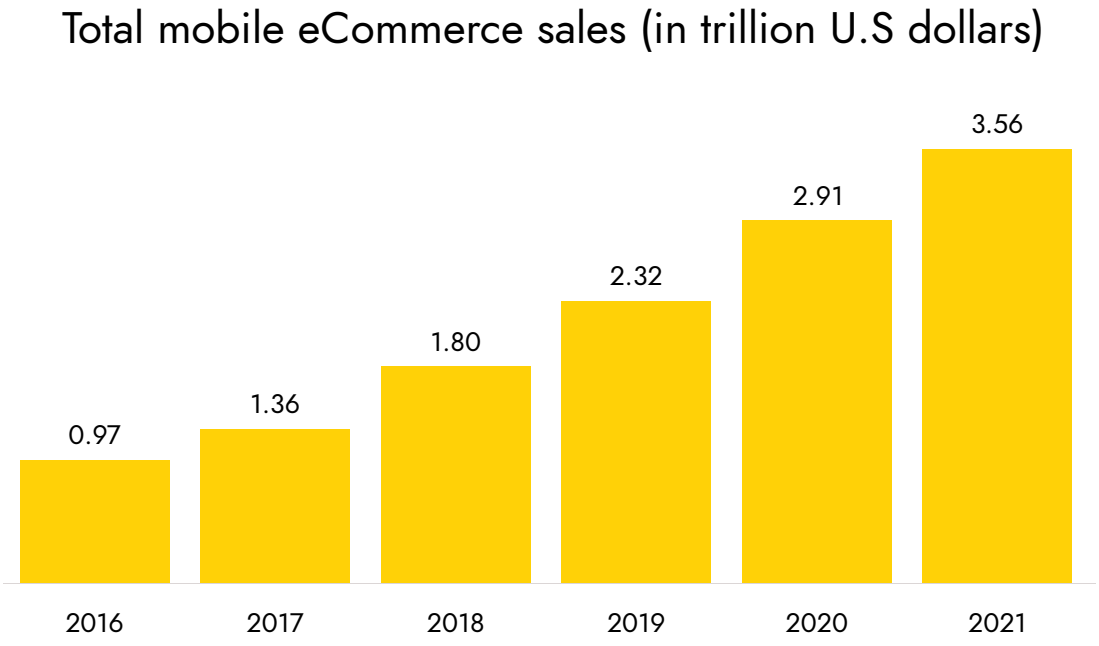
Mobile-first is a must in modern eCommerce

The progressing expansion of mobile internet entails changes in shopping behaviors, which are drifting more and more towards smartphones.

Mobile retail is already forecasted to reach $2.91 trillion in 2020 – 25.4 percent more than the $2.32 trillion registered in 2019. Revenue from mobile commerce sales in 2019 is already more than twice that of 2016.

Mobile eCommerce is up and Poised for Further Growth
Estimated mobile eCommerce sales worldwide





Source: Statista

Smooth and personalized UX as a driver for conversion

That's just the way it is! Managing user experience (UX) is a means to an end. It is not about enchanting the crowds with the unique beauty of a site—although the aesthetic part of UX, the user interface or UI, certainly matters.

UX is a part of the entire customer experience (CX). It is a way of defining the emotions evoked by getting in touch with the digital channels, such as the website, mobile app, or social media profile of a brand. It is by no means limited to visual aspects, despite the fact they attract direct attention.

CX concerns many other factors like accessibility, ease of use, and load speed, to mention only the most important ones. All of them, combined, build the general feeling, i.e. customer experience.

However, UX is more than JUST an intuitive page structure, a beautiful interface, or lightning-fast performance. It is all of them and more. Good UX requires combined analytical, technical, and creative competencies. The end game has a clear goal: encouraging the customer to act in a certain way.

It doesn't require great imagination to realize that if the user gets tired of waiting to the page loads, gets annoyed with filling out endless forms, or is confused trying to find out what to do next, he will ditch the store and move on. Nothing, not even the low price, will not stop frustrated users from resigning from sales. Besides, lowering prices continually is a dead-end for every business; it's far more effective and sustainable to raise the quality of the site.

We'll let the numbers speak for themselves. Forrester Research stated that a well-designed UI could increase the conversion rate by up to 200%, and a better UX design can take that up to 400%. On the flip side, poor user experience can wreck all your efforts.

The frontend on the front line in the battle for customers' wallets

UX-related issues, as we pointed out before, are not just about the visual layer; the frontend technology is vitally important. And let's not be deceived, it is still a huge area that is

dynamically changing and can be a real headache for developers who are trying to embrace all the latest trends.

New frameworks and libraries are popping up so often that businesses of any scale are constantly being tempted to try new things in the name of finding a competitive advantage. This is, by all means, a good approach; after all, we all know that the saying "if you are standing still, you are going backward" is as or more true in the eCommerce industry than anywhere else.

The frontend domain is a real battlefield where all the struggles to win users' attention take place with new features, plugins, and themes. When its boundaries are strictly drawn - for example, adding something like a new payment gateway doesn't jeopardize the stability of the system - it is all good. There is then a safe space for testing how new things work and how they resonate with the client's demands.

Headless architecture, since its general idea is dividing the backend from the frontend, gives space for both better user experiences and for experimentation.

Developers are free to try out new innovations while being sure of platform stability at the same time. This is a way for assembling more future-proof systems that don't force reimplementing entire units with every revamp of UX design.

Why did the UX start to be crucial in driving conversion, and why with the headless approach, it is easier to take care of it correctly?

A MACH or headless approach to building a technology stack helps solve a lot of challenges organizations face nowadays in trying to provide customers with the best possible user experience across every touchpoint. Presence across multiple channels is a must for brands that want to stay relevant, but there are still brands who struggle to provide a seamless experience across the board Shopping can begin on Google or social media, go through email, stop off on mobile but end with desktop...every single interaction has to reinforce the brand at every moment in the shopping journey to differentiate, inspire, and eventually to purchase.

MACH enables eCommerce businesses to leverage best-of-breed frontend technologies such as modern JavaScript frameworks like React and Vue, alongside other JAMstack related technologies to reduce page load speed and reduce network latency over world-class CDNs. Faster page load speeds positively impact organic traffic through higher positionings on Google SERPs, resulting in more site visitors and customers eventually.

Adam Sturrock, Product Marketing Consultant & Enterprise Solution Architect, Amplience

The key advantages of the microservices approach in business

Aleksandra Kwiecień, Content Manager at Divante
Kaja Grzybowska, Content Manager at Vue Storefront

Success in online business means traffic increases, dealing with robust databases, adding new features, and integrating external tools.

The once convenient and easy-to-use all-in-one platform is soon overwhelmed as the market moves on and the business grows. Microservices are the next chapter of this story.

Microservices architecture is an approach in which the structure of an application collects loosely coupled and business-oriented services. Though it is definitely not an ironclad protector against all of the issues that come with traditional architecture, it deflects many of them.

The key advantages of the microservice approach:

1 Business constancy

With the microservices approach, all business features are granularly split into separate micro-applications, even when migrating from monolith architecture to microservices.

Each can be developed, tested, and deployed separately and without shutting down the whole system or risking losing stability.

2 Faster time to market

Leveraging a decentralized development process, you can test out ideas in a short time and introduce innovative solutions to customers before your competitors. Progressive web apps are one such solution and we have run a proof of concept for the Magento platform. What's more, by promoting teams whose features are published for production, you can raise their accountability, effectiveness, and engagement.

3 Domain expertise

Microservices allow you to have separate services for things like promotions, checkouts, or product catalogs. Each can be continuously improved by a dedicated development team with business analysts and developers on board. Such an approach builds engagement of the team and speeds up development and innovations.

4 Simpler knowledge transfer

Leveraging the Single Responsibility Principle means that a single microservice only performs one business function. Therefore, developers can create more efficient, clear, and testable code. The overall microservices architecture is much easier to understand so even new developers can maintain or modify it quickly and with fewer mistakes.

5 Easier outsourcing

With the microservices approach, all services are separable and relations between them usually have to be well documented. It's therefore quite easy to use ready-made products, such as mobile-first solutions, or outsource particular services to other companies.

Are there any disadvantages to microservices?

The benefits of microservices should be clear by now but how about the drawbacks? No system is perfect, so we can point to a few.

- Microservices aren't always the best fit for every organization as it requires a lot of research to establish the goals and needs.
- Microservices speed up business growth where monoliths fail. However, if the monolith is not complex, switching to microservices will not bring any benefits.
- Microservices require a mature, agile culture and a deep understanding between business and tech departments. The API-focused approach means the entire system is designed to bend technology to the needs of the business... but those business needs must be recognized and clearly translated to the tech team.

Traditional architecture has some clear advantages and is good for getting a business up and running with limited in-house technical knowledge. However, it is too slow and too inflexible for turning a company into a leader, shaking up the market, or outpacing the competition. Microservices are a much better choice in this case but also make sense for small businesses that are unsure what their technology needs will be in future.

From an organizational point of view, introducing microservices architecture into the company can also positively influence management, push teams towards an agile way of working, and help the company adapt quickly to changing market needs. The pace and dynamism of the company starts to mirror the technology it uses.

For whom the microservices are the best possible solutions?

Companies need to transform the way they do business, rapidly. In some cases, they need to transform completely in order to become more responsive, agile and to stay competitive. Faced with increasing competition, unprecedented disruption and constant innovation, brands need to invest in a modern approach to enterprise architectures and build next generation platforms if they want to stay above water.

With increasing market competition and in the face of massive disruption, brands are looking for new ways to engage with and mobilise their customer base. We are no longer in a world where companies can have a single online presence and expect to meet the demands of their customers, partners or suppliers.

Brands must deliver multifaceted, personalised and connected experiences across various physical and digital touchpoints; experiences that engage, move and convert. They need to offer a customer experience that surpasses that of their competition and to capitalise on the ancillary 'added-extra' services that will help them to secure recurring revenue streams, across multiple channels.

For the manufacturers and distributors who were typically fine-tuned for the 20th century economy which rewarded the efficiencies of scale and the globalisation of the supply-chain, their market position now depends on their ability to transform their business at speed.

They are racing to become more agile, responsive and adapted to continuous changes and market competition. Brands need to embrace new digital capabilities and harness the tools that will enable them to deliver better customer experiences and improve business operations. From empowering sales reps with the insights and capabilities to better respond customer needs, to the development of a more versatile supply chain that's more in line with how customers are consuming services and products.

When we apply a MACH approach to those challenges; the introduction of microservices means faster deployment times, removing the dependence on legacy technology integrations / barriers. Cloud hosting means the ability to scale at pace, adding new sites, systems and functionality without slowing everything else down in the process.

By decoupling the front and back-end technologies brands can enable their teams to respond quickly to changing market conditions and deliver a UX that maximises conversion. It's this ability to make connections between traditionally siloed streams of the business like CRM, Data and Commerce, that gives brands the power to see what's really going on in their business and be much more intuitive in their response. In times of disruption or rapidly changing market conditions, you need tools and technologies that will enable you to make better business decisions faster, to continuously innovate and to incorporate continuous feedback to drive better outcomes.

Companies need to rethink their organisations if they want to become more agile, reactive and innovative. They also need to rethink their digital platforms to make them more responsive, frictionless and to be able to direct content to any touchpoint at any time. Finally, these platforms must continuously evolve to support always-on consumers and to provide the necessary scalability to cope with fast-growing demand.

Pascal Lagarde,
VP Commerce at Valtech

What is headless CMS and how does it work?

Piotr Karwatka, co-founder of Vue Storefront and CTO at Divante

Headless apps, in general, is a relatively new approach, but developers are very often the early adopters for new tools and practices. Moreover, when it comes to content management, especially text content, they have a lot to say. They manage heavy code-bases on a daily basis and the separation of the backend concern from the frontend simplifies software engagement. Here's what you need to know about headless CMS and how it works.

But first things first.... What are the differences between a "traditional" CMS and a headless one?

Headless CMS can be perceived as a backend-only repository that has only one focus: storing and delivering structured content. It makes content accessible - via a RESTful API (JSON, XML) - for display on any device. There is not just one "head" (the frontend, i.e. the website); instead, there may be several heads (websites, mobile devices, wearables) connected to API.

The general idea of the **traditional CMS** is based on a tight connection between the backend and the frontend. The backend includes a database with code and plugins that make it possible to store, manage, and edit content. The frontend - thanks to built-in templates or themes - displays the content on the one dedicated frontend; others can be added but only with the help of plugins.

How does a headless CMS enable a more sustainable development process?

When organizations are choosing a CMS system as a cornerstone of building a digital experience, they are most commonly also doing significant development in order to meet business needs - and this need is accelerating as most businesses pivot to digitally-driven experiences (interactive experiences, subscriptions, mobile and wearable apps, chatbots, voice, kiosks, etc.).

There are two real advantages to a headless architecture; development agility and risk mitigation.

Development agility is one of the key concepts for a modern MACH architecture approach. As Gartner says, most organizations "want to invest in cloud-native solutions to deliver digital experiences faster, enable developer agility, application scalability and resilience, reduce technical debt". On the frontend, agility means the ability to choose frameworks and SDKs that make sense for your organization.

On the backend development process is often even more important as it relates to long-term sustainability. As Gartner notes: "Some DXPs come with capabilities for DevOps and continuous integration/continuous delivery (CI/CD) pipeline management to assist integration and deployment"; these are actually must-have features in order to practice sustainable development. Agile organizations are typically working with changes on both the frontend as well as the backend and the content model - simultaneously - in order to build new experiences. The API-first approach of headless CMS systems means that developing and coordinating these changes is typically far easier and provides teams with the flexibility they need to launch impactful digital experiences across their customers journey.

Companies like Shiseido Professional use Contentful to power localized websites across the globe. Shiseido Professional's digital estate covers 12 regional markets across Asia, Europe, Africa, the Middle East and North and South America. In only five months, R/GA, one of Contentful's partners, built a complete modern tech stack for Shiseido Professional. Now, content that used to take a month to publish takes only minutes — or even seconds.

Lastly, MACH headless systems usually make this task of building and testing code far quicker and easier. Legacy systems usually rely on building an entire local copy of an entire environment for each developer or test instance (sometimes even requiring full copies of integrated systems as well!). In contrast, headless CMS often can create a sandbox copy with a few clicks.

Risk mitigation is particularly important. The fast pace of technological change being pulled forward as a result of new programming languages and channels mean that there is a regular need to have systems that can easily adapt to these new formats, lest you be unable to drive an experience on those new channels, or are punished with poor marketing execution (search results, social enablement, etc.). In the case of headless systems, this de-linking of the CMS system from your front-end code means that any possible future change can be accommodated.

In contrast, with a legacy CMS system, you often need to wait for a vendor to release an update or plug-in to accommodate these new requirements. In some cases, this can mean that the tight coupling either becomes an anchor (in that it will hold you back from implementing newer, supported technologies and frameworks) or a leash (in that a new version of the software may use a different front-end language or framework - forcing you to rewrite that code if you want to upgrade at all).

Mark Demeny, Director of Content Management Strategy at Contentful

Headless CMS explained

Usually, a Headless eCommerce app is a set of backend services—such as CMS, Search, Order or Stock Management—and a single, lean frontend application that integrates all the services and features into one coherent user environment. It means that the UI layer can be managed by a different team, utilizing separate skills and technology stacks than the backend technologies.

Headless CMS opens up new possibilities

The most crucial advantage of headless CMS is its flexibility as it is not only about content itself. It also concerns the third-party services, tools, and features that might be integrated freely with the CMS system. It can be said that, even though headless CMS doesn't imply anything by design, it allows almost everything by providing huge integration options.

That means the eCommerce businesses can freely choose whatever suits them best, no matter if it is an analytics tool, comment system or search functionality. And it is not JUST a technical advantage; for example, an exceptional search engine is one of the crucial factors driving success in eCommerce. When every second counts, sites need to work lightning fast and product searches should be near-instantaneous. Users need to be given what they want without having to dive into an ocean of vaguely similar items.

In what way can headless CMS boost business growth?

Headless CMS follows API-driven concepts and separates the frontend code layer and the content data. It allows for reusability, more flexibility in frontend development, scalability for complex systems, and better collaboration between different teams. These benefits result in cost-savings in introducing new functionality, omnichannel selling support, more extensible and adaptive content infrastructure. Also, content editors can enjoy a faster learning experience.

Editorial team now can efficiently create new layouts and modify content across multiple platforms, while developers can stay focused on code maintenance and integration with new technology in parallel. A good content infrastructure reduces the reliance on developers and the need of copy-pasting. By unifying the content layer under one CMS, companies can ensure the accessibility of content for use in any digital platform, ease of applying localization, hence improving brand consistency, and content compliance across all the products.

As a result, creating Marketing and Sales campaigns eventually becomes simpler, giving the freedom to focus more on the quality of content and user engagement.

Maya Shavin, Senior Frontend Developer at Cloudinary

How does Headless CMS work?

Separation of concerns

Developers hate WYSIWYG editors. They generate pretty messy HTML reminiscent of that created by MS Word in about 2003. It's improving over time but a visual editor still equals a lot of additional work. The reason for that, I guess, is that visual content editing does not separate roles enough.

Front-end developers don't like taking care of the content details. They prefer to have full control over how it renders, optimized for the user's device. Editors, on the other hand, use all the tools the WYSIWYG editor provides them with—including styling and formatting, which can cause a lot of trouble.

Modern static page generators start with a separation of concerns in order to provide the content in the form of semantically meaningful documents. The editing process is separated from the render process by some kind of intermediate format which is both easy for a human to edit and easy to process.

Here are just a few of the most popular intermediate content formats:

- MarkDown language: Popularized by Github, MarkDown looks okay even in its plain form, and can look really cool when rendered. It's used, for example, by VuePress; and Jekyll;
- WikiText: Started with Wikipedia and is now being used by popular Confluence (Atlassian.com) and GitBook projects,
- JSON: Generated from a WYSWIYG editor (like ProseMirror) or Headless CMS (like Prismic).
- These human-oriented formats, as we might call them, have gradually replaced the old XML, RSS and ATOM formats.

Most modern, Enterprise CMS systems, like Contentful, Contentstack, Amplience, Adobe Experience Manager, Prismic or Storyblok support JSON as an output data format. This is great for separating the Data model from the rendering pipeline, which is crucial for introducing headless architecture.

Object-based model

For most rendering frameworks, the rendering process is based on components/widgets that are fed with the structural content. There is usually a JSON config file or feed making up the page of a particular component (written, for example, in React or Vue.js). The provided data is injected and the components take care of the proper rendering process.

Rendering pipeline

Typically, the content starts the flow within the WYSIWYG editor where Visual Merchandisers can design the nice-looking pages, as well as managing the blocks and widgets. The content itself is mapped into an Object-based data model. Usually, each visual block/widget is represented by a single JSON data object.

The rendering pipeline is then fed by the GraphQL or REST API with these JSON objects that are then mapped to the JavaScript components (React, Vue, Angular) that contains all the business logic required to render the content - optimized to the device.

AMPLIENCE CASE STUDY

Headless CMS in practice with Harry Rosen

Harry Rosen's is a 66-year-old Canadian luxury menswear retailer with stores across the country. They follow modern, best practices in business providing personalized customer service and a high touch experience. Their goal was to bring this customer-first experience into all web and digital touchpoints.

For Harry Rosen, a customer-first experience means:

- Lightning fast - especially the mobile experience.
- A shopping experience that is personalized and contextualized, to put one of thousands of the products they sell in front of the right customer at the right time.
- Lastly, the team at Harry Rosen decided early on that their digital experiences can't be a competitor to their in store experience. Instead, it must complement and enhance.

Tying this together with a real-world scenario, advisors who bring customers to the site or post something to their social media that lead to sales are appropriately credited. This ensures a consistent customer experience and internal alignment to deliver a superior customer-first experience. Harry Rosen recognized that their customers were in the driving seat, they choose when to shop with an advisor, when not to, and when to engage customer service. There are thousands of potentially unique customer journeys that combine these personalized in-store and online moments and interactions together.

From Myopic Monoliths to Modern Microservices

With this vision set, it was time for Harry Rosen to begin to take a deeper analytical approach

as to where they were today when it came to their technology platform and if it could support the types of customer-first experiences they were looking to deliver. A forcing function began to drive this to a head. Harry Rosen faced a decision in early 2019, they were on an older version of a traditional, legacy commerce platform and had to face upgrading to a newer version or look back to the market for another route. With the experience of running a monolith behind them, they knew that development required a lot of specialized knowledge but simply wasn't agile enough to deliver on Harry Rosen's customer-first vision. Harry Rosen understood that they could use the technology as a competitive edge.

> eCommerce in a box is a commodity.
>
> **Ian Rosen**, VP Digital & Strategy at Harry Rosen

As Harry Rosen's team evaluated the market, it was pretty evident that there was a standard eCommerce framework the majority of retailers and brands were working with: the monolithic commerce platform suite. Once key stakeholders began to consider how to replicate and translate in-store experiences and processes to the digital world, it quickly became apparent that APIs are the critical component and that Harry Rosen had to "own the glass" and connect the backend processes seamlessly. To deliver their vision, they quickly realized that a one size fits all approach and selecting a monolithic commerce suite wasn't the right approach. Instead, Harry Rosen required the flexibility in selecting best-in-class components for each function that together would create an unparalleled experience, greater than the sum of its parts.

The Migration Challenge

Harry Rosen employed the services of Myplanet to assist them with the solution architecture planning, vendor selection and implementation itself. To compliment these efforts and to ensure Harry Rosen maintained control over their vision, they also employed two developers to handle ongoing maintenance and leveraged internal business analysts to translate processes into project requirements and scopes. eCommerce sites don't operate in a vacuum and require collaboration with other systems and teams. A divide and conquer approach was adopted where individual teams contributed to specific integration requirements and specifics, such as merchandising and content teams requiring input from the PIM, ERP and CRM versus the fulfillment side of the organization when it comes to inventory and the OMS.

Harry Rosen and Myplanet spent a lot of time in store and talking to customers and store employees, mapping out the white glove customer-first experience they are renowned for.

There is no one way that Harry Rosen sells, and it was challenging to distill all the processes and nuances down to a single way to serve customers. Some customers wanted a fast, low touch, predictive service while others wanted to spend more time with the Harry Rosen team and understand the details of each product as part of their purchasing decision process. The most important thing, therefore, was to deliver a degree of flexibility, building in the foundational pieces to allow Harry Rosen brand online and in-store to rapidly change, morph and adapt as required.

From a technology standpoint, the monolithic platform is broken down into its constituent components.

Harry Rosen Headless Architecture

- Next.js was selected as the front-end javascript framework, deployed and hosted on Vercel.
- Amplience was selected for the CMS and DAM capabilities to power the experience layer, including the menu navigation, pages, slots, content types, and digital media assets.
- Commercetools was selected to provide commerce functionality, including the product catalog, carts, orders, pricing, and promotions with a flexible data model under the hood, enabling Harry Rosen to handle a wide array of complex commerce use cases.
- Lastly, Algolia was selected to provide an unparalleled search experience for both customers and staff, with advanced search capabilities, refinement, personalized results, and voice search.

The Project Roadmap with mounting pressure

With a rapidly changing retail landscape, compounded by the effects of Covid-19, Harry Rosen and Myplanet split the implementation down into components to be tackled in short sprints. As more traffic began to hit the monolithic Hybris platform and the pricing/promotion strategy shifted, Hybris began to come under strain and required the new technology stack to be set up in parallel. Harry Rosen then had to begin closing stores due to lockdowns. All these factors meant that the new online experience became critically important to continue operations in this new world we all now find ourselves in.

> This accelerated the MVP launch down to a 16 week deployment schedule.
>
> **Kathrine Jones**, Director of Platform Strategy at Myplanet

Harry Rosen Roadmap

These solutions are stitched together on the frontend to create the overall customer experience and are synced on the backend to automatically and seamlessly handle data transformations, which we'll cover in more detail over the next few sections.

Experience & Content Modelling

Traditional monolithic platforms have a page centric model with one site template engine that pulls in various objects and elements from an underlying database.

When we start to decouple this with a MACH-based approach that has a frontend such as Next.js that is based on more of a component based approach, content modelling is required. This is where Harry Rosen began to map the content managed components on the backend to the frontend components. This extends beyond pages to a more granular view that revolves around slots and delivery keys that is easier to map at the component level.

Vercel to Amplience Mapping

Breaking this down further, you can begin to understand a content graph based approach. One example is highlighted below where we see a composite view of a content block consisting of a title, image and CTA. This content block can be embedded into a page or listing and is made editable on the backend. Content types are matched with a frontend component that can render it.

More complex content types can be created by nesting where required. Complexity should however be kept as low as possible to balance how simple it is to implement and maintain in code. A great tool like Storybook will allow you to quickly design these components first and share them with stakeholders and content creators to ensure feedback on scope, agreed creator flexibility and signoff is attained before schema creation in Amplience and binding to the APIs takes place.

Headless Preview Capabilities

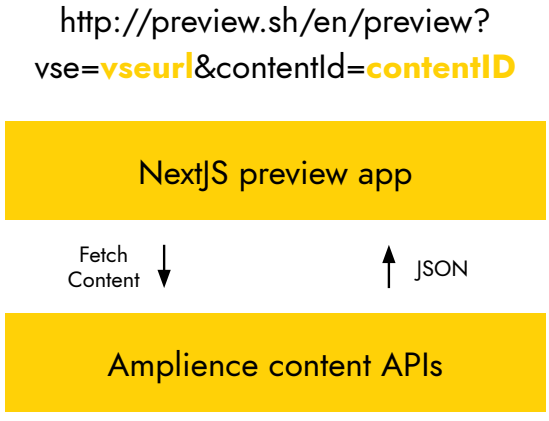
Once a components design and schema are implemented, this enables Harry Rosen to make use of the preview feature in Amplience based on a component's unique delivery key, enabling content creators to see what the changes they are making look like before scheduling and/or publishing takes place.

Harry Rosen - Amplience headless preview

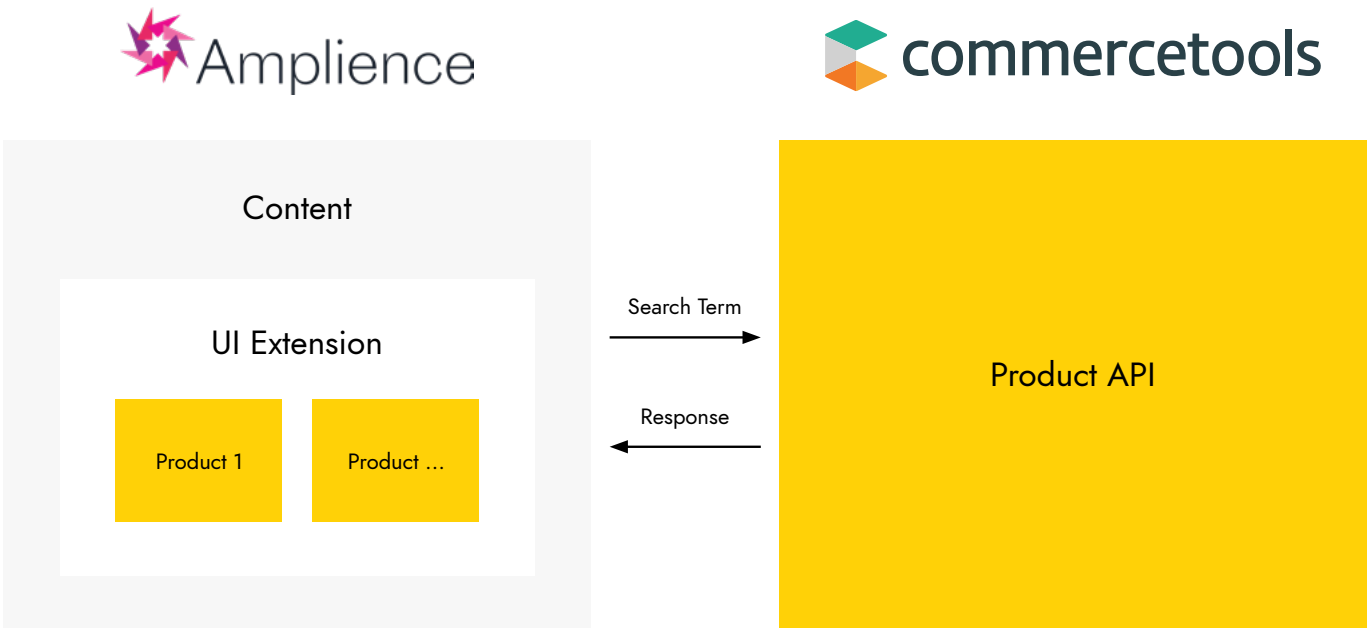
Myplanet built this preview application by taking the virtual staging environment (VSE) and content IDs to hydrate and render an individual component. The code in the preview environment should match the production code to ensure preview accuracy for the content creator. Once the preview environments are created, the developer is no longer required and content creators no longer face bottlenecks in their content production pipeline beyond the content creation itself!

> Mature best-of-breed MACH solutions, such as Amplience, can rival and surpass both the developer and editorial experience requirements.
>
> **James Brooke**, Founder & CEO at Amplience

Commerce & Content



Amplience's merchandising extension allows a content creator to pull in product content types from commercetools and leverage them in the content model and subsequent content types, components and assets on the frontend. For example, a product slider can be populated with product information held in commercetools with Amplience storing the commercetools product ID in the Amplience content model to then be leveraged by the frontend to make the call for fresh data to be pulled down and rendered from commercetools.



* Only relevant reference ID is stored

Amplience and Commercetools integration

This bypasses the need to sync vast amounts of data between providers, ensuring content displayed on the frontend to customers is always up to date and enables internal merchandisers at Harry Rosen to work independently from the content and marketing specialists.

Frontend Integration

As mentioned previously, Harry Rosen leverages Vercel to host both the frontend Next.js site, serverless functions and CDNs. Vercel can cache both the pages and the API requests to ensure a highly performant site, even when hydrating and rendering dynamic content from APIs.

Harry Rosen - frontend integration

A regular React application typically doesn't play well with search engine optimization, rendering a blank HTML file and then making dynamic requests to load content in from APIs. To remedy this, Harry Rosen took a server-side rendering approach delivered via cached pages/responses over Vercel's CDN. Instead of the frontend talking directly to

commercetools or Amplience, most requests are directed through the serverless functions caching layer. In some circumstances, this doesn't make sense, such as with Algolia search or with Amplience DAM static assets that are already host assets on it's own CDN.

Caching Strategies for MACH architecture

The Harry Rosen caching strategy makes for incredibly fast load times and a highly performant site experience. It's worth reviewing Harry Rosen's caching strategy in a little more depth.

A regular cache caches content that expires after a set period of time. Harry Rosen utilizes a "stale-while-revalidate" approach. The difference here is that the cache will refresh and rehydrate itself on expiry every 24 hours (TTL). Site traffic automatically keeps the site cache up to date, including JSON API responses. The result was response times that are typically 2-4x faster. This also ensures that APIs are protected from traffic surges and leads to a more robust solution.

Lessons Learned

Harry Rosen and Myplanet have kindly broken down why they chose commercetools, Amplience and Algolia, the biggest wins, and what they learned with each vendor:

	commercetools	Amplience	Algolia
Why we chose it	Ease of development, platform performance, flexibility	Retail focus, headless support with an understanding of business user	Speed & flexibility, personalization around product
Biggest win	Greater developer experience, speed of development/prototyping	Visualization for Business and Users, DAM, Planning Calendar, Delivery Keys	Lightning front end application, ability for business users fo make changes
What we learned	Important to have underlying data sorted out especially in retail use case with very dynamic data	Leverage UI extensions - hey enable you to keep flexibility in components, but governance for end users	Data management, Dealing with dynamic products/data/

Go headless or go home... Headless architecture as the way to win global markets

Piotr Karwatka's interviews with Jonathon Ribas, CTO at Zadig & Voltaire and Halil Köklü, CTO at LoveCrafts

Zadig & Voltaire, the French-based fashion company, while migrating to Magento 2 decided to reach out for headless PWA front-end to unify their UX across off the markets. LoveCrafts, a platform for a global community of makers, did the same, when the company decided to give up Magento and switch to commercetools.

Both of these firms believes that the market is moving from all-in-one or full-stack platforms to a bring-your-own-frontend, modular proposition.

> We are an international brand present in Europe, the USA, the Middle East, Asia, Australia, and more. Our idea was to have the same frontend to serve all our countries and empower all our local teams with the same tools in order to facilitate communication and skills learning.
>
> **Jonathan Ribas**, CTO at Zadig & Voltaire

> At LoveCrafts we are weaving together commerce, community and content for the world of makers. We needed the flexibility of a truly Headless Architecture that commercetools provides to stitch all of the different elements together to make unique making experiences for crafters everywhere on any device.
>
> **Halil Köklü**, CTO at LoveCrafts

ZADIG & VOLTAIRE

Zadig & Voltaire, over its over 20-year history, gathered a die-hard fan base among everyday customers and fashion icons as well, but recently this Paris-located fashion house has blossomed. The company was planning to expand its brick-and-mortar representation, reach out from the new markets, including fast-growing ones in Asia, and - at the same time, bolstering eCommerce channels.

Smoothing the path customers needed to take to make a purchase at Zadig & Voltaire, was the number one priority for Jonathan Ribas hired to lead the company's digital transformation, but building a friendly and efficient work environment for non-techies was no less important. As Zadig & Voltaire got very serious about digital the company faced the challenge of rebuilding its entire eCommerce architecture to replace some of the manual processes with automation.

The company decided to move from Magento 1 to Magento 2 with its headless and use the Vue Storefront as a PWA front-end to improve mobile UX, and boost performance. The switch allows Zadig & Voltaire customers to purchase products online and pick them up in one of the company's retail stores, unifying their experience with the brand across all channels.

What challenges were you facing in 2018 when you first began looking to re-platform?

Jonathan Ribas, CTO, Zadig & Voltaire: To be honest, we'd already made the decision to re-platform but my team's challenges were more connected to showing others how our vision could play out and making sure things went smoothly. We have to convince top management with an agile way to deal with projects and show them the first results of a POC based on Vue Storefront to get their buy-in. We had to make the significant choice of the partner to deal with the re-platforming and hosting and choose the correct rollout strategy. Once we'd done that, we had to make difficult decisions and prioritize features when we saw it wouldn't be possible to launch with all our initial scope. Just to add an extra challenge, it was our first PWA and Magento 2 was a brand new platform.

How did you come to the decision that Vue Storefront was the right frontend?

Jonathan Ribas, CTO, Zadig & Voltaire: We chose to do a PoC to see how easy it would be to implement Vue Storefront in front of our existing Magento 1 store, using our actual design for the homepage, PLP, and PDP in a new theme on Vue Storefront. It worked like a charm and we knew that we were in good hands with a strong technology team and business-oriented company. Our team and top management were impressed and couldn't wait to see the rollout happening.

Performance is the great business value that headless architecture brings, as it allows stores to create the best UX for new visitors and customers. One additional key point in our final choice of architecture was that Vue Storefront is an open-source and agnostic platform.

I saw a growing community that was active at events like Hackathons around Europe. In many ways, it reminded me of the good old times in the early days of Magento 1.

Did you consider any platform other than Magento 2 and how did the migration process look?

Jonathan Ribas, CTO, Zadig & Voltaire: At that time that we chose Vue Storefront, we were already ending our migration from M1 to M2 for the USA, so that part of the platform was pretty much set in stone. If that migration had not been in progress,

I would certainly have looked around more but, in the end, we use Magento 2 for merchandising and taking orders.

When migrating from M1 to M2, we decided to do a totally new project from scratch, taking all of the most important features into an MVP. It's never an easy process, as any tech team knows. No matter how well you plan, you can find out halfway down the road that you forgot something, and deadlines come around fast so you are often forced to prioritize. But we got there in the end.

Tell us about the journey and challenges to getting the first store up and running.

Jonathan Ribas, CTO, Zadig & Voltaire: We are a global brand with a complex IT infrastructure. It took 7 months of hard work. We had to work on Vue Storefront and Magento 2, as well as our in-house orchestrator which communicates between M2 and our IT applications.

There are challenges along the way but two specific ones during our process were integrating our catalog from PIM to Magento 2 and Vue Storefront and hosting a PWA.

What are the key aspects of rolling out the next stores? What stays the same, what changes?

Jonathan Ribas, CTO, Zadig & Voltaire: The biggest challenge is that the business windows for launching new stores are very short. As retailers, we have a lot of key moments during the year where we can't risk disruption to our business. Our decision was to do a release with some new features for the next countries to rollout, with a lot of testing and improvements between each release.

After launching in all our main countries, we've now got to the point where it is fast to implement a new PWA store with an existing language and the same shipping and payment methods. We can now achieve that in under a month, including all the testing.

Our rollout process becomes more refined as we go along but always starts with the same questions about currency, language, shipping and payment methods, and integrations.

How can integrations differ between countries?

Jonathan Ribas, CTO, Zadig & Voltaire: We have tried to choose the best partners to minimize new integrations in the future. This basically means choosing integrations with more global reach. For example, we have chosen a transporter that can ship everywhere in Europe and also China. And on the payment side, we choose Adyen because it has almost all the local payment methods which we want to use in the countries we plan to cover.

The USA will be a new challenge in the near future because they have different transporters. We will need country-specific integrations to best cover the market.

Was there a stabilization and fine-tuning period after launching the first shop?

Jonathan Ribas, CTO, Zadig & Voltaire: We had done a lot of testing internally on mobile and desktop before launch but the amount of real traffic and peaks around things like the release of newsletters is a different scenario and our site went down a few times due to bottlenecks in our custom integrations. Real life is not the same as the pre-release environment. The first calls to our support team after launch were very important in helping us to find some bugs we hadn't identified before.

We had precious help from Michael Bouvy from Click & Mortar and the Vue Storefront team and decided to set up a Redis cache in front of NodeJS because it was consuming too much CPU and memory.

Have you done A/B testing comparing the conversion rates or other KPIs on the new and old pages?

Jonathan Ribas, CTO, Zadig & Voltaire: We have looked at load times and we made sure to perform the best SEO migration in order to not lose our ranking, so that is something we have monitored. After launching all our countries, we will certainly see a better conversion rate but we're already happy with the move we've made.

Zadig & Voltaire handles pretty heavy traffic. Which scalability techniques are you using for stores?

Jonathan Ribas, CTO, Zadig & Voltaire: Traffic was a hard reality for us. Looking back now, we were not ready to handle the volume, though we didn't imagine it would cause such downtimes. If we had known, we would have waited before launching on bigger markets. I was really frustrated and looking at our Magento 1 handling it way better with an older architecture.

After reading and listening to some recommendations, we did a PoC on Kubernetes, as well as implementing the Vue Storefront Proxy with Redis, and then got much better results on our load tests. We also worked hard with an agency in order to remove all unnecessary blocking calls to Magento 2.

We had to rethink some of the ways we were developing things and fixing memory leaks but we now have an auto-scaling platform on both the frontend and backend.

"Real-life" performance, like good quality images and great user experience, is key for us. We don't care about reaching an 80% score on Lighthouse right now. We're still continuously working on improving the fundamentals and that is okay because Rome wasn't built in a day.

LoveCrafts

LoveCrafts is building online communities for crafters: homes for makers where they can find inspiration for their latest projects, learn new techniques, easily source all their supplies and share their creations with like-minded makers around the world.

Halil Köklü, CTO at LoveCrafts, doesn't shy away from saying that the headless approach caught his attention years ago as the best way to make technology to serve the business needs. Whilst that approach was used for building the content and community pillars of their product, the time to make changes to the eCommerce pillar eventually came.

LoveCrafts decided to move away from Magento and go headless with PWA as a frontend. The modular approach supported LoveCrafts in weaving together commerce, community and content for their makers.

What technological stack does LoveCrafts use currently?

> **Halil Köklü:** The transactional part of our product is – at the time of this interview – still run by Magento and we have spent many years scaling it from an infrastructural as well as functional perspective. To run a truly global business at scale, it needs significant investment such as multi-warehousing, proper multi-currency, and taxation, as well as performance tweaks for indexing, caching, bulk changes, and rendering. They are not achievable overnight.
>
> Magento is, however, only one part of our stack. We have built the community and content parts of our product bespoke with, amongst others, Symfony. There is also a good set of internal systems running both business and infrastructure processes. We are operating in the AWS cloud using various AWS services. We do infrastructure-as-code, continuous delivery, and so on.

Why is headless architecture an answer to the limitations of monolith-implied technology?

> **Halil Köklü:** The simple answer is that platforms providing business value such as eCommerce and content management systems are coming without a frontend, the head. They instead provide value through fast, extensive APIs. If done right, all functionality, including admin processes, is available through APIs. You've got full control over the frontend; you can focus on building great user experience and you are not held back by rigid constraints of the backend platforms used.
>
> You can have multiple frontends or touchpoints like native apps, kiosks, in-store, IoT or whatsoever interacting with the same APIs, so you don't end up implementing the same processes in many places. This is the reason why more and more eCommerce platforms support GraphQL to provide this feature at scale. This is, in contrast, to let's say model-view-controller (MVC) applications where business logic can be accessed by including

code. This does not work if the consuming code is not the same, is on a different server, written a different language, etc. amongst other benefits, with the headless approach, you can scale the frontend and backend separately.

When did the idea of going headless spark in LoveCrafts?

Halil Köklü: Honestly, as the engineering team at LoveCrafts, we have been talking about this architecture since the beginning. Obviously, the idea evolved over time with the technologies available. Whilst we adopted that target architecture in community and content, the commerce part has not made it yet for various reasons. The Magento frontend still dominates the user experience. At LoveCrafts, we are weaving together commerce, community, and content for our makers. You can't really fragment the user experience by which backend or legacy system is used. All these three pillars are closely interlinked.

Until now, we were trying this with technologies like edge-side includes (ESIs), but it's not working that well from a performance perspective. Running multiple frontends intertwined, as we ended up with, has many productivity and performance issues. These include starting from sharing CSS and JavaScript to ensuring they are up to date. We need what we call a backend-agnostic, unified user experience.

In the case of Magento 1, you have code and business logic in the controllers, models, and views. It's a nightmare for consistency. Headless Architecture is, however, not new at all. At the very least, if you have a frontend talking to APIs, you have some element of this already in place. I would say Headless Architecture and headless commerce are buzzwords for the emergence of API-first or even API-only off-the-shelf solutions. So, we are not talking about bespoke implementations but major eCommerce software vendors adopting this trend. The market is moving from all-in-one, one-stop or full-stack platforms to a bring-your-own-frontend, modular proposition.

What were the reasons LoveCrafts decided to go with commercetools?

Halil Köklü: It was not an easy nor quick decision, and we gave this a due process. A platform change like this does not happen every day and we want to make sure that we will not only cover our needs now but also gain the ability to scale for the future

without the need of migrating again for another 5 or 10 years. We did market research and looked at 30 platforms, of which we reviewed 10 in tail by speaking to solution architects and going through each of our requirements.

The main factors for going with commercetools were actually straightforward:

- Chemistry, approachable, lack of
- marketing gibberish
- API-only approach
- GraphQL support
- Modularity – when something does not meet your expectations or you have new use cases, you do not need to do a full migrationagain
- Great and complete documentation
- Really good overlap with our requirements
- Sensible and scalable product data model – e.g. translations are represented in localizable attributes rather than requiring a new store
- Flexible pricing model
- Staging capability for product data, which allows making changes and QA'ing

Why did Vue Storefront turn out to be a good fit in terms of commercetools customization?

Halil Köklü: It's actually funny. We have spent so much time discussing headless commerce options that, when we have decided on commercetools, my next topic was to decide on a frontend.

Nigel, the co-founder, joked "you have convinced us to go headless and now you say we need an actual head"? I think we had four options: First, to extend the community frontend which is built on Symfony and Backbone to render the commerce pages like product page and checkout. However, the technologies were rather outdated so it felt like we are missing out on opportunities here.

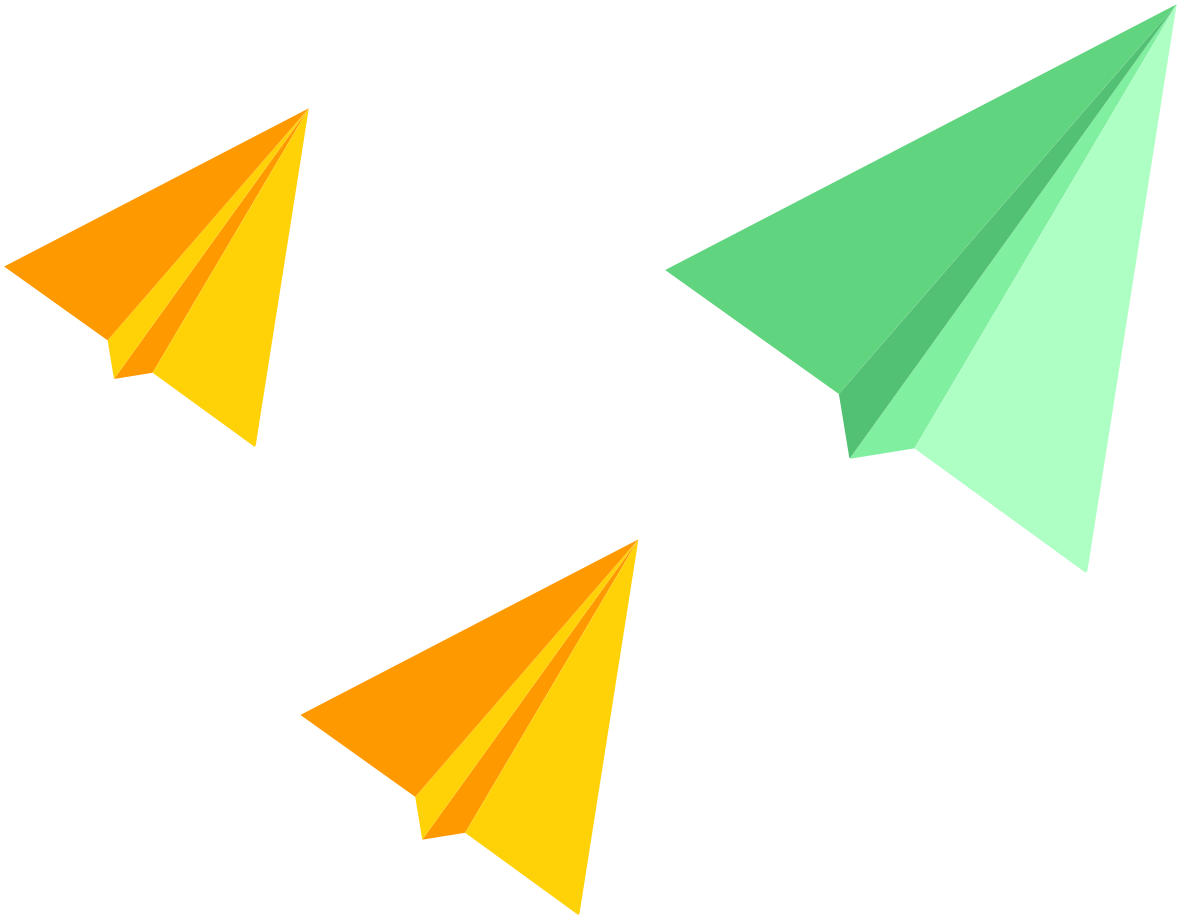
Second, to use a somewhat traditional CMS with a commercetools integration like Hippo. But no one had experience with that and we would probably be back at square

one with rigid structures defining our user experience. Third, to build our own frontend. But then again, how long is it going to take? Fourth, build on top of an existing but modern and flexible backend-agnostic frontend.

A colleague mentioned Vue Storefront to me and we were excited straightaway. I messaged Patrick and the next morning we were talking. We discussed that VSF is relying on normalization to be backend-agnostic, so things like product data are indexed into an ElasticSearch cluster which VSF can read from.

We said, "Hey, commercetools already has great APIs, we don't need that. What do you think"? And Filip said, "Hold on, let me share my screen." He then pitched us the concept of VSF Next on the spot.

VSF Next interacts with backends through contracts, each integration responsible to implement the contract. These contracts are composable, so you can swap loading e.g. categories from a different backend than products. Or you can go and build your own composable. It sounded exciting, almost too good.



Vue Storefront

If you want to know more about **MACH**, contact us!

Patrick Friday
CEO at Vue Storefront
patrick@vuestorefront.io

Filip Rakowski
CTO at Vue Storefront
filip@vuestorefront.io