A Guide for Digital Leaders

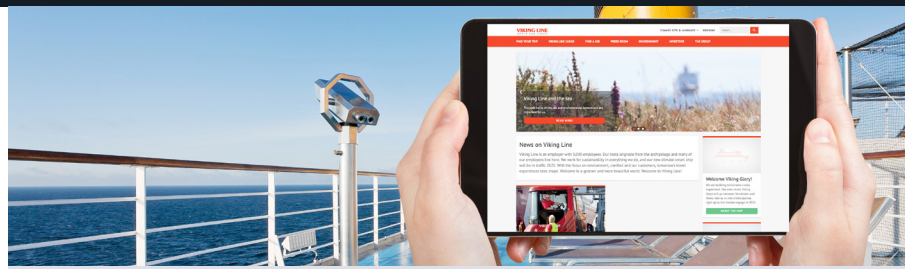
Headless CMS in the Enterprise

What to Consider Before Kicking off a Headless CMS Project in an Enterprise

ibexa

Table of Contents

Headless CMS
Introduction



Headless Content Management is a trend that has been around for a good while. Already a staple for boutique implementations for smaller companies and campaign sites, it continues to gain traction in enterprise projects too.

But before diving head-first into a headless CMS project you should consider how it fits your organization's needs.

Like a lot of IT jargon, the term 'Headless CMS' is not clear-cut. Before we begin, it is good to have a quick overview of some of the concepts used in the eBook:

- **Content API:** A tool used by editors to manage content which provides a raw content feed to be consumed by other development tools – not to users directly
- **Display layer:** A system (or systems) that render the feed provided by a Content API to be displayed to users. A website or a mobile app for example.

For both these components, there are a number of options on the market. To learn more about different Content API options, be sure to check out our Introduction to Content as a Service.

For display layers consuming content provided by the API, there the landscape is even more diverse, and you can go with a custom front end. See the sidebar for some popular technologies used for web front ends for Headless CMSs.

Together the combination of a content API and display layer result in a Headless CMS implementation. This differs from a typical, traditional CMS where the content API and display have been more coupled together forming a single system.

Both approaches have their advantages and challenges, some of which we will discuss in the following pages.

This eBook covers the considerations you should take into account before kicking off a Headless CMS project in an enterprise.

Technologies used for building a front end (display layer) for a Headless CMS:

Static site generators:

These generate a set of HTML files using a publishing process. High scalability at low cost. Very secure but can be limiting for dynamic features or where user level permissions need to be enforced for access to content

Popular products: Gatsby.js, Hugo, Jekyll

JavaScript server-side frameworks:

A web framework running on a server. Fetches content dynamically from the Content API. High flexibility for integrations, permissions etc. Requires a hosting environment and active maintenance.
Popular products: Express, Next.js, Koa

Function as a Service (FaaS):

A simplified framework that is fully managed. Pricing and performance scale with usage. The landscape for headless CMS front-end use is immature and requires some investment in custom development. A vendor lock-in situation is also something to consider.

Popular products: Azure Functions, AWS Lambda, Google Cloud, Open FaaS

Headless CMS in the Enterprise

From Killer Apps to Killer APIs

Today the Content Management industry is mature, meaning there exists a lot of accumulated experience and best practices. This in turn makes CMS implementation projects relatively humdrum routine work in the world of IT.

The challenges are in the business logic, whereas the technology itself might not seem very exciting to technologists. Riding the API economy hype wave, headless Content Management tools challenge the status quo – providing an alternative approach to delivering content.

APIs (Application Programming Interfaces) have been around since the beginning of computing. In the past, most APIs were internal ones used for developing software in close proximity: Applications running on Microsoft Windows use its APIs to interact with the hardware.

The recent excitement around APIs is mostly around distributed ones using web technologies. The headless CMS is a manifestation of this trend for Content Management.

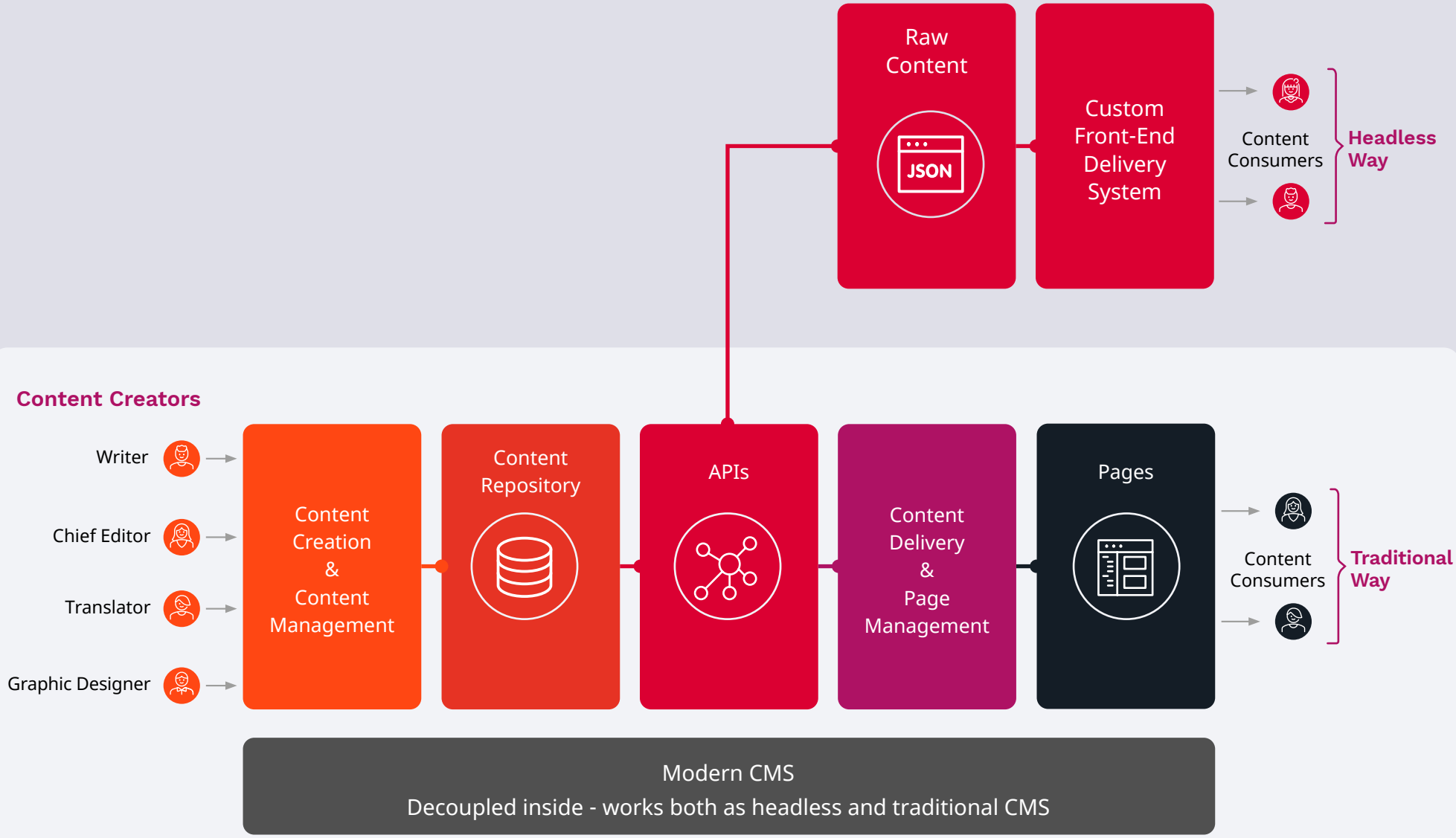
But just like an API in a desktop operating system, a web API in itself is just a tool. The value lies in the features and functionalities that can be built using their capabilities. Windows APIs enabled developers to create killer apps that allowed Microsoft to dominate computing for more than two decades.

With the arrival of the web and smartphones, the desktop operating system lost its dominant position. The company had to adopt and let go of its reliance on Windows to stay relevant and be a part of the new world introducing cloud services and subscription-based pricing for its Office productivity suite.

But things are rarely black and white. While desktop computing is certainly not a growth industry, there are areas where it continues to be the best option - from fields requiring real-time precision like making music to mundane tasks like writing blog posts and doing random productivity work. Personal Computer ownership has reached a tipping point and is in decline however the vast majority of office workers continue to wield a desktop or laptop computer every single day.

This story is an analogy to the Content Management space as well. It is easy for entrenched CMS vendors to point out the missing capabilities in pure headless systems, whereas CaaS providers' marketing teams labor to produce content underlining the deficiencies of "legacy CMS" products. In reality there is room for both approaches. It's a diverse field with different requirements and use cases.

At Ibexa, we don't see the need for the juxtaposition. Our product infrastructure allows us to build our internal APIs and then expose them through web APIs such as REST or GraphQL. You can use Ibexa Platform headless, full stack or something in between. The choice is yours.


Raw Content
JSON
Custom Front-End Delivery System
Content Consumers
Headless Way
Content Creators
Writer
Chief Editor
Translator
Graphic Designer
Content Creation & Content Management
Content Repository
APIs
Content Delivery & Page Management
Pages
Content Consumers
Traditional Way
Modern CMS
Decoupled inside - works both as headless and traditional CMS

User Needs and Product

A Headless CMS is primarily used for managing content

Today the term CMS (Content Management System) is used to describe a system whose function reaches well beyond creating, modifying and editing content.

These systems have outgrown their initial scope, and many are now on their way to transitioning from a CMS to a DXP (Digital Experience Platform). This topic was covered in-depth in a blog post titled From CMS to DXP to Deliver Digital Success.

A headless CMS is truer to its definition. It is a system primarily used for MANAGING CONTENT. Period. This makes it more approachable to users, but also less capable than a traditional enterprise CMS.

This ambivalence of what a CMS does can create some surprises during the implementation process. You might take for granted some features which are (and have always been) beyond the scope of a Content Management System.

When replacing an existing traditional CMS or DXP with a headless CMS configuration, an important task is to identify the tasks that your content editors, marketing staff and other users perform daily.

Map out any feature gaps between the two in key functionalities. In addition to content management needs, things like rich content models and workflows, be sure to map features for peripheral site management tasks such as:

- Managing navigation, redirects and other similar tasks
- Adding tracking scripts for marketing, usability tracking, etc.
- User management with your corporate user backend
- Scheduling promotions for weekends
- Modifying landing page structures
- Previewing content and a collection of content items.

All of the above can be achieved with either an integrated system or a combination of a content API and a display layer. The difference is that for the latter you might need to add third-party tools and additional integrations.

Don’t take product features for granted when you’re moving from an integrated software suite to a microservices architecture built from highly specialized individual components.

Implementation Complexity and Robustness

A combination of the right skills and technology are required

Because of their maturity, traditional CMS implementations of large projects are easier (not easy!) to estimate than a headless implementation project of similar scope. This is not due to the technology being better or worse, but because there are more established ways of working, expectations for what a system does and who does what. One can rely on past experience rather than guesswork for realistic estimations.

Today this is not often the case for complex headless implementations. The client is often doing this type of implementation for the very first time and the integration partner may not have sufficient experience.

This can lead up to some hiccups and unexpected costs during and after the project, even if you've done your due diligence on features as described in the second section in this eBook. Well managed, these are not an issue.

One key characteristic of a CMS project is that it is most likely a multivendor project using multiple technologies. While some CMS vendors do implementation projects themselves, it is quite rare for a CaaS (Content as a Service) provider to provide implementation services.

It is simply not something that is compatible with the business model of an API provider. Like with any technology project you should examine an implementation partner's experience and references using the content API of your choice. You don't want to be their first try at it.

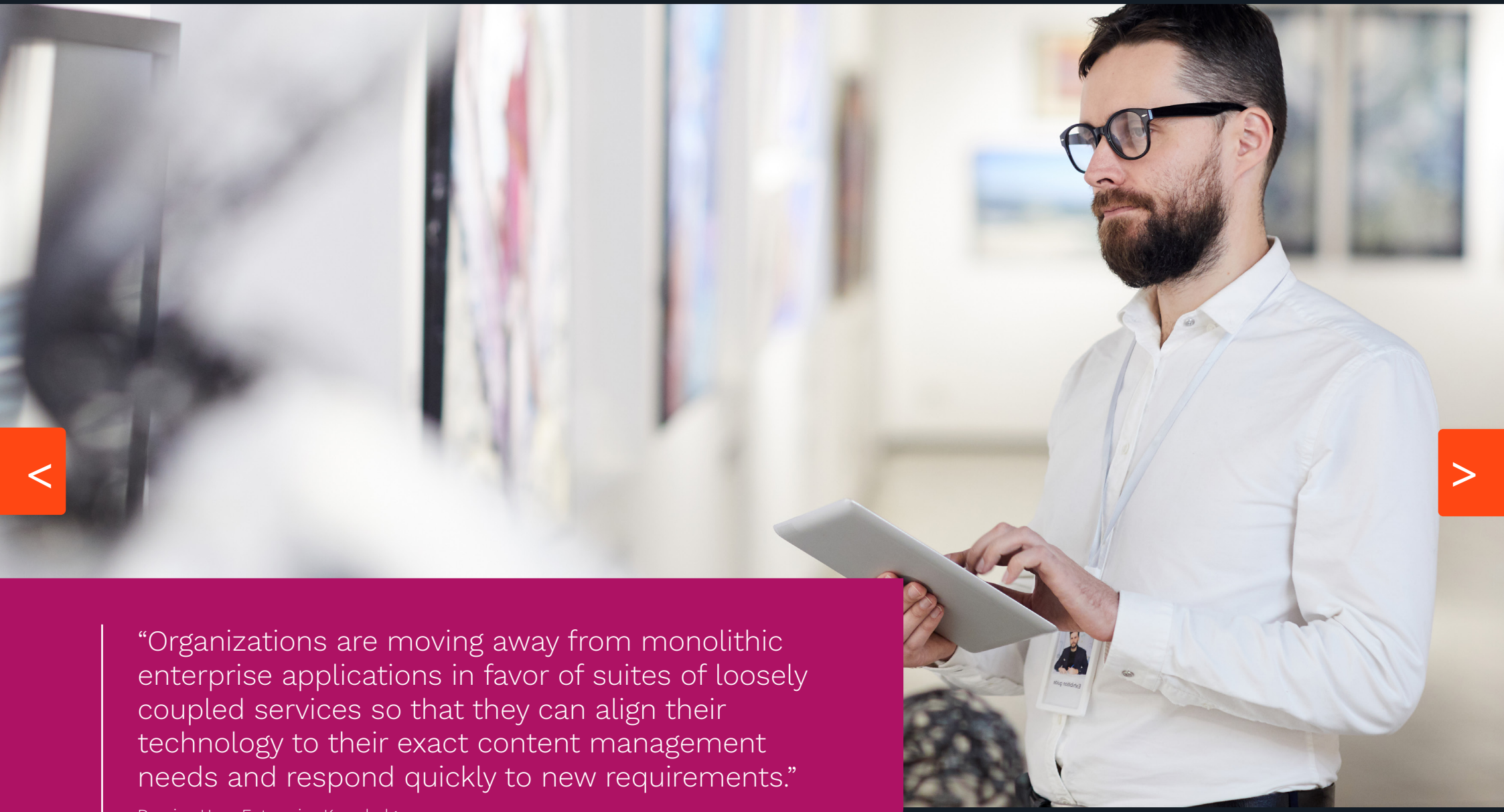
Integration architecture is another point to consider. Content Management Systems have traditionally been server-side systems, and it is natural that connections to external systems have been built on the server.

An emerging trend is moving integrations to the client – the browser. This trend is not directly related to whether you are using a headless CMS or not, but the two often coincide in the same projects because both are novel technologies.

JAMStack is one moniker given to the technical architectural approach where the browser becomes the integration platform.

The term comes from JavaScript, APIs and Markup (JAM). The core idea is to shop around for best-of-breed API providers for authentication, e-commerce, payments and so forth and integrate them in the browser, freeing you from running server-side systems.

At best a JAMStack implementation allows you flexibility to choose the best technology at low cost, at worst it's a fragile mess of services that can be time-consuming to figure out when there is trouble. In reality, the average experience is likely somewhere in between.



> “Organizations are moving away from monolithic enterprise applications in favor of suites of loosely coupled services so that they can align their technology to their exact content management needs and respond quickly to new requirements.”
>
> Demian Hess Enterprise Knowledge

Project and Long-Term Maintenance Costs

Pay attention to the total cost of ownership and budget for growth

The cost of a headless CMS implementation boils down to the cost of the Content API and the API Consumer(s). The total cost is accrued during the initial development phase and over the years of development and maintenance.

The Content API is easier to figure out, since you'll need to consider the available options on the market and choose the one that gives you the capacity and required features at the most competitive price. A game of numbers, pluses and minuses.

When choosing a content API be sure to take into account at least the following points:

- Required capacity (number of content items, repositories, editors...)
- Available features vs. custom development
- Hosting cost (for CMS vendors)
- Subscription cost (for CaaS)

Here you need to find the solution with the best balance for your individual needs.

A CaaS provider's pricing might be very attractive at low volume but may skyrocket at the scale your organization needs. An open source systems' lack of licensing fees can be offset by higher cost for hosting. Be sure to calculate future expenses if your desire is growth.

A PaaS like Ibexa Cloud on the other hand could provide a cost benefit over IaaS providers by reducing the personnel cost of running your own internal DevOps team.

With a content API provider chosen, it's time to consider the display layer. This is an area where there is literally an unlimited of options available.

From completely tailored boutique builds to product-based implementations based on frameworks like Next.js or Nuxt.js to static site generators like Gatsby. js or Hugo. Virtually all of these tools are open source, so the cost is based on labor. Again, look for experience when selecting an implementation partner, as it yields higher quality and higher development velocity.

For long-term cost, things can become tricky. The content API costs are subject to change – generally prices are usually quite stable - but technically a CaaS provider could hike prices literally overnight based on the subscription terms that you've agreed on.

However, this is quite unlikely because there are plenty of alternatives and moving between providers is relatively trivial. You will need some reworking of your implementation, but a well-architected content display layer, "the head", can accommodate this without introducing prohibitive cost.

Continuously Architect for Change

Reap the benefits of a microservices environment

In a headless CMS solution, the display layer (API Consumer) becomes a strategic choice. When deploying a traditional CMS, the templates and front-end implementation are often disposable.

In enterprise implementations where a Web Content Management System (WCMS) can be live for five to 10 years after the initial implementation project, the design undergoes multiple clean slate refreshes. In the backoffice features keep adding up.

In an integrated end-to-end content management solution, the content API and display layer are by nature tightly coupled together - they're a part of the package.

One of the key value proposals of the headless architecture is that the two are independent. But there is a natural tendency for feature investments to accumulate to the display layer, making the original goal of being independent to swap out either of the components a moot point.

If you couple your implementation tightly to a given design layer, you can end up with rising cost. Over time as your display layer accumulates technical debt, which in turn requires increased development efforts.

Your capability to evolve is hindered and a heavy rewrite might be in the cards. It is easy to live in an illusion of having a decoupled system simply because of the high-level architecture you chose initially. Staying agile requires effort.

Software is by nature bound to be obsolete sooner or later but you can architect a system so that parts are replaceable.

The key to maintaining the flexibility of a headless implementation is to consciously develop functionalities to be disposable.

Split complex capabilities into reusable services that are not tied to any given display layer. This will allow you to reap the benefits of the decoupled approach even over years of development.

> Ibexa was implemented as the content management solution for the onboard digital platform for Viking Lines. Content from Ibexa is displayed on-board Viking Line ships for mobile app, browser UI, in-cabin TV's and digital signage screens.

Conclusion

It’s essential to invest in a DXP with strong content capabilities and open APIs

Today there are more options than ever for a technical implementation of an enterprise IT solution. The headless approach can be a breath of fresh air to someone working with a cumbersome system whose technical service life is way beyond its best-before date, but it is not necessarily always the best match.

The abundance of options can make it more difficult to choose the right option, and it can be easy to choose something that is not the ideal match. For example, if you do pure multichannel content publishing, you have no need for site management then headless CMS implementation is a perfect fit. Don’t pay for features you don’t need.

At the same time keep in mind that having a new option available does not render existing tools obsolete. While features do become genuinely obsolete, there are functionalities that are essential — a sort of necessary evil — to have in a content management project implementation.

For these needs you can use features from integrated offerings like DXPs or go with a pure headless CMS and integrating third party tools to fill in the gaps. In large projects adding more systems, partners and systems add up and can start consuming the original perceived benefits.

Instead of being able to introduce changes rapidly and at lower cost, you could spend that same time and resources on coordinating multiple system vendors and paying fees to fragmented service providers. This is especially true when you consider the total cost of ownership of a solution over its lifetime.

Remember that no technology guarantees success. The team and their collaboration make or break your project implementation. This is true for small and large efforts, but the odds of success tend to fall the more people and technologies you introduce. A very successful approach for a team of five can yield opposite results when your team size is thirty.

If you’d like to know more about Headless CMS solutions, or if you’re currently in the midst of an evaluation project, why not request a demo with our Digital Experience experts or to discuss your requirements for successful digital transformation in your organization?

About Ibexa

Ibexa's Digital Experience Platform (DXP) enables organizations to stay competitive, optimize their revenue streams, launch new products and services and test-drive digital strategies. With Ibexa's DXP, B2B companies can transform sales strategies and build frictionless buying experiences in a sustainable, cost-effective and fast manner.

Offering content creation, website building, e-commerce and personalization along with accelerated development capabilities, Ibexa helps companies to quickly embrace new business and sales strategies and build memorable experiences for their users across channels, unifying brand, content, product information, commerce and their existing business processes. With the help of our certified, trustworthy partners – digital agencies, system integrators and consultancies – they will progress in their digital transformation and quickly adapt to changing market needs while always putting the customer experience at the heart of their businesses.

Ibexa's reliable and secure DXP is trusted by thousands of users worldwide and a dedicated global partner ecosystem. Brands such as Crédit Agricole, Comscore, Hibu, Groupe Atlantic and Whirlpool rely on Ibexa as a strategic partner in their business. Founded in 1999, Ibexa is headquartered in Oslo, Norway and has offices in Germany, France, the UK, Spain, Poland, USA and Japan.

Resources

- Explore Ibexa software: www.ibexa.co/products
- Request a demo: www.ibexa.co/demo
- Read success stories: www.ibexa.co/success-stories
- Get the latest Ibexa news and insights: www.ibexa.co/blog
- Read our software information and documentation: www.ibexa.co/software-information
- Find an Ibexa partner: www.ibexa.co/partners

Want to Know More? Please Get in Touch

www.ibexa.co/contact

www.ibexa.co