Elaboración de productos de información con *JAMstack*: del sistema de gestión de contenidos al web estático

Development of information products with *JAMstack*: from the content management system to the static web

Jesús Tramullas

Tramullas, Jesús (2020). "Elaboración de productos de información con *JAMstack*: del sistema de gestión de contenidos al web estático". *Anuario ThinkEPI*, v. 14, e14f05.

https://doi.org/10.3145/thinkepi.2020.e14f05

Publicado en *IweTel* el 12 de mayo de 2020

Jesús Tramullas
https://orcid.org/0000-0002-5374-9993
Universidad de Zaragoza
Departamento de Ciencias de la Documentación
C/ Pedro Cerbuna, 12. 50009 Zaragoza, España
https://tramullas.com
tramullas@unizar.es

Resumen: La evolución de los sistemas de gestión de contenidos ha propiciado la aparición de nuevas arquitecturas técnicas. Entre los desarrollos más novedosos destaca la aparición y consolidación de los generadores de web estático, y su integración en entornos de programación de aplicaciones web gracias a las *JAMstack*, pilas formadas por *JavaScript*, API y *markups* (*markup languages*, o lenguajes de marcado).

Palabras clave: *JAMstacks*; Gestión de contenidos; *Headless CMS*, Generadores de web estático.

Abstract: The evolution of content management systems has led to the emergence of new technical architectures. Among the newest developments stands out the appearance and consolidation of static web generators, and their integration into web application programming environments thanks to *JAMstacks*, stacks formed by *JavaScript*, API and markup languages.

Keywords: *JAMstacks*; Content management; Headless CMS; Static site generators.

1. El dominio de los sistemas de gestión de contenidos

Durante la segunda mitad de la década de 1990, en plena expansión de internet y sus servicios, los sistemas de gestión de contenidos (*Content Management Systems*, CMS) se convirtieron en la herramienta clave para el desarrollo y la implementación de productos de información digital en la Red. Hubo que esperar a los inicios de la década siguiente para que los CMS se popularizaran a todos los niveles, gracias a la publicación, bajo licencias de software libre, de aplicaciones como los desaparecidos PHP *Nuke* y *Postnuke* (2001), *Zope* (1999), *Plone* (2001), *Drupal* (2001) o *WordPress* (2003). El auge de la denominada web 2.0, con sus promesas de participación e interacción, y el bajo coste de un buen número de proveedores de *hosting*, facilitaron que todo tipo de organizaciones adoptaran los CMS

como base para su presencia en internet. Blogs, portales, wikis, gestores de colecciones, gestores de documentos, plataformas colaborativas… sirvieron para instalar y operar una amplia gama de servicios de información digital (**Tramullas**; **Garrido-Picazo**, 2006a).

La arquitectura básica de los sistemas de gestión de contenidos, establecida en aquellos momentos, ha permanecido casi invariable hasta la actualidad. La base se identificaba, y se identifica todavía, con el acrónimo LAMP, que correspondía casi invariablemente a la combinación de *Linux*, *Apache*, *MySQL* y *PHP* o *Perl*. Sobre un servidor de espacio dedicado o compartido, la combinación del software para servidor web *Apache*, del sistema de gestión de bases de datos *MySQL*, y de intérpretes de lenguajes *PHP* y *Perl*, permitía instalar y operar casi cualquier tipo de servicio o de producto de información digital (**Eíto-Brun**, 2013; **Barker**, 2015). Los sistemas de gestión de contenidos instalados sobre estas bases han respondido a una arquitectura casi estandarizada, en la cual pueden identificarse tres subsistemas principales, correspondientes a los bloques de procesos de gestión y administración, de colección y de publicación (**Tramullas**; **Garrido-Picazo**, 2006b).

Como en otros campos de la gestión de información digital, las herramientas precedieron a las formulaciones teóricas y a la mejora y afinamiento de métodos y técnicas. La gestión de contenidos se convertía, de manera empírica, en una disciplina cuyo objetivo era el diseño y publicación de contenidos ajustados a las necesidades informacionales de una organización o de un conjunto específico de usuarios (**Pérez-Montoro**, 2008). En la misma se conjugaban métodos y técnicas de gestión de proyectos, análisis de usuarios, arquitectura de la información, definición de políticas y productos de información… hasta dar forma a una estrategia de contenidos (**Meghan**, 2015; **Halvorson**; **Rach**, 2012). La aparición y desarrollo de la noción de *Enterprise Content Management* (ECM), auspiciada por la AIIM, generó un marco de referencia para las organizaciones, dentro del cual se podía llevar a cabo una gestión integral de la información que, atendiendo a los objetivos y planes de las organizaciones, estaba basada en la gestión de contenidos y en los sistemas que la hacían posible.

A mediados de la década de 2000, la disponibilidad de sistemas de gestión de contenidos de todo tipo, tanto con licencias privativas como de software libre, y la abundancia de proveedores de *hosting*, con costes asequibles para individuos y organizaciones de diferentes enfoques y clases, hicieron posible una gran expansión de la publicación en internet. El mercado y las tecnologías habían madurado, y los sistemas de gestión de contenido se consideraban la solución ideal para la gestión y publicación de información, desbancando a las clásicas páginas web estáticas, escritas en código HTML. El dominio de los sistemas de gestión de contenidos durante las dos últimas décadas ha resultado casi incontestable.

2. Límites y problemas de los sistemas de gestión de contenidos

El éxito de los CMS no debe ocultar que, como en cualquier entorno o sistema sociotécnico, hay factores, internos y/o externos, que limitan las posibilidades y el desarrollo de un producto o servicio (**Tramullas**, 2015). El mercado y las tecnologías han madurado, de la misma forma que ha sido posible identificar contextos en los que la utilización de un CMS no es la solución óptima. La evolución del mercado de soluciones muestra una consolidación alrededor de un grupo limitado de actores en la ECM. En el campo de la gestión de contenidos web, aunque las opciones son más numerosas, los datos disponibles también señalan hacia la concentración de actores (véase el caso de *Wordpress*). En lo tecnológico, la evolución de las sucesivas versiones de los CMS ha traído como consecuencia el aumento de los requerimientos de base para su implementación y ejecución. La casi desaparición de *Perl*, y el aumento del uso de lenguajes como *Java*, *Ruby on Rails* y *Python*, demandan la existencia de proveedores con una mayor carta de servicios, incluso la necesidad de servidores dedicados, lo que trae como consecuencia un aumento de los costes en hosting. Tras la crisis de 2008 y los recortes económicos posteriores, los recursos disponibles en muchas organizaciones han sido más limitados.

El aumento de prestaciones de los CMS ha hecho más complicada su gestión. Si bien en sus primeros momentos se pretendía simplificar y democratizar los procesos de publicación en internet, el aumento de la complejidad de los sistemas, la proliferación de plataformas y canales de publicación, y el aumento de fuentes de información externas de las que es necesario integrar información han limitado las posibilidades de los CMS que se podrían considerar "tradicionales". La falta de flexibilidad ha sido señalada como uno de los problemas de esta generación de CMS. Ahora las organizaciones requieren mayor simplicidad de las herramientas y elevada velocidad de despliegue y respuesta de los productos y servicios creados.

"El éxito de los CMS no debe ocultar que, como en cualquier entorno o sistema sociotécnico, hay factores, internos y/o externos, que limitan las posibilidades y el desarrollo de un producto o servicio"

A ello hay que unir factores de coste oculto, que no fueron adecuadamente valorados en los procesos de diseño y lanzamiento de un buen número de servicios a mediados de la década de 2000. Los CMS requieren de una vigilancia y mantenimiento constantes, de procesos de actualización o de migración, de optimización de rendimiento, y de controles para evitar o minimizar posibles problemas de seguridad. A pesar de una aparente simplicidad, los CMS tienen curvas de aprendizaje, tanto a nivel de administración, como en formación de usuarios, que pueden llegar a ser realmente elevadas.

3. La evolución de los CMS: *headless*, *decoupled* o *hybrid* y otras bestias...

El modelo tradicional o "monolítico" de CMS incorpora las capas de *frontend* y de *backend*:

- el *frontend* se encarga de todas las tareas de definición y de transformación para la presentación de los datos y contenidos al usuario;
- el *backend* ofrece las interfaces necesarias de gestión de contenidos y de administración. En este caso se habla de un CMS "acoplado", ya que ambas capas están integradas en la misma aplicación.

Esto adolece de los problemas indicados en el apartado anterior, y para superar sus limitaciones se han desarrollado tres nuevos modelos de arquitectura: el *headless CMS,* el *decoupled/hybrid CMS*, y el *cloud CMS* (**Heslop**, 2018; **Cuesta**, 2019; **Winkels**, 2019).

- El *headless CMS* es un modelo de arquitectura en el cual no existe un *frontend* como tal, ya que este es sustituido por un conjunto de aplicaciones diferentes, que intercambian la información con el CMS y la presentan en diferentes plataformas, siendo todo ello gestionado desde un *backend* integrador. Las aplicaciones utilizan las API del CMS para comunicarse con este y acceder a los datos necesarios para cumplir sus funciones. Como ventaja destaca la capacidad de desarrollar aplicaciones específicas, que no requieran todo el potencial de un CMS clásico para sus funciones; como inconveniente, requieren una mayor inversión en desarrollo y en tecnologías. En los últimos años se ha producido un notable aumento de plataformas de *headless CMS* (**Ismail**, 2020; **Kothari**, 2020).
- Los *decoupled/hybrid CMS* ofrecen un modelo mixto. Se trata de un CMS "desacoplado" que ofrece tanto el *frontend* clásico, como un conjunto de API que permiten utilizar sus contenidos en aplicaciones independientes. La gestión de todo el contenido se lleva a cabo desde el *backend* propio del CMS. Esta aproximación potencia el enfoque de "crear una vez, publicar en muchos lugares". Se trata de una solución intermedia que intenta ofrecer lo mejor de ambas aproximaciones (**Ottervig**, 2020), pero que también tiene los problemas de ambas. Esta sería la estrategia adoptada por *Drupal* para entrar en estos modelos de gestión de contenidos (**Mayekar**, 2017; **So**, 2018).
- Los *cloud CMS* corresponden a servicios preparados y disponibles en la nube, que se hacen cargo del *backend* y del almacenamiento de contenido, ofreciendo a sus usuarios un conjunto ya preparado de API para que desarrollen sus *frontend* según sus necesidades. Son altamente personalizables, pero la capacidad de administración del contenido es limitada.

La clave de estas arquitecturas es la utilización de las API (*Application Programming Interface*) (*altexsoft*, 2019). La implementación y la utilización de las API para hacer posible la interacción con otros sistemas ha sido integrada en los principales CMS desde hace ya tiempo. Estas API responden al modelo CRUD (*Create, Read, Delete, Update*), y mediante las mismas se intercambian datos estructurados y etiquetados en XML o en JSON. *Drupal* ofrece desde sus primeras versiones diferentes API para desarrolladores que facilitan el acceso a sus servicios y contenidos.
https://api.drupal.org/api/drupal

"La clave de estas arquitecturas es la utilización de las API (*Application Programming Interface*), que responden al modelo CRUD (*Create, Read, Delete, Update*), y mediante las cuales se intercambian datos estructurados y etiquetados en XML o en JSON"

Wordpress también incluye diversas API, entre las cuales hay que señalar la *WordPress REST API*, que es la usada por otras aplicaciones como interfaz para enviar y recibir datos, estructurados en JSON, desde o hacia una instalación de *WordPress*.
https://codex.wordpress.org/WordPress_APIs
https://developer.wordpress.org/rest-api

Esta configuración conforma lo que se ha denominado *Headless Wordpress* (**Tellado**, 2019). En *Drupal* la generación de REST API es más compleja y requiere de módulos y diseño específicos (**Melgar**, 2019).

4. De vuelta a los generadores de web estático

Los nuevos modelos de arquitectura para la gestión de contenidos presentan una estructura más compleja, y requieren de habilidades y competencias en programación para poder desarrollar su potencial y crear productos de información. Sin perder esto de vista, también resulta evidente que existen situaciones en las cuales determinados productos de información no requieren de las prestaciones que ofrecen los sistemas de gestión de contenidos. La codificación de documentos mediante HTML puede ser suficiente en contextos en los que no sea necesaria una actualización continua de contenido. Sin embargo, si se atiende al principio citado previamente de "crear una vez, publicar en muchos lugares", HTML no es esquema de codificación que se preste a un proceso de edición ágil.

La necesidad de generar documentos web de forma rápida y sencilla ha llevado a la aparición de herramientas generadoras de páginas web estáticas. Derivados principalmente del campo de la elaboración de documentación de software, se basan en la utilización de lenguajes de etiquetado ligero, como *MarkDown* o *reStructuredText*, que luego son procesados por generadores como *Sphinx* o *MkDocs* (**Tramullas**, 2019). El resultado se envía a un espacio de *hosting*, en el cual se dispone inmediatamente del contenido generado. Si se trabaja con repositorios como *GitHub* o *GitLab*, existen servicios que conectan con el contenido de los proyectos en estos repositorios y generan las páginas HTML tomando como fuente los ficheros etiquetados, como en el caso de *Read the Docs*, por ejemplo.
https://docs.readthedocs.io/en/stable/intro/import-guide.html

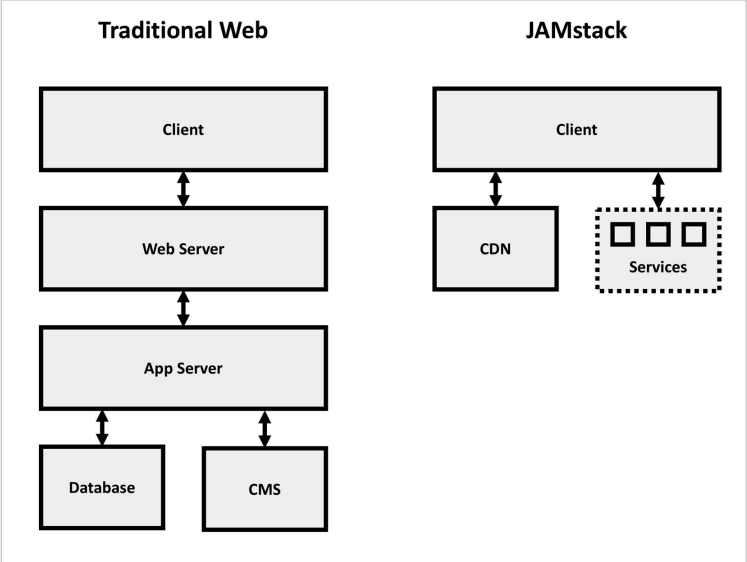

Figura 1. Arquitectura basada en CMS frente a arquitectura *JAMstack*
https://www.lifeintech.com/2017/12/20/jamstack

La velocidad de respuesta de los servidores al enviar páginas HTML estáticas es mucho mayor que al interactuar con un CMS. La gestión de las modificaciones en estructura y contenido del conjunto de documentos, en un entorno local de producción, también es más ágil y sencilla, ya que es el generador el que se encarga de actualizar y regenerar la estructura de acuerdo con los cambios introducidos. Si a esto se acompaña el control de versiones y un cliente de sincronización con repositorios, se dispone de un entorno completo para la elaboración de documentos para internet.

5. El desarrollo de las *JAMstack*

En el amplio campo de los diferentes tipos y aproximaciones que se ofrecen para la gestión de contenidos están adquiriendo una creciente importancia las denominadas *JAMstack*.
https://jamstack.org

Netlify o *Gatsby* aparecen en 2018, precisamente el año en el que se celebra la primera conferencia sobre *JAMstack*.
https://jamstackconf.com

No se trata de una herramienta o de un lenguaje de programación: es una arquitectura de desarrollo web. Su definición más aceptada es la propuesta por Mathias Billman, para el cual son

> "una moderna arquitectura de desarrollo web basada en cliente JavaScript, API reutilizables y marcado preconstruido" (**Ayodeji**, 2019).

Se trata de un entorno en el que no existe un servidor, en el sentido tradicional que se ha acostumbrado a entender desde el inicio de internet. Las *JAMstack* son la combinación de código escrito en *JavaScript*, diferentes API o microservicios reutilizables, y documentos cuyo contenido se ha etiquetado con lenguajes de marcado ligero (**Tramullas**, 2019).

El flujo de operación es simple (**Billman**; **Hawksworth**, 2019):

- Los ficheros con el contenido etiquetado y el código se almacenan en un repositorio, con funcionalidades de edición y modificación de los mismos. Esto quiere decir que las fuentes se encontrarán en un repositorio Git, como *GitHub* o *Gitlab*.
- Cuando se produce un cambio en los ficheros, se lanza un proceso de construcción del sitio, que da como resultado un conjunto de páginas web etiquetadas en HTML, y que integran contenido, datos, capa de presentación… trabajo que desarrollan las herramientas conocidas como generadores de sitios estáticos.
- El resultado se publica automáticamente en una CDN (*Content Delivery Network*), asegurando un rápido despliegue y la disponibilidad física de las páginas, eliminando tiempos de espera y latencias típicas de un CMS.

"*JAMstack*: una moderna arquitectora de desarrollo web basada en cliente *JavaScript*, API reutilizables y marcado preconstruido"

Es importante destacar que, en las *JAMstack*, al estar basadas en repositorios de código, se mantiene un control de versiones, lo que significa que se automatiza el histórico de cambios, y es posible recuperar y cambiar a versiones anteriores en cualquier momento. Además, al estar separada la capa de presentación, en plantillas y hojas de estilo, los cambios de esta, o del contenido que muestra, no afectan una a la otra, ni tampoco a las funcionalidades que se hayan integrado con *JavaScript*. Finalmente, hay que recordar que el código *JavaScript* se ejecuta en el navegador del usuario, lo que resulta más rápido, seguro y eficiente.

Sin embargo, esta pretendida simplicidad no debe llevar a pensar en un entorno sencillo de programación basado en un editor de código. Las plataformas de desarrollo se antojan más complicadas y especializadas. Los programadores utilizan de manera combinada diferentes herramientas (**Dionne**, 2020):

- Entornos de desarrollo (*frameworks*) para *JavaScript*, para facilitar la programación en este lenguaje, como *React*, *Vue* o *Angular*.
  https://reactjs.org
  https://vuejs.org
  https://angular.io
- Generadores de sitios estáticos, para la creación de las páginas web en HTML estático (**Candem**; **Rinaldi**, 2017), como *Hugo*, *Jekyll* o *Gatsby*. En *StaticGen* se encuentra disponible una completa lista, constantemente actualizada, de generadores de web estático.
  https://gohugo.io
  https://jekyllrb.com
  https://www.gatsbyjs.org
  https://www.staticgen.com
- Gestores de contenido *headless*, como *Contentful* o *Netlify CMS*, que ofrecen un *backend* para gestionar los flujos de trabajo de producción y despliegue de los productos creados. En *headless CMS* es posible consultar una lista actualizada de esta clase de CMS especiales para desarrollos *JAMstack*.
  https://www.contentful.com
  https://www.netlify.co
  https://headlesscms.org
- CDN, con el espacio de almacenamiento necesario para el despliegue del producto finalizado, específicas como *GitHub Pages*, *Vercel* o *Netlify*, o genéricas como *Cloudflare* o *Amazon CloudFront.*
- Funcionalidades de terceros, que permiten la interacción y la integración de datos y de servicios en las aplicaciones *JAMstack*, como búsqueda, pasarelas de pago, mensajería, identificación… que pueden ser a través de API o de SaaS.

Sin embargo, la aceleración que puede producir esta propuesta de desarrollo de productos web, con sus promesas de eficiencia y velocidad, no debe enmascarar los problemas inherentes a esta aproximación:

- la curva de aprendizaje es elevada, y requiere de habilidades de programación, por lo que la figura del desarrollador-programador será imprescindible en estos proyectos;
- las funcionalidades de interacción con fuentes de datos o de integración de servicios dependerán de terceras partes, lo que puede generar problemas de actualizaciones, coste o mantenimiento;
- no todos los productos se ajustan a las características de las *JAMstack*, ya que aquellos que requieren frecuentes actualizaciones y creación dinámica de nuevos contenidos, o interacciones complejas, son mejor servidos por las prestaciones y funcionalidades de un CMS "tradicional".

6. A modo de recapitulación

El auge que está teniendo el desarrollo web sobre *JAMstack* indica que esta arquitectura de producción de contenidos web va a estar presente y activa durante los próximos años. Resulta claro que su enfoque le da ventaja en determinados tipos de productos de información digital, frente al uso de un CMS clásico. Sin embargo, no debe obviarse la necesidad de contar con personal especializado en diversas herramientas de programación, lo que puede resultar un inconveniente en determinados entornos de actividad. Cabe poner en consideración que las *JAMstack* se engloban dentro del amplio campo de las metodologías de desarrollo ágil, lo que va a conllevar la necesidad de integrar perfiles de programación y de ingeniería de sistemas en los grupos de proyecto (**Norwood**, 2019).

Los generadores de sitios web estáticos ya han sido utilizados con éxito en la creación y publicación de productos de información basados en colecciones de documentos digitales, como por ejemplo *The Historical Topographic Maps Digitization Project*, en Ontario, que ha combinado *Jekyll*, *Hugo* y *GitHub* en la creación de una interfaz simplificada y de rápido acceso para material cartográfico (**Newson**, 2017).

También ha sido objeto de aplicación en los procesos de publicación de documentos académicos, como revistas, manuales o actas de congresos, por las bibliotecas de *Northwestern University*, mediante la utilización de *Jekyll* y *Bookdown* (**Diaz**, 2018). Las *University Libraries at Virginia Tech* han creado y utilizado una API propia para desarrollar nuevos servicios para profesionales y usuarios (**Bradley**, 2020). La utilización de *GitHub* y *GitLab* en el desarrollo de aplicaciones para bibliotecas se va a convertir en algo común (**Engwall**; **Roe**, 2020). Teniendo en cuenta el potencial de los generadores de sitios web estáticos y de las *JAMstack*, son numerosas las posibilidades que se abren para mejorar los procesos y los flujos de edición y publicación de contenidos en ámbitos especializados.

7. Referencias

Altexsoft (2019). "What is API: Definition, Types, Specifications, Documentation". *Altexsoft*, 18 junio.
https://www.altexsoft.com/blog/engineering/what-is-api-definition-types-specifications-documentation

Ayodeji, Bolaji (2019). "An introduction to the JAMstack: the architecture of the modern web". *freeCodeCamp*, 17 mayo.
https://www.freecodecamp.org/news/an-introduction-to-the-jamstack-the-architecture-of-the-modern-web-c4a0d128d9ca

Barker, Deanne (2015). *Web content management: Systems, features, and best practices*. Sebastopol, CA: O'Reilly. ISBN: 978 1491908129

Billman, Mathias; Hawksworth, Phil (2019). *Modern web development on the JAMstack: modern techniques for ultra fast sites and web applications*. Sebastopol, CA: O'Reilly Media. ISBN: 978 1 492 05854 0
https://www.netlify.com/pdf/oreilly-modern-web-development-on-the-jamstack.pdf

Bradley, Jonathan (2020). "How to use an API management platform to easily build local web Apps". *The Code-4Lib journal*, n. 48.
https://journal.code4lib.org/articles/15190

Candem, Raymond; Rinaldi, Brian (2017). *Working with static sites. Bringing the power of simplicity to modern sites*. Sebastopol, CA: O'Reilly Media. ISBN: 978 1491960943

Cuesta, Jesús (2019). "Evolución del CMS a CMS headless u otras opciones". *OpenExpo Europe*, 15 marzo.
https://openexpoeurope.com/es/evolucion-del-cms-a-cms-headless-u-otras-opciones

Diaz, Chris (2018). "Using static site generators for scholarly publications and open educational resources". *The Code4Lib journal*, n. 42.
https://journal.code4lib.org/articles/13861

Dionne, Mathieu (2020). "New to JAMstack? Everything you need to know to get started". *Snipcart*, 16 enero.
https://snipcart.com/blog/jamstack

Eíto-Brun, Ricardo (2013). *Gestión de contenidos*. Barcelona: Editorial UOC. ISBN: 978 8490297834

Engwall, Keith; Roe, Mitchell (2020). "Git and GitLab in library website change management workflows". *The Code4Lib journal*, n. 48.
https://journal.code4lib.org/articles/15250

Halvorson, Kristina; Rach, Melisa (2012). *Content strategy for the Web*. New Riders, 2ed. ISBN: 978 0321808301

Heslop, Brent (2018). "History of content management systems and rise of headless CMS". *ContentStack*, 18 diciembre.
https://www.contentstack.com/blog/all-about-headless/content-management-systems-history-and-headless-cms

Ismail, Kaya (2020). "34 headless CMS that should be on your radar". *CMS Wire*, 29 marzo.
https://www.cmswire.com/web-cms/13-headless-cmss-to-put-on-your-radar

Kothari, Abhishek (2020). "11 headless CMS to consider for modern application". *Geekflare*, 15 abril.
https://geekflare.com/headless-cms

Mayekar, Deepali (2017). *Decoupling Drupal. A decoupled design approach for web applications*. Berkeley, CA: Apress. ISBN: 978 1 4842 3321 4
https://doi.org/10.1007/978-1-4842-3321-4

Meghan, Casey (2015). *The content strategy toolkit: Methods, guidelines, and templates for getting content right*. New Riders. ISBN: 978 0134105109

Melgar, Julen (2019). "Creación de un API REST en Drupal e integración con Angular/IONIC". *Blog.Crecemos.Contigo, Hiberus Tecnología*, 19 noviembre.
https://www.hiberus.com/crecemos-contigo/creacion-de-un-api-rest-en-drupal-e-integracion-con-angular-ionic

Newson, Kaitlin (2017). "Tools and workflows for collaborating on static website projects". *The Code4Lib Journal*, n. 38.
https://journal.code4lib.org/articles/12779

Northwood, Chris (2019). *The full stack developer: Your essential guide to the everyday skills expected of a modern full stack web eeveloper*. Berkeley, CA: Apress. ISBN: 978 1 4842 4152 3
https://doi.org/10.1007/978-1-4842-4152-3

Ottervig, Vegard (2020). "Hybrid or headless CMS – what to choose?". *Enonic*, 10 febrero.
https://enonic.com/blog/hybrid-or-headless-where-to-go

Pérez-Montoro, Mario (2008). *Gestión del conocimiento en las organizaciones: fundamentos, metodología y praxis*. Gijón: Trea. ISBN: 978 84 9704 376 2
https://bit.ly/3gxqHZA

So, Preston (2018). *Decoupled Drupal in practice. Architect and implement decoupled Drupal architectures across the stack*. Berkeley, CA: Apress. ISBN: 978 1 4842 4072 4
https://doi.org/10.1007/978-1-4842-4072-4

Tellado, Fernando (2019). "¿Qué es eso de 'Headless WordPress' o 'Headless CMS' y para qué sirve?". *Ayuda WordPress*, 30 abril.
https://ayudawp.com/headless-wordpress-cms

Tramullas, Jesús (2015). "Gestión de contenidos, 2005-2015: una revisión". *Hipertext.net*, n. 13.
https://raco.cat/index.php/Hipertext/article/view/294025

Tramullas, Jesús (2019). "Desarrollos en elaboración de documentación técnica: lenguajes de marcado ligero". *Anuario ThinkEPI*, v. 13, e13f03.
https://doi.org/10.3145/thinkepi.2019.e13f03

Tramullas, Jesús; Garrido-Picazo, Piedad (coords.) (2006a). *Software libre para servicios de información digital*. Madrid: Pearson Prentice Hall. ISBN: 978 84 8322 299 7
http://pdfhumanidades.com/sites/default/files/apuntes/softlibrepservdeinfdig.pdf

Tramullas, Jesús; Garrido-Picazo, Piedad (2006b). "Los sistemas de gestión de contenidos". En: Tramullas, Jesús (coord.). *Tendencias en documentación digital*. Gijón: Trea, pp. 135-161. ISBN: 84 9704 270 0

Winkels, Niklas (2019). "The definitive guide to CMS architecture". *BloomReach*.
https://developers.bloomreach.com/blog/2019/cms-architecture.html