



DESIGN AND DEVELOP
CONTENT-DRIVEN WEBSITES
AND APPLICATIONS



Where Content Means Business

 #phpEz

eZ Platform, un CMS Headless

Madrid, @phpmad, 21/03/2018

Sobre mí



Carlos Revillo



<https://twitter.com/crevillo>



<https://github.com/crevillo>



<https://linkedin.com/in/carlos-revillo>



crevillo@gmail.com

carlos.revillo@the-cocktail.com



Trabajo en The Cocktail donde soy uno de los Team Leads del departamento de Tecnología.



Comencé con eZ Publish en Tanta, hace aprox. 10 años.



“Community member of the year” en dos ocasiones



Agenda

1

Acerca de eZ
Systems

2

eZ Publish, la primera
generación

3

eZ Platform, un
CMS *headless*

4

Casos de uso



Acerca de eZ Systems

eZ Systems, la empresa detrás de eZ Platform

eZ Systems se fundó en Noruega en el año 1999

Más de 15 años como **proveedor comercial** de open source.

Colabora con **más de 500 empresas** repartidas en más de 25 países.

ELLE



Cuentan con alrededor de **80 partners**, además de alrededor de **45.000 miembros** en su comunidad.

Links de interés


- ez.no
- ez.no/Resources/Case-Studies
- ezplatform.com
- share.ez.no
- ez-community-on-slack.herokuapp.com
- github.com/ezsystems
- doc.ezplatform.com



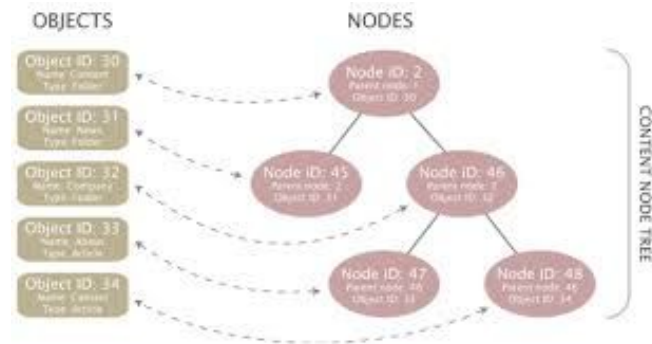
2 eZ Publish, la primera generación

eZ Publish incluía un modelo de datos muy flexible...

System Specific Elements		
Element	Value	
Content ID	13	
Name	"The penguin has a megaphone"	
Type	Article	
[...]	[...]	

Versions		
Version	Language	Attributes and content
1	 English	Title: The penguin has a megaphone
		Intro: First draft
		Body: Blah, blah, blah...
2	 English	Title: The penguin has a megaphone
		Intro: First draft
		Body: Blah, blah, blah...
	 Norwegian	Title: The penguin has a megaphone
		Intro: First draft
		Body: Blah, blah, blah...
	 Swedish	Title: The penguin has a megaphone
		Intro: First draft
		Body: Blah, blah, blah...

- ★ Define un modelo de datos que permite añadir tipos de contenidos sin modificar estructura de base de datos. No es un Active Record.
- ★ El modelo sigue siendo el mismo 15 años después.
- ★ Los contenidos se asocian a localizaciones, permitiendo relaciones “padre / hijo” entre ellos



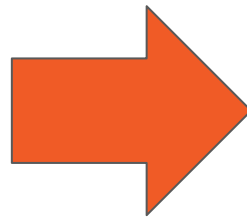
...pero con el tiempo descubrieron otra serie de problemas

- Como todos los proyectos PHP entonces, **sus componentes dependían los unos de los otros.**
- No había una **separación real entre contenido y presentación.** Las consultas a la Base de datos o al Solr se lanzaban desde las propias plantillas.
- Su curva de aprendizaje era **muy** alta.
- Dependía de un framework propio, y eso hacía difícil integrar otro tipo de librerías.

```
{def $nodes=fetch( 'content', 'tree',  
hash( 'parent_node_id', 42 ) )}  
  
{foreach $nodes as $node}  
{ $node.name|wash} <br />  
  
{/foreach}
```


No les quedó otra solución que buscar una alternativa

- Como todos los proyectos PHP entonces, **sus componentes dependían los unos de los otros.**
- No había una **separación real entre contenido y presentación.** Las consultas a la Base de datos o al Solr se lanzaban desde las propias plantillas.
- Su curva de aprendizaje era **muy** alta.
- Dependía de un framework propio, y eso hacía difícil integrar otro tipo de librerías.



Symfony

¿Por qué eligieron Symfony?

- Les permite mantener compatibilidad hacia atrás. Incluso se puede combinar el uso de Symfony controllers y sus módulos legacy.
- Twig permite a los equipos de front trabajar sin mucho conocimiento del API.
- Personas con experiencia en Symfony se sienten como en casa
- Permite e incluye integración sencilla con librerías y bundles que no tienen nada que ver con eZ.
 - Liip
 - Fos Http Cache
 - ...
- Mejora en rendimiento y mayor funcionalidad de comunicación con Reverse Proxies (Varnish), motores de búsqueda (Solr, Elastic) o trabajar con sistemas como Memcache o Redis.





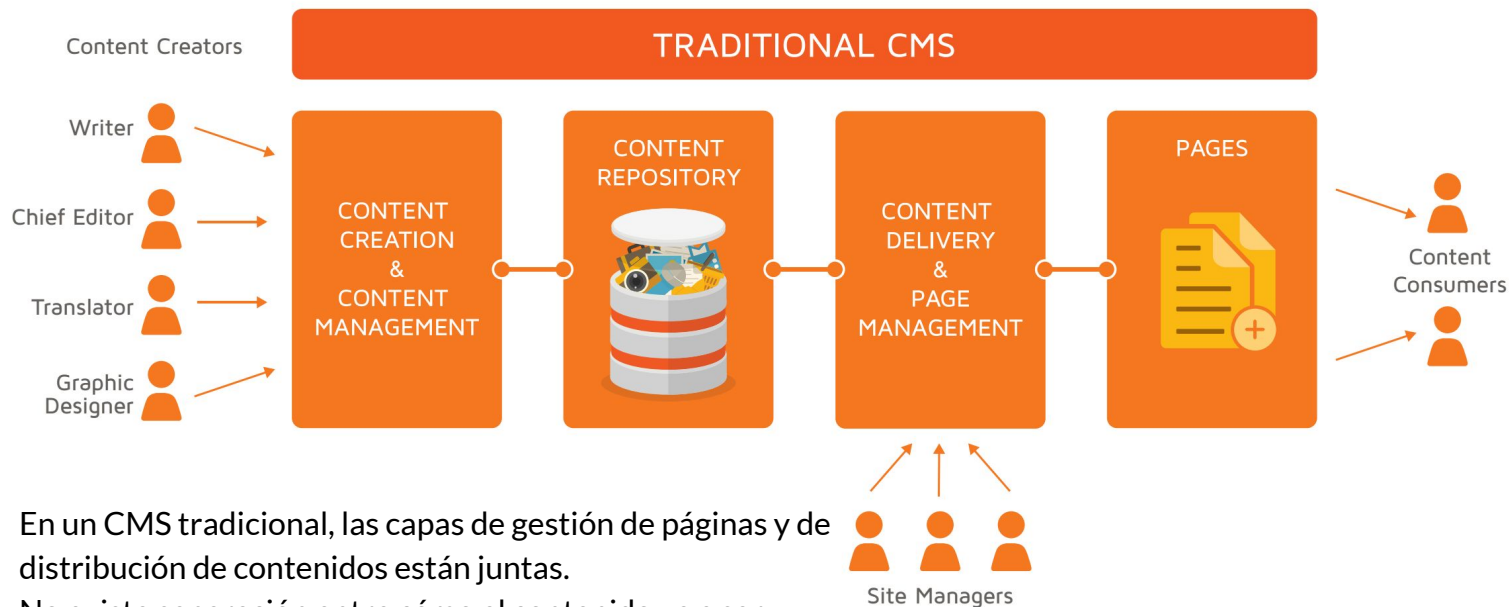
eZ Platform, un CMS Headless

Pero ¿qué es un CMS Headless?

- ★ Es un CMS que está enfocado en el funcionamiento del backend ofreciendo a los creadores de contenidos las herramientas necesarias para éstos estén listos para ser consumidos en un caso de CaaS.
- ★ Se enfoca en ayudar a los usuarios en la siguientes tareas
 - Modelar contenido
 - Crear contenido
 - Facilitar la colaboración entre los creadores de contenido.
 - Organizar el contenido de ese repositorio (content trees, taxonomías...)
- ★ Un CMS Headless no entra en cómo el contenido será distribuido o presentado a los usuarios finales



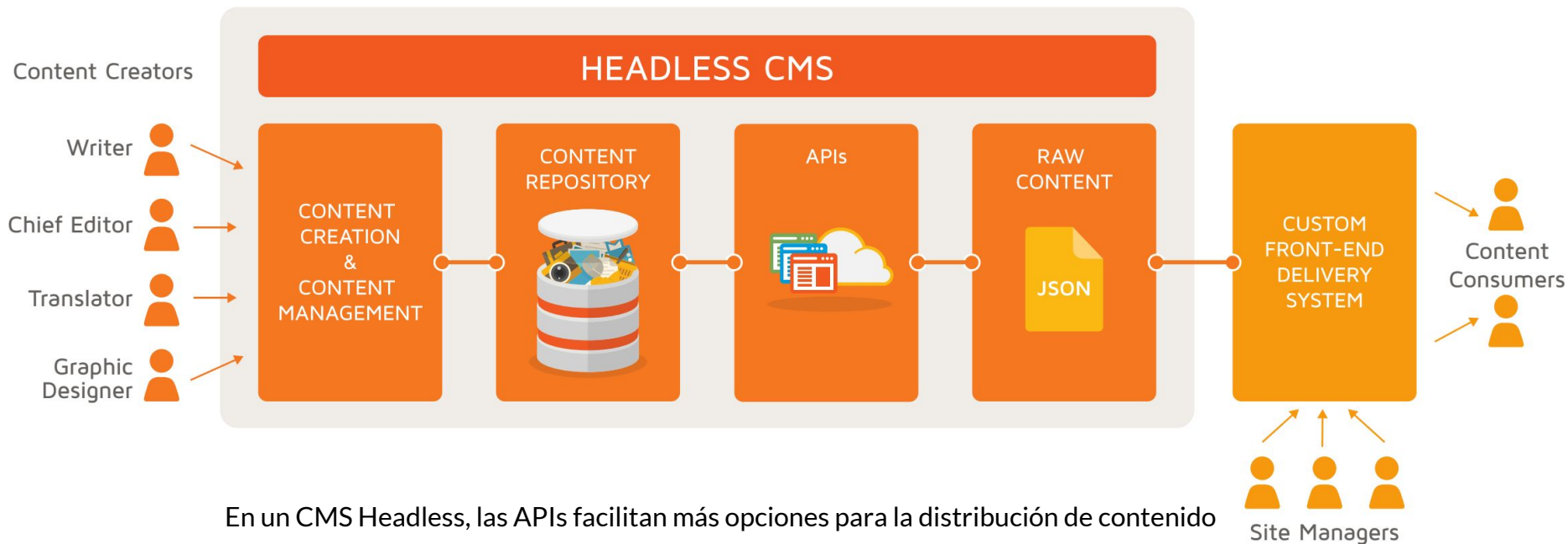
Los CMS tradicionales se centran en construir webs



En un CMS tradicional, las capas de gestión de páginas y de distribución de contenidos están juntas.

No existe separación entre cómo el contenido va a ser distribuido y presentado.

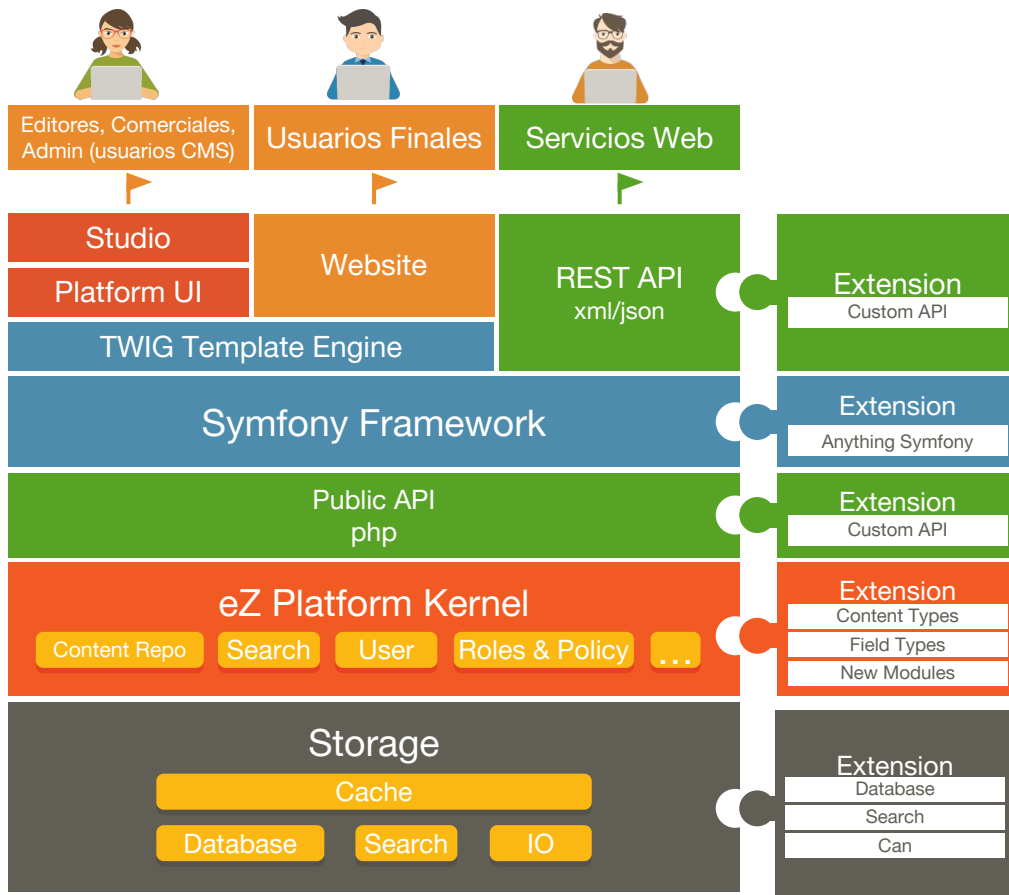
Los *Headless* se centran en el contenido y no en su destino



¿y por qué *Headless*?

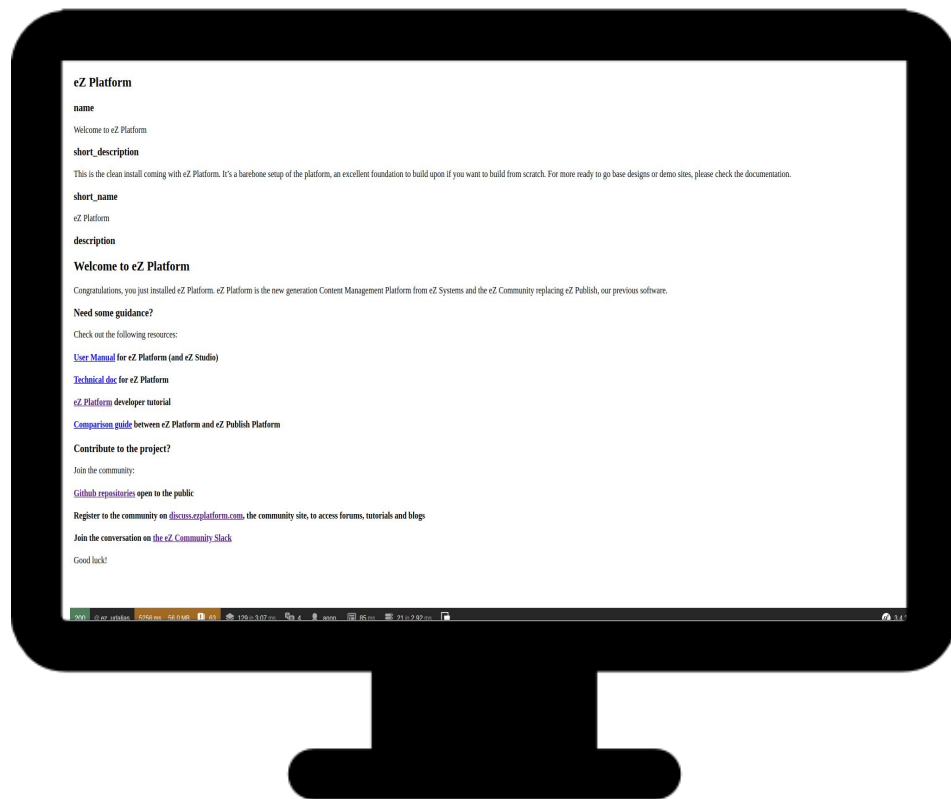


Para ser Headless, eZ definió esta arquitectura para eZ Platform



- Define una capa de negocio (kernel) que se expone a la aplicaciones a través de un API. (Public API)
- En el kernel se define el repositorio de contenido, la api de búsqueda, la gestión de usuarios y permisos.
- En la capa de almacenamiento, se añade un layer de cache entre el kernel y la bd o sistema de archivos
- Para construir nuestros sites, los desarrolladores interactuamos con la Public API.
- Para poder integrar eZ en otras aplicaciones, se expone una API Rest, que depende de la Public API.
- Para extender sus funcionalidades se usa Symfony y la Public API si esas funcionalidades necesitan acceder al contenido.
- Nota del autor: En 10 años trabajando con eZ no he necesitado una línea de SQL.

Decidieron olvidarse del CSS y JS.



- eZ Platform da libertad de elección a la hora de usar frameworks CSS o JS.
- Todas las urls son procesadas por un único controlador, independientemente del tipo de contenido que se quiera mostrar.
- Ese controlador manda una serie de variables a determinadas plantillas.
- En desarrollo, añade información específica de eZ a la barra de desarrollo de Symfony.

Simplemente incluyeron una serie de plantillas básicas...

El pagelayout

```
<!doctype html>
<html lang="{{ app.request.locale|replace({'_': '-'}) }}">
<head>
  <meta charset="utf-8">
  {% if content is defined and title is not defined %}
    {% set title = ez_content_name( content ) %}
  {% endif %}
  <title>{{ title|default( 'Home' ) }}</title>
  <meta name="generator" content="eZ Platform"/>
  {% if content is defined and content.contentInfo.mainLocationId %}
    <link rel="canonical" href="{{ path( 'ez_urlalias', {'locationId': content.contentInfo.mainLocationId} ) }}" />
  {% endif %}
</head>
<body>
{% block content %}{% endblock %}
</body>
</html>
```

- ★ El layout es configurable mediante settings
- ★ Desde cada plantilla sobre escribiremos el bloque content.
- ★ Podemos añadir tantos bloques como sea necesario.

...que apenas contienen una serie de tags...

presentando un contenido

```
{% extends noLayout == true ? viewbaseLayout : pagelayout %}
{% block content %}
  <h2>{{ ez_content_name(content) }}</h2>
  {% for field in content.fieldsByLanguage(language|default(null))-%}
    <h3>{{ field.fieldDefIdentifier }}</h3>
    {{ ez_render_field(content, field.fieldDefIdentifier) }}
  {% endfor %}
{% endblock %}
```

```
{% block ezstring_field %}
{% spaceless %}
  {% set field_value = field.value.text %}
  {{ block('simple_inline_field') }}
{% endspaceless %}
{% endblock %}

{% block eztext_field %}
{% spaceless %}
  {% set field_value = field.value.nl2br %}
  {{ block('simple_block_field') }}
{% endspaceless %}
{% endblock %}

{% block ezrichtext_field %}
  {% set field_value = field.value.xml|richtext_to_html5 -%}
  {{ block('simple_block_field') }}
{% endblock %}

{% block simple_inline_field %}
{% spaceless %}
  {% if field_value is not defined %}
    {% set field_value = field.value %}
  {% endif %}
  <span {{ block('field_attributes') }}>{{ field_value }}</span>
{% endspaceless %}
{% endblock %}
```

ez no guarda html sino xml. Así, a través de transformadores, podemos devolver html o cualquier otro formato.

...y añadieron mecanismos para que se pudieran customizar con plantillas propias...

Para customizar la presentación de los contenidos definimos mediante configuraciones las plantillas a usar.

```
content_view:
  full:
    home:
      template: "@ezdesign/full/home.html.twig"
      params:
        place_list_location_id: '%app.home.place_list_location_id%'
        tastes_location_id: '%app.home.tastes_location_id%'
      match:
        Id\Location: 2
        Identifier\ContentType: [folder]

    article:
      template: "@ezdesign/full/article.html.twig"
      match:
        Identifier\ContentType: [article]

  line:
    blog_post:
      template: "@ezdesign/line/blog_post.html.twig"
      match:
        Identifier\ContentType: [blog_post]
```

Para decidir qué plantilla ha de usar, se definen una o varias condiciones que han de ser cumplidas para que esa plantilla se use.

Las plantillas se pueden sobrescribir por tipo de vista (full, line, foo, bar...)

Nota: @ezdesign es una notación especial, útil para multisite. Ver en el apartado dedicado a ello.

article.html.twig

```
<div class="meta">
  <span class="field-author"> {{ ez_field_value(content, 'author') }} </span>
  <span class="field-date"> <time pubdate datetime="{{ ez_field_value(content,
'publish_date')|date("Y-m-d") }}">{{ ez_field_value(content,
'publish_date')|date("d M Y") }}</time></span>
</div>
```

...o con controladores de Symfony específicos para cada caso.

```
content_view:
  full:
    subscribe:
      controller: app.controller.contactform:showContactFormAction
      template: "@ezdesign/full/subscribe.html.twig"
    match:
      Identifier\ContentType: [subscribe]

blog:
  controller: "ez_query:contentQueryAction"
  template: "@ezdesign/full/blog.html.twig"
  match:
    Identifier\ContentType: [blog]
  params:
    query:
      query_type: "AppBundle:Children"
      parameters:
        parent_location_id: "@=location.id"
      assign_results_to: children
```

```
public function showContactFormAction(View $view, Request $request)
{
    $form = $this->formFactory->create(ContactType::class);
    if ($request->isMethod('POST')) {
        // other logic
    }
    $view->addParameters([
        'form' => $form->createView(),
    ]);
    return $view;
}
```

Desde los controladores podemos añadir variables a las plantillas.

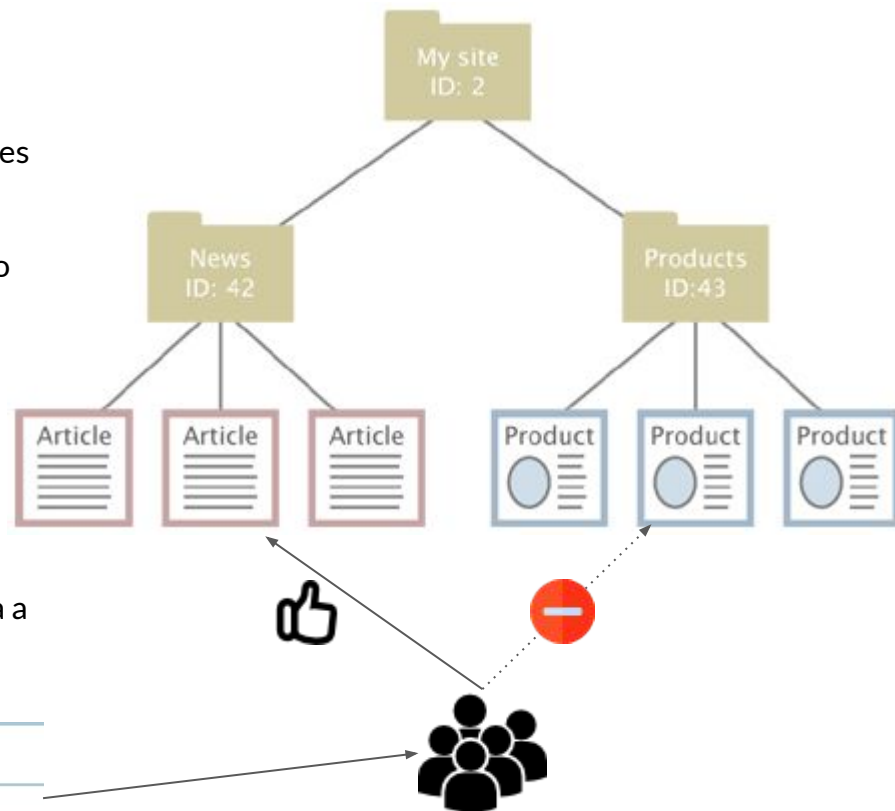
eZ ofrece un controlador que permite que ejecuta una con una query definida por el desarrollador y asigna resultados a una variable

https://doc.ezplatform.com/en/latest/guide/content_rendering/#querytype-example-latest-content

```
<div class="children">{* extract from blog.html.twig *}
  {% for child in children.searchHits %}
    <div class="container">
      {{ render_esi(controller('ez_content:viewAction', {
        locationId: child.valueObject.contentInfo.mainLocationId,
        contentId: child.valueObject.contentInfo.id, viewType: 'line'
      }))) }}
    </div>
  {% endfor %}
</div>
```

eZ Platform destaca por la granularidad en su modelo de permisos

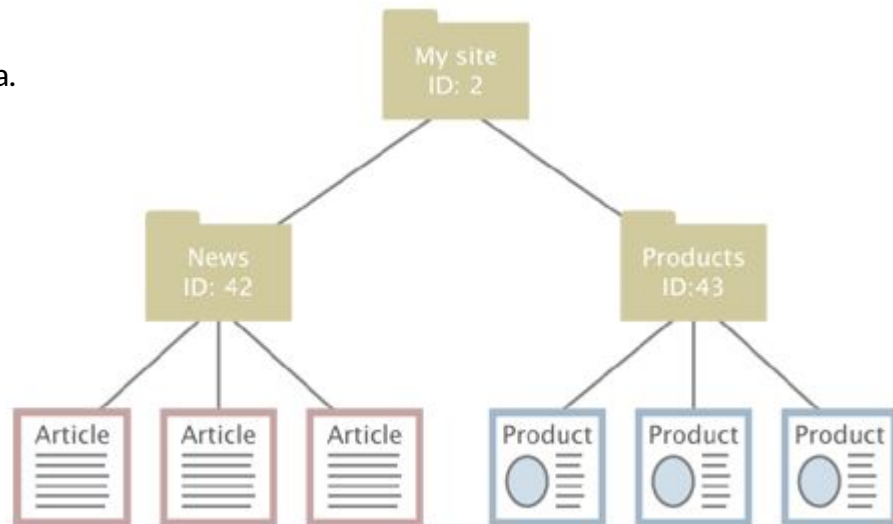
- Los usuarios también son contenidos.
- Normalmente organizados en “grupos de usuarios”, los cuales también son contenidos.
- Un rol define un conjunto de políticas para las que el usuario tendrá acceso.
- Una política estará definida por un módulo, una o varias funciones y, opcionalmente, una serie de limitaciones
- Los roles pueden ser asignados a grupos de usuario. Los usuarios heredan los permisos asignados a su grupo.
- Los roles pueden ser también asignados de forma específica a cada usuario



Module	Function	Limitations
Content	Create	Content Type: Article

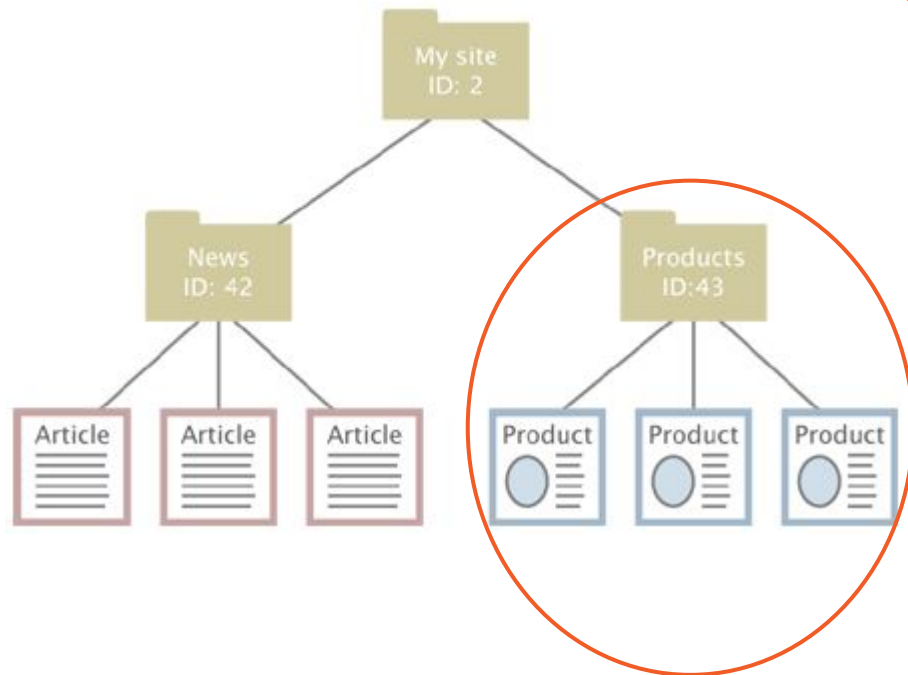
Todo en eZ es contenido y para encontrarlo usamos un API

- Actualmente, eZ Puede trabajar con 3 motores de búsqueda.
 - Base de datos
 - Solr
 - Elastic Search (experimental)
- Los 3 permiten búsqueda “fulltext” o búsqueda en base a criterios, así como ordenación mediante criterios.
- En eZ todo es contenido, por tanto, todo lo que hagas dependerá de ellos y de cómo están organizados.
 - Menús
 - Listados
 - ...
- Se puede cambiar de motor de búsqueda sin tener que cambiar el código que hayas desarrollado previamente.



Definiendo criterios y ordenaciones podemos obtener resultados

```
$query = new \eZ\Publish\API\Repository\Values\Content\Query();  
  
$query->criteria = new Criterion\LogicalAnd([  
    new Criterion\ParentLocationId(43),  
    new Criterion\ContentTypeIdentifier('product')  
]);  
  
$query->sortClauses = [  
    new SortClause\DatePublished(Query::SORT_DESC)  
];  
  
$query->offset = 10;  
$query->limit = 5;  
  
$result = $searchService->findContent($query);
```



eZ Platform pone mucho énfasis en el *performance*

- eZ Platform trabaja por defecto con el Symfony Reverse Proxy y se integra perfectamente con Varnish.
- Por defecto, cuando un contenido se crea o modifica, eZ lanza peticiones a Varnish para decirle que ha de expirar el caché de ciertos contenidos.
- Para que esto ocurra, varnish debe estar configurado con el VCL provisto por eZ
https://github.com/eZsystems/ezplatform/blob/master/doc/varnish/vcl/varnish4_xkey.vcl
- Este VCL permite que la misma versión cacheada de una página pueda ser servida a distintos usuarios.
- Tus controladores pueden setear un Vary para mostrar el mismo contenido a usuarios con las mismas políticas:

```
use Symfony\Component\HttpFoundation\Response;
```

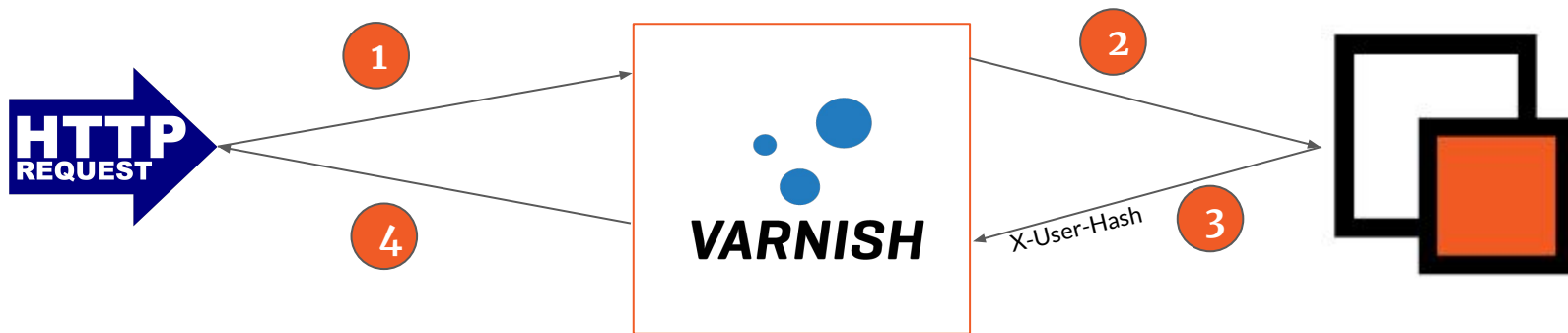
```
// En tu controlador
```

```
$response = new Response();
```

```
$response->setVary('X-User-Hash');
```

ofreciendo un archivo vcl especial

- Esta funcionalidad está basada en el artículo [Context aware HTTP caching](#).
- Más info en https://github.com/ezsystems/ezplatform-http-cache/blob/master/docs/fos_http_cache.



1 Varnish recibe la petición http (sin el X-User-Hash)

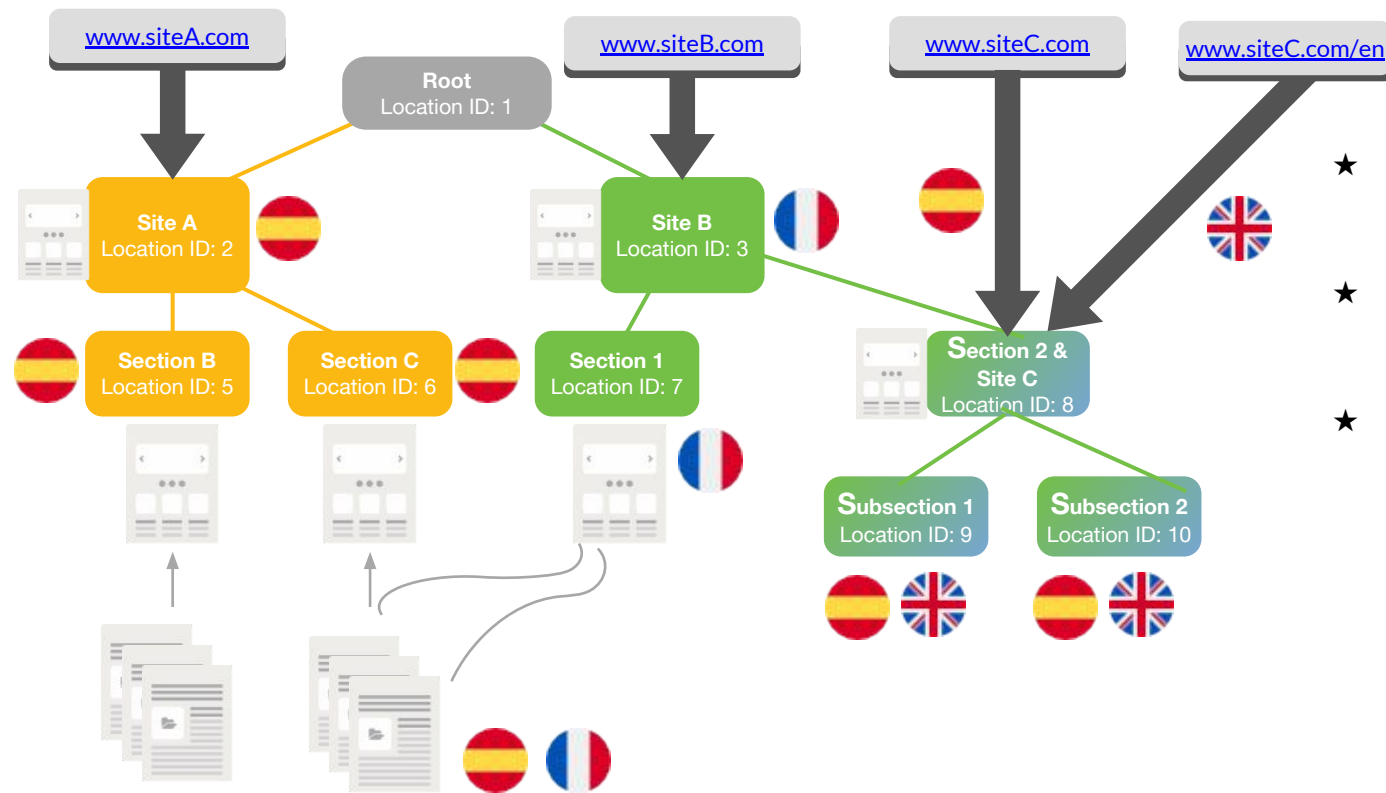
2 Varnish hace una subrequest que debe incluye estos headers

- `X-HTTP-Override: AUTHENTICATE`
- `Accept: application/vnd.ez.UserHash+text`
- Original cookie

4 Varnish añade el X-User-Hash a la request original

3 eZ devuelve una respuesta que incluye el X-User-Hash

Con eZ Platform es sencillo servir varios sites desde una instalación



- ★ Los sites pueden compartir contenidos y configuraciones entre ellos.
- ★ Además, esas configuraciones también pueden personalizarse a nivel de site.
- ★ Esta funcionalidad también posibilita versiones idiomáticas de cada site.

```
ezpublish:
  siteaccess:
    list: [es, eng, other, foo, bar]
    default_siteaccess: es
    match:
      URIElement: 1
```

Podemos hostear subdominios o dominios distintos

- Un siteaccess es un conjunto de configuraciones que se usan cuando se accede al sitio a través de una determinada dirección. .
- El siteaccess a usar se determina a través de uno o varios matchers.
- Hay varios matchers ya definidos.
 - URISearch
 - URIText
 - HostElement
 - HostText
 - Map\Host
 - Map\URI
 - Regex\URI
 - etc

```
ezpublish:
siteaccess:
match:
  Map\Host:
    www.foo.com: foo_front
    adm.foo.com: foo_admin
    www.bar-stuff.fr: bar_front
    adm.bar-stuff.fr: bar_admin
```

```
ezpublish:
siteaccess:
match:
  Regex\URI:
    regex: "^/foo(\\w+)bar"
    # Default is 1
    itemNumber: 1
```

```
ezpublish:
siteaccess:
match:
  Map\URI:
    es: site_espana
    fr: site_europa
    en: site_europa
    other: other_siteaccess
```

```
ezpublish:
siteaccess:
match:
  URIText:
    prefix: foo
    suffix: bar
```

Más en <https://doc.ezplatform.com/en/2.0/guide/siteaccess/>

así como personalizar diseños o decidir qué idiomas mostrar

- Los siteaccess pueden organizarse en grupos y compartir alguna o todas sus configuraciones.
- Por ejemplo, todos los sites pueden compartir base de datos o cada site puede usar la suya propia.
- De igual forma, pueden compartir todas las plantillas, algunas, o ninguna.
- Y también permiten el multidioma. Cada siteaccess puede configurar qué idiomas va a mostrar

Internamente, eZ Platform se encargará de presentar la versión idiomática o plantilla adecuada sin necesidad de cambiar el código desarrollado

ezdesign:

You declare all available designs under "design_list".

design_list:

my_design will be composed of "theme1" and "theme2"

"theme1" will be tried first. If a template cannot be found in "theme1", "theme2" will be tried out.

my_design: [theme1, theme2]

ezpublish:

...

system:

my_siteaccess:

my_siteaccess will use "my_design"

design: my_design

ezpublish:

system:

Configuration for the 'eng' SiteAccess

eng:

languages: [eng-GB]

esl:

languages: [esl-ES, eng-GB]



4

Algunos casos de uso



Añade ajo,
guisantes y
jamón



El mismo contenido es utilizado para ser servido en Web, apps móviles, generación de libros de recetas e incluso para mostrar recetas en el mismo robot de cocina. (en preparación)

Viking Line

4



- Desarrollado por la empresa [Ixonos](#)
- “Each ship of Viking Line fleet is unique which means that service contents must be easily customizable both by the crew and the land organization.”
- “eZ was implemented as the content management solution for the on-board digital platform. Content from eZ is displayed on-board Viking Line ships for mobile app, browser UI, in-cabin TV’s and digital signage screens.”

Mikko Sjöblom,
Business Director at Ixonos

<https://ez.no/Blog/Ixonos-creates-innovative-app-with-eZ-Publish-Platform-for-Viking-Line-passengers>

Borussia Dortmund



- Desarrollado por el propio equipo IT del club
- Cada departamento (marketing, venta de tickets, información sobre jugadores o partidos...) cuenta con accesos específicos y pueden trabajar por separado
- El club ganó el premio a la mejor web de los equipos de la Champions en el año 2014. En el año 2013 habían quedado en la posición 29 de 30 participantes.
<http://www.championsleaguewebsites.com/2014/>
- Desde un mismo repositorio se sirve contenido para web, apps en IOS o Android e incluso anuncios en los displays del Westfalenstadion, ahora llamado Signal Iduna Park


<https://ez.no/Blog/Case-Study-BVB-A-premier-football-club-with-clear-goals>

Movistar originales



- Desarrollado por [The Cocktail](#)
- Todos los sites están alojados en una única instalación y comparten base de datos. Comparten también infraestructura.
- Gracias al multisite se pueden aprovechar parte de las estructuras html además de los controladores Symfony para crear un nuevo site.
- Añadir un nuevo site es un proceso sencillo y que prácticamente no necesita trabajo a nivel de eZ Platform (sí a nivel de estilos)

<http://originales.movistarplus.es>

 Pulley y yo
os damos las
gracias.



Graphics by [Hakan Ertan](#). Thanks so much to [Bertrand Maugain](#), rest of the eZ Crew and the #ezcommunity for their invaluable help all these years.