CMS

Written by [Febogamedev]

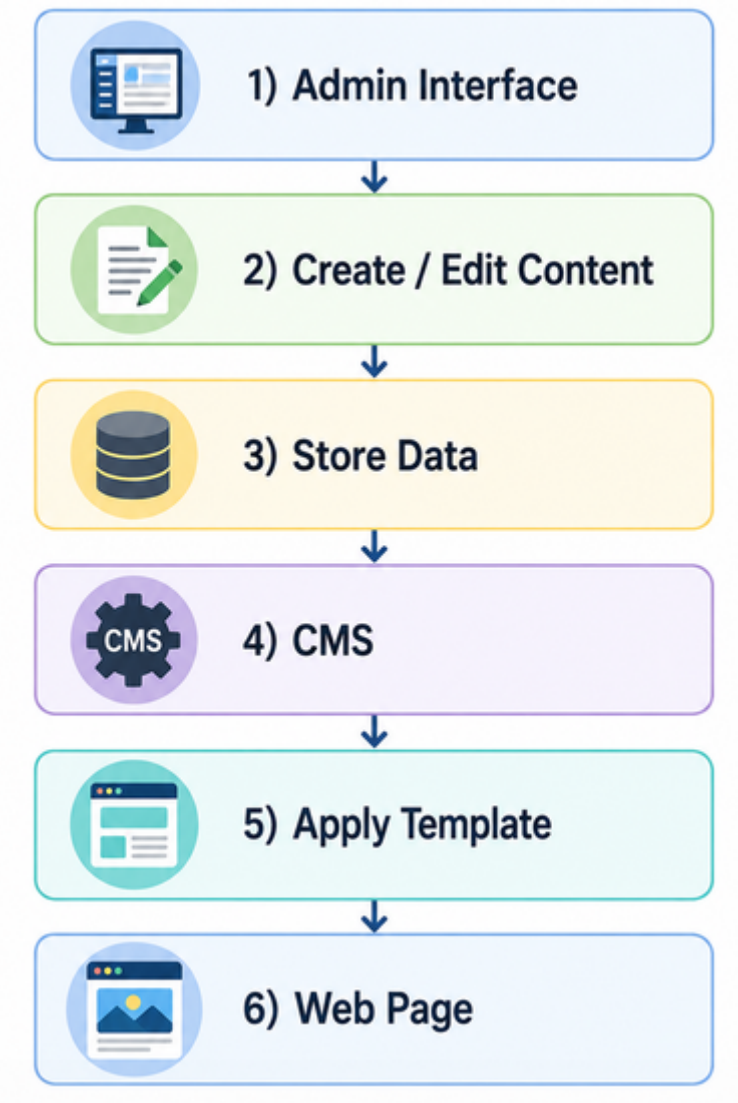
A Content Management System, abbreviated *CMS* and translated into Italian as sistema di gestione dei contenuti, is software that allows digital content to be created, organized, edited and published through a management interface (Barker, 2016).

On the Web, *CMS*s are mainly used to manage sites whose content needs frequent updates, avoiding the need to manually modify *HTML* code whenever a page is added or updated (Barker, 2016).

For example, to publish a new article, the author can fill in an interface containing fields such as:

- Title.
- Author.
- Text.
- Image.
- Category.
- Publication date.

The *CMS* stores this information and uses it to build the page that will be shown to users.



A CMS therefore allows content management to be separated, at least in part, from its visual presentation (Barker, 2016).

Most *CMS*s provide an administration interface, often called an **administration panel**, **dashboard**, **admin panel** or, in some contexts, **back-end** (Barker, 2016).

Through this interface, authorized users can manage the site directly from their browser.

For example, they can:

- Create and edit pages and articles.
- Upload images and other media.
- Organize categories and menus.
- Manage users.
- Change certain settings.
- Install extensions where applicable.
- Publish or schedule content.

Using a *CMS* therefore reduces the need to work directly on the code during ordinary editorial activities (Barker, 2016).

However, this does not mean that a *CMS* eliminates the need for technical skills. Advanced customization, developing new features, extensively changing the design, improving performance or managing security may require skills in *HTML*, *CSS*, JavaScript, server-side programming, databases and system administration (Barker, 2016).

A traditional web *CMS* is generally a server-side web application accessible through a browser (Barker, 2016).

It often uses several components:

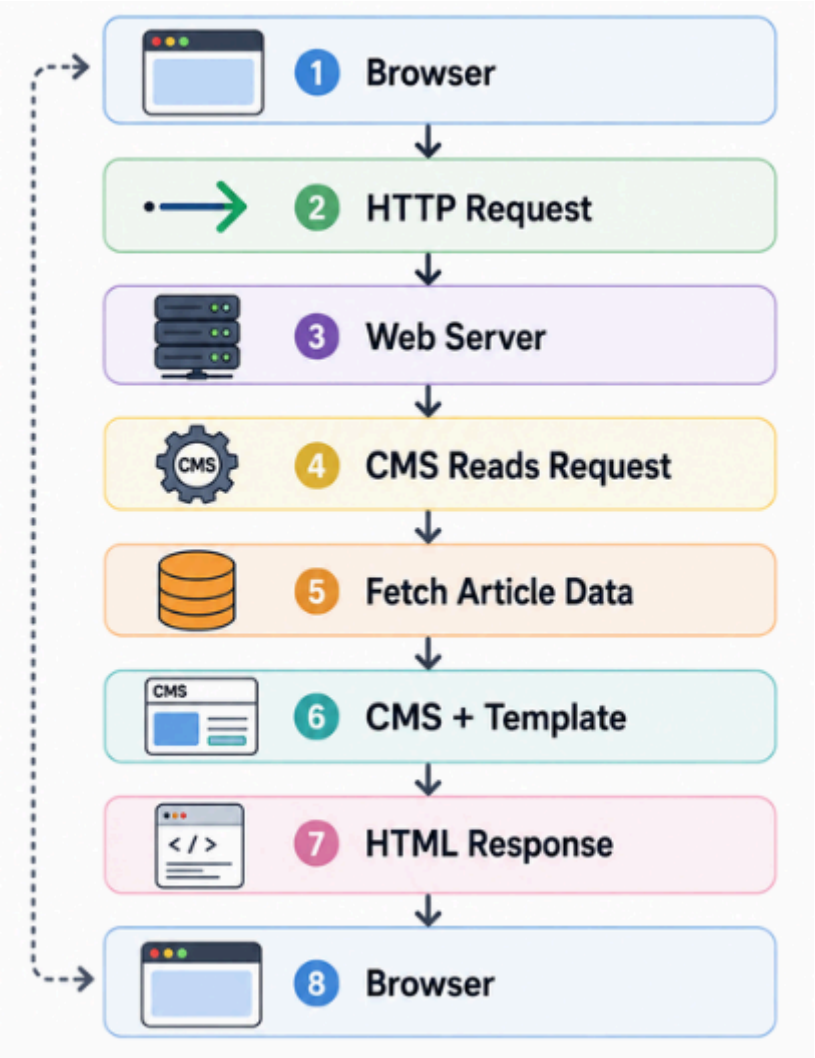
- Server-side applications.
- Databases.
- Media files.
- Templates or themes.
- Optional plugins or extensions.

The database can contain information such as:

- Articles.
- Pages.
- Users.
- Categories.
- Comments.
- Configurations.
- Metadata.

Images, videos and other files, on the other hand, can be kept in a separate storage system (Barker, 2016).

When a user requests a page, the *CMS* can retrieve the necessary data and dynamically generate the response (Barker, 2016).
For example, the user requests https://example.com/articoli/fotografia
the request is received and processed by the server/*CMS*, which retrieves the necessary information from the database or other storage systems and combines the content to generate a response.



A physical file called fotografia.html therefore does not necessarily have to exist on the server
The page can be dynamically generated by the *CMS* using the data stored in the database (Barker, 2016).

Not all *CMS*s are designed for the same purpose.
Some *CMS*s are more specialized in managing particular types of services or content, such as the following:

- Blogs.
- Wikis.
- Forums.
- E-commerce.
- Online newspapers.

There are also more general-purpose *CMS*s, designed to be adapted to different types of sites through settings, themes and extensions (Barker, 2016).
For example, a system originally created to publish articles can be extended to manage portfolios, business websites, e-commerce sites, online magazines and educational platforms.

However, the distinction is not always clear-cut: many *CMS*s can evolve and adapt to functions very different from those for which they were originally designed (Barker, 2016).

In traditional *CMS*s, the system handles both content management and the generation of the user-facing interface (Barker, 2016).

However, there is also an architecture called a **Headless *CMS*** (Contentful, 2024).
In a Headless *CMS*, the system primarily handles:

- Creating.
- Organizing.
- Storing.
- Managing.
- Delivering content.

But it does not necessarily determine how that content should be displayed (Contentful, 2024).
Content is generally made available through an *API* (a set of rules that allows different software systems to communicate with one another (Contentful, 2024; WordPress Developer Resources, 2024).
The same content can therefore be used by different interfaces at the same time (Contentful, 2024).
This model separates content management from its presentation even more clearly (Contentful, 2024).



The Features of a CMS

This chapter examines the most important features of a *CMS* (not headless):

- Content and presentation.
- Content management.
- Media management.
- User management.
- Plugins and extensions.
- Multilingual management.
- SEO.

Content and Presentation

One of the most important features of a *CMS* is the separation between: (Barker, 2016) content and presentation.

Content can consist of information such as a title, author, text, date, category and image.

Presentation, on the other hand, determines how this information should appear: title size, typeface, colors, spacing, number of columns, image placement and navigation structure.

The structure and arrangement of the elements are created through the **template**, a repeatable structure for different content (Barker, 2016).

By modifying the template, it is therefore possible to change the appearance of many pieces of content without editing each article individually.
Templates define the structure in which particular content is displayed (Barker, 2016).
An article template could include:

- Header.
- Navigation.
- Title.
- Author and date.
- Image.
- Content.
- Related articles.
- Footer.

The CMS automatically inserts the data for the requested article in the appropriate places.

In many *CMS*s, different templates, style sheets, images and other resources are grouped together within a **theme** (Barker, 2016).

A theme can therefore define much of the site's visual identity and overall structure.
Templates and themes allow ready-made structures to be used and many aspects of presentation to be changed without working directly on the code.
More advanced customization may still require web design and development skills.

Content Management
One of the main functions of a *CMS* is to enable management of the content lifecycle (Barker, 2016).

A piece of content can pass through several states:

```
Creation
   ↓
Draft
   ↓
Review
   ↓
Publication
   ↓
Update
   ↓
Archiving or deletion
```

Many CMSs also allow users to:

- Schedule future publication.
- Save previous revisions.

- Organize content through categories or tags.
- Link pieces of content to one another.
- Define readable URLs.
- Assign images and metadata.

These features become particularly important when the site contains hundreds or thousands of pages (Barker, 2016).

Media Management
CMSs often provide tools for uploading and organizing files such as images, audio, video, PDFs and documents (Barker, 2016)

In addition to the file itself, associated information, called metadata, can be stored, such as:
- Title.
- Alternative title.
- Alternative text.
- Caption.
- Author.

Separating the file from its metadata makes it easier to manage and reuse within the site (Barker, 2016).

User Management
A *CMS* can be used by several people at the same time (Barker, 2016).

For this reason, it generally provides a system of users, roles and permissions (Barker, 2016).

For example:
- Administrator: can manage the entire system.
- Editor: can review and publish content.
- Author: can create and edit their own content.
- Contributor: can prepare content without necessarily being able to publish it.

More complex systems may include fully developed editorial workflows (Barker, 2016).
In this way, the *CMS* manages not only the content but also the process through which it is produced and published (Barker, 2016).

Plugins and Extensions
Many *CMS*s can be expanded through additional components called, depending on the system (Barker, 2016):
- Plugins.
- Modules.
- Extensions.

These components can add new features (Barker, 2016).

For example:

- E-commerce.
- Contact forms.
- Newsletters.
- Bookings.
- Photo galleries.
- Multilingual support.
- SEO.
- Caching
- Payment systems.

Extensions make the *CMS* very flexible, but also increase the system's complexity. Each additional component can affect: (Barker, 2016):

- Security.
- Performance.
- Compatibility.
- Maintenance.

For this reason, it is important to use only the extensions that are actually needed and keep them up to date (Barker, 2016).

Multilingual Management

Some *CMS*s allow the same content to be managed in multiple languages (Barker, 2016). Multilingual management does not necessarily concern only the text of articles.
The following may also need to be managed separately:

- URLs.
- Menus.
- Categories.
- Images.
- Metadata.
- Interface text.
- *SEO*.

Multilingual support may be integrated directly into the *CMS* or provided through extensions (Barker, 2016).

SEO

Many CMSs provide tools for managing certain elements that are important to search engines (Google, n.d.).
For example:

- Page titles.
- URLs.
- Metadata.
- Sitemaps.
- Redirects.
- Structured data.

These tools can facilitate a site's technical optimization, but using a *CMS* does not automatically guarantee good rankings in search results (Google, n.d.).

A site's visibility also depends on many other factors, such as:

- Content quality.
- Information structure.
- Performance.
- Accessibility.
- Usability.
- Authority.
- Technical correctness.

The *CMS* is therefore a tool that can facilitate *SEO* management, not a system that automatically guarantees good rankings (Google, n.d.).

WordPress

WordPress is an open source *CMS* used to create and manage websites (WordPress.org, n.d.-a).

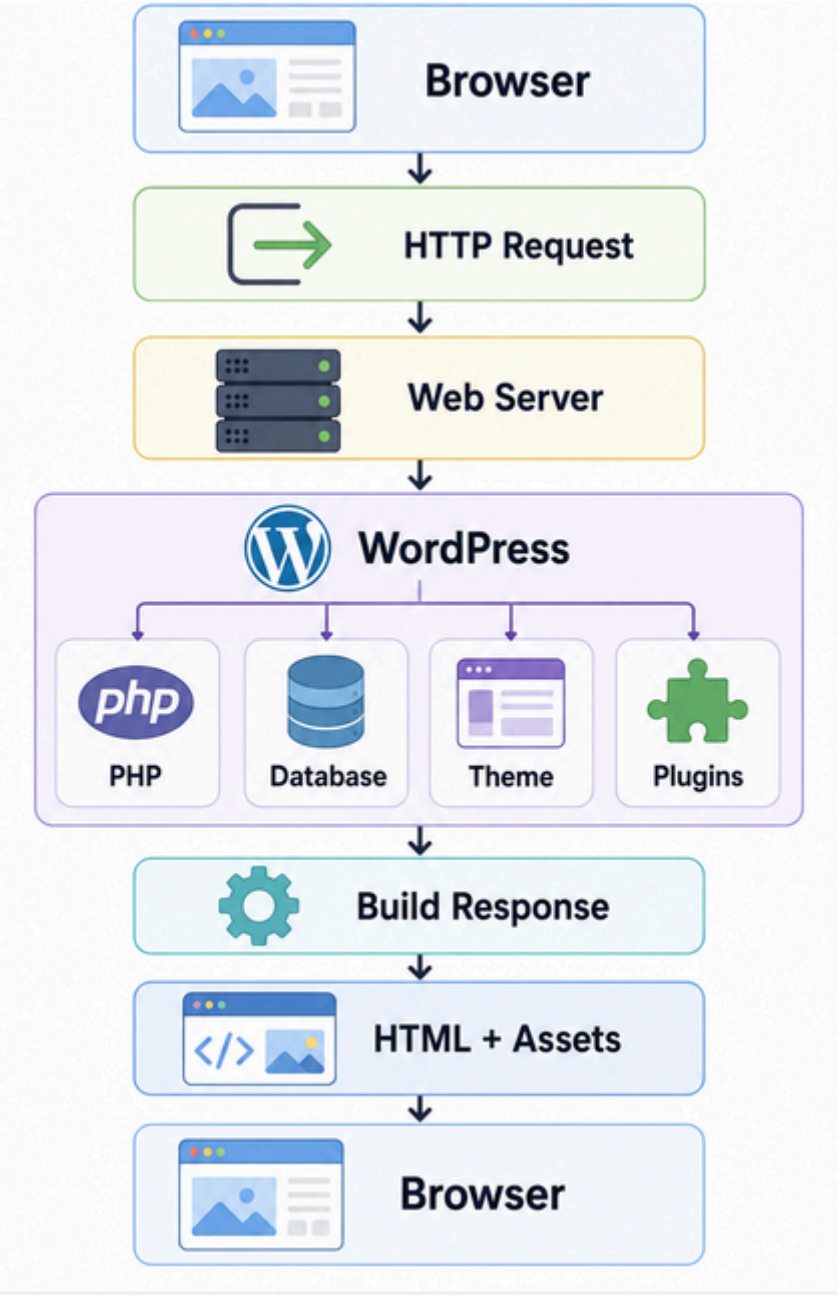
It began as a platform focused on publishing blogs, but over time evolved into a general-purpose *CMS* that can be used to create different types of sites, such as blogs and online magazines, portfolios, business websites, institutional websites, e-commerce sites, educational platforms, communities and web applications based on content management (WordPress.org, n.d.-a).

WordPress is **open source** software: its source code can be studied, modified and distributed under the terms of its license. The project is developed by an international community of developers and contributors (WordPress.org, n.d.-b).

It is important to distinguish WordPress.org from WordPress.com, because they do not refer to exactly the same thing (WordPress.com, 2026).

- **WordPress.org** is the website of the WordPress open source project. From here, you can obtain the WordPress software and install it on a hosting service. In this case, the owner manages, directly or through their provider, aspects such as the domain, hosting, WordPress installation, database, themes, plugins, updates, backups and settings. This is commonly called **self-hosted WordPress**, meaning that it is hosted on infrastructure chosen by the website owner.
- **WordPress.com**, on the other hand, is a commercial hosting service based on WordPress that provides the infrastructure needed to create and publish a site. In this case, part of the technical management is provided directly by the service.

WordPress is a server-side web application.
In its traditional use, we can represent its architecture as that of an ordinary *CMS*: it receives a request and combines its content to generate the page.



WordPress uses *PHP* to execute much of its server-side logic and normally uses *MySQL* or *MariaDB* as its database management system. (WordPress.org, n.d.-d)
The server environment must also provide the services needed for the application to work and, on modern sites, support *HTTPS* to allow secure communication (WordPress.org, n.d.-d).

The database can store information such as articles, pages, users, comments, categories, tags, settings and metadata

Media files, such as images and documents, are normally also stored as files in the site's storage space, while WordPress keeps the information needed to manage them in the database (WordPress.org, n.d.-c).

Example
Imagine that the user opens “https://example.com/articoli/come-funziona-http/”.
A physical file called come-funziona-http.html does not necessarily have to exist
In simplified terms, the process may be:

```
BROWSER
GET /articoli/come-funziona-http/
        ↓
WEB SERVER
        ↓
WORDPRESS
interprets the requested URL
        ↓
DATABASE
retrieves: title, text, author, date, images, categories, etc.
        ↓
WORDPRESS
combines the data with the appropriate template
        ↓
generates the HTML response
        ↓
BROWSER
```

WordPress is therefore a concrete example of what we saw when discussing request routing in web applications and dynamic page generation.

The public U*RL* represents a logical resource on the site and does not necessarily have to correspond to the location of a physical file on the server.

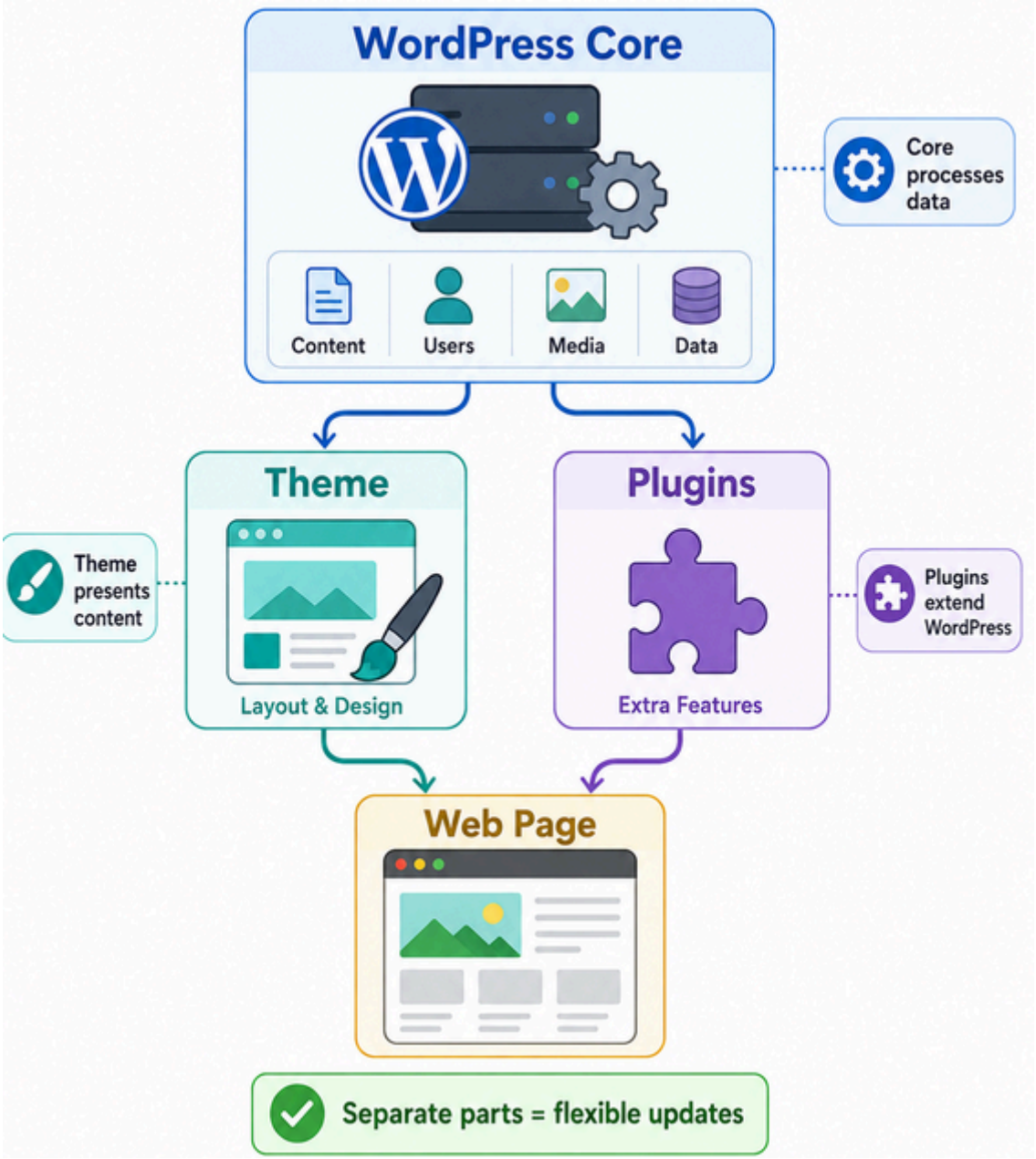
One of the most important features of WordPress is its extensibility (WordPress.org, n.d.-a). We can start from the same basic WordPress installation and build very different systems through themes and plugins (WordPress Developer Resources, 2023b, n.d.-b).
When you install WordPress, you first install the **Core**, which is the *CMS*’s basic system (WordPress.org, n.d.-a).

- The Core provides fundamental features, such as managing content, users, media, the database, themes and plugins.
- The theme primarily determines how content is structured and presented in the site’s public interface.
- Plugins, on the other hand, are optional components that add or modify features.

During the normal operation of a WordPress site, the Core retrieves and processes the requested data, while the theme determines how it should be presented to generate the page intended for the browser (WordPress Developer Resources, n.d.-b).
Plugins are not required for WordPress’s basic operation, but can extend or modify its features and also interact with the site’s presentation (WordPress Developer Resources, 2023b).

The separation between Core, themes and plugins is important because it allows part of the system to be updated, replaced or modified without necessarily rewriting the entire site. In practice, however, the boundaries are not always perfectly clear: some themes include advanced features, and some plugins can also profoundly change the interface's appearance and behavior (WordPress Developer Resources, 2023b, n.d.-b).

This allows a content management platform to be designed that can be configured, designed and extended in different ways.



WordPress Features

The most important features of WordPress are listed below:

- The administration panel.

- Content.
- Content taxonomy.
- Block editor.
- Patterns.
- Templates.
- Themes.
- Plugins.
- Media management.
- URL management.
- Updates and maintenance.
- Security.

The Administration Panel
WordPress provides a restricted area through which authorized users can administer the site (WordPress.org, n.d.-a).

This interface is commonly called the **Dashboard** or administration panel.
From here, depending on the user's permissions, it is possible to manage:
- Posts.
- Pages.
- Media.
- Comments.
- Appearance.
- Plugins.
- Users.
- Tools.
- Settings.

The panel therefore allows much of the site to be modified through a graphical interface, without working directly on files and the database (WordPress.org, n.d.-a).

However, this does not mean that WordPress eliminates the need for technical skills: advanced customization, theme and plugin development, security, performance and infrastructure configuration may require specialist knowledge.

The official documentation describes the Dashboard as the central place for managing publishing, media, settings and other administrative functions (WordPress.org, n.d.-a).

Content
WordPress allows different types of content to be managed (WordPress Developer Resources, 2023a).
Two of the most important are:
- Posts.
- Pages.

<u>Posts</u>
Articles, or posts, are content normally intended for publication organized over time.
For example, news, blog articles, updates, reviews and announcements.

They can be associated with organizational elements such as date, author, categories and tags.

Pages

Pages are generally used for relatively stable content related to the site's structure.
For example, "Home", "About Us", "Contact", "Services" and "Privacy"
The distinction does not concern the final *HTML* format: both are turned into web pages.
It mainly concerns the way WordPress organizes content semantically (WordPress Developer Resources, 2023a).

Example
Imagine this website: fotografiaexample.com

In the main menu, you find:
- Home.
- About Us.
- Services.
- Blog.
- Contact.

The first pages are relatively stable content:

For example, the "Services" page could have this address: fotografiaexample.com/servizi
Its content rarely changes and mainly serves to define the site's structure.

In the Blog, on the other hand, new content is published periodically:
- September 10 → How to Photograph at Night.
- September 18 → The Best Lenses for Portraits.
- September 25 → A Guide to Urban Photography.

An article could therefore be: fotografiaexample.com/blog/fotografia-notturna

and WordPress can associate it with:
- A date.
- An author.
- A category, for example Tutorial.
- Tags, for example night, photography and tripod.

The important point is that both "Services" and "How to Photograph at Night" ultimately become ordinary HTML web pages displayed by the browser.

The difference lies in how WordPress treats them:
- Page → structural and stable content.
- Post → editorial content organized over time.

If pages and posts are not enough, WordPress uses the concept of a **Post Type**, meaning a type of content (WordPress Developer Resources, 2023a).
Posts and pages are two examples.

The system also includes other types and can be extended with new Custom Post Types (WordPress Developer Resources, 2023a).

Imagine a website devoted to cinema.
In addition to posts and pages, we could create films, directors and festivals.
For example, a film-type content item could have specific information such as title, director, year, running time, poster, genre and synopsis.

In this way, WordPress does not have to be used solely as a system for writing pages, but can become a tool for modeling and organizing structured content (WordPress Developer Resources, 2023a).

WordPress distinguishes between different post types and allows developers to register custom ones (WordPress Developer Resources, 2023a).

Content Taxonomy
Content can be organized through classification systems called taxonomies (WordPress Developer Resources, 2023c).

Two common examples in WordPress are:
- Categories.
- Tags.

Categories allow content to be organized through a classification that can be hierarchical. (WordPress Developer Resources, 2023c)
A category can therefore contain subcategories.
For example, the “Technology” category can have “Internet”, “Hardware” and “Artificial Intelligence” as subcategories.
A content item can be associated with one or more categories.

Tags, on the other hand, allow characteristics or topics shared across content to be described and, unlike categories, are normally not organized hierarchically (WordPress Developer Resources, 2023c).
A content item can be associated with multiple tags.
For example, a website can have tags such as “*HTTP*”, “*CSS*”, “WordPress” and “accessibility”.

Categories and tags therefore allow large amounts of content to be classified and related to one another, facilitating navigation and the construction of the site’s information architecture (WordPress Developer Resources, 2023c).

Therefore:
- A Custom Post Type defines a new type of content.
- A taxonomy allows content to be classified and related to one another.

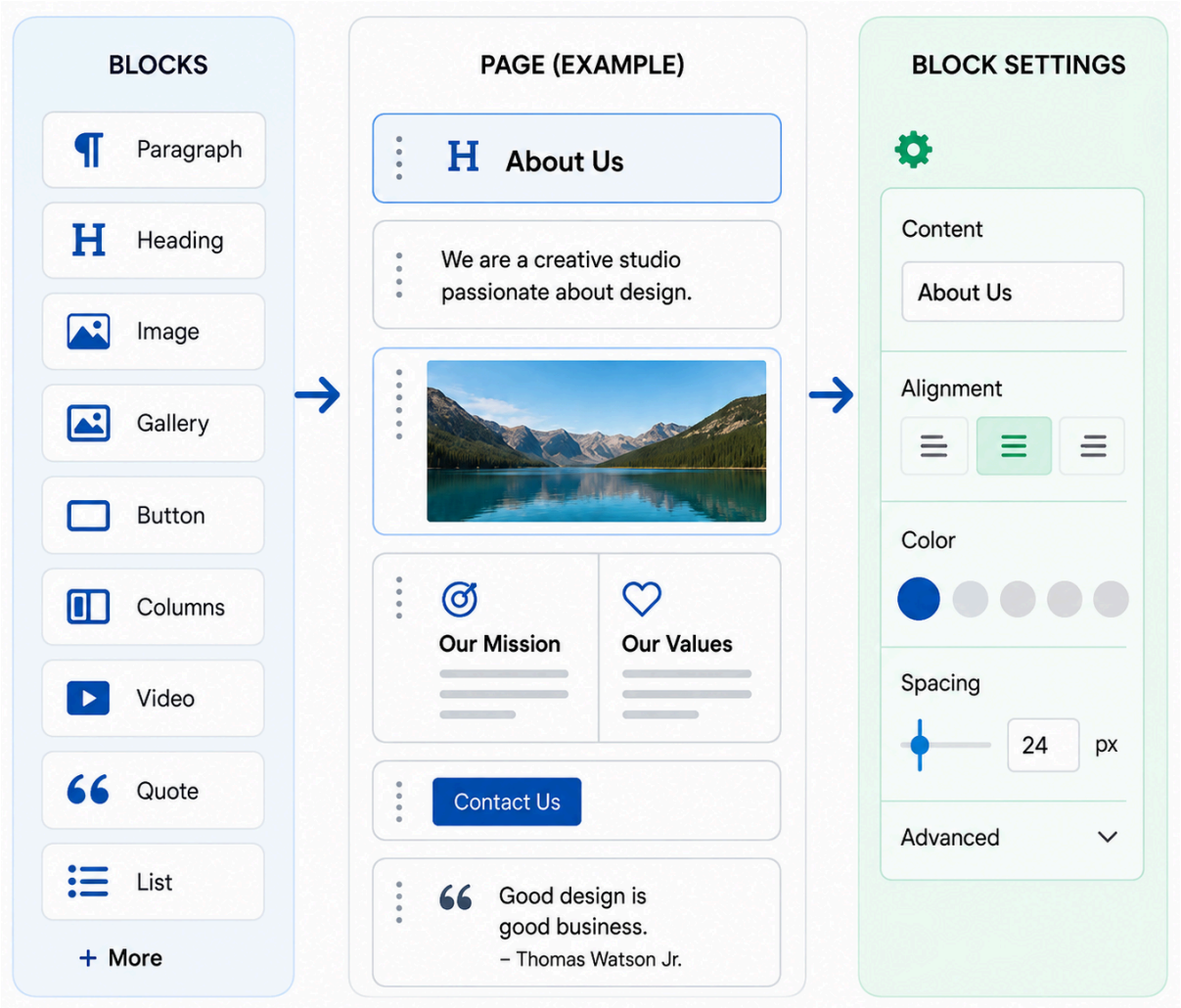
Block Editor
WordPress uses a block-based system as its main editor, commonly called the **Block Editor** (WordPress.org, 2026b).

A page is not simply seen as a single text field, but as a composition of different elements. For example, a page can consist of the following blocks: paragraph, heading, image, gallery, button, column, video, quote, list, etc.

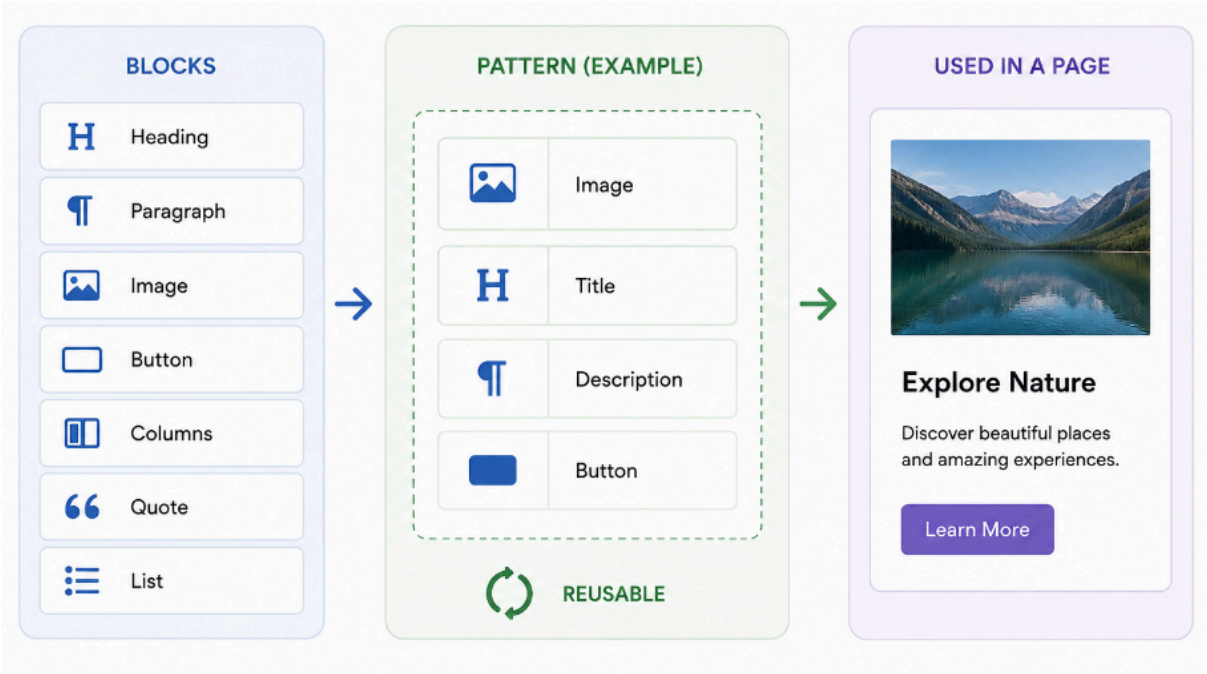
Each block has its own settings relating to content and, to varying degrees, presentation (WordPress.org, 2026b).

The Block Editor is the default WordPress editor and allows multimedia content to be built by combining different types of blocks (WordPress.org, 2026b).

A page can therefore be designed as a composition of modular components, which can be combined and, in some cases, reused (WordPress.org, 2026b).



Patterns
Multiple blocks can be organized into reusable structures called patterns (WordPress Developer Resources, 2026b).
For example, a pattern consists of an image, title, description and button to represent a card.



A pattern can serve as a starting structure to be inserted and edited freely or, depending on its configuration, it can be synchronized so that changes are reflected across its different instances (WordPress Developer Resources, 2026b).

Patterns allow coherent, repeatable visual systems to be created, a principle particularly important in web design and design systems.

Templates
Templates define the structure used to display particular categories of content (WordPress Developer Resources, n.d.-a).

For example, a post template can include:

- Post title.
- Author and date.
- Image.
- Content.
- Related articles.
- Footer.



All posts can use the same structure while the data changes.
The template therefore determines how a type of content is presented, while the database contains the specific data (WordPress Developer Resources, n.d.-a).
WordPress can also use different templates for posts, pages, archives, search results, 404 pages and other situations (WordPress Developer Resources, n.d.-a).

Themes
A WordPress theme can define visual aspects such as typography, colors, layout, templates, styles, the header, the footer, post presentation, archives and error pages (WordPress Developer Resources, n.d.-b).
Changing the theme can therefore profoundly alter the site's appearance without necessarily changing the stored content (WordPress Developer Resources, n.d.-b).
The web page is, in fact, a combination of content (posts, pages, images, etc.) and the theme.



The WordPress documentation describes the theme as the system that controls the site's design and presentation (WordPress Developer Resources, n.d.-b).

WordPress supports different theme models:

- In classic themes, many aspects of the structure are defined through template files written mainly in *PHP*, *HTML* and *CSS*.
- In modern Block Themes, many global parts of the site are also built using blocks.

When a block theme is used, WordPress provides the **Site Editor** (WordPress.org, 2026a). The Site Editor allows changes not only to the content of an individual page but also to the site's overall structure (WordPress.org, 2026a).
For example, Templates, Patterns, Global Styles, etc.

We can therefore distinguish:

- Block editor: mainly edits content.
- Site editor: edits the site's global structure and design.

Plugins
Plugins are software components that allow WordPress features to be added or modified (WordPress Developer Resources, 2023b).
Plugins do not normally modify the Core directly (WordPress Developer Resources, 2023b).

This is because WordPress provides specific mechanisms, called **hooks**, through which plugins can connect to the system's operation and modify or extend its behavior (WordPress Developer Resources, 2023b).

Plugins can add features such as e-commerce, contact forms, SEO, multilingual support, galleries, bookings, security, caching, newsletters, social networks, ..

For example, a WordPress site could be turned into an e-commerce site through a plugin that adds:

- Products.
- Shopping cart.
- Orders.
- Payments.
- Customer management.

The official WordPress directory provides a vast ecosystem of open source plugins that extend the features of the basic software (WordPress.org, n.d.-a).

Media Management

WordPress has a Media Library for managing (viewing, searching, editing and deleting) images, videos, audio and documents uploaded to the site (WordPress.org, n.d.-c).

Each resource can be associated with different information called metadata, including a title, caption, description and alternative text (WordPress.org, n.d.-c).

Alternative text plays a fundamental role in accessibility because it provides a textual alternative to an image's content or function, which can be used, for example, by assistive technologies such as screen readers. (W3C Web Accessibility Initiative, n.d.)
It can also be used in certain situations where the image is not displayed (W3C Web Accessibility Initiative, n.d.).

Users, Roles and Permissions

WordPress allows multiple users to collaborate on managing the same site (WordPress.org, n.d.-e).
Each user can have a particular role, which corresponds to specific capabilities, meaning permissions (WordPress.org, n.d.-e).
WordPress associates roles with sets of specific capabilities, such as publishing posts, editing content, moderating comments or managing users (WordPress.org, n.d.-e).

The default roles include:

- Administrator: manages settings, users, plugins and much of the system.
- Editor: manages and publishes content.
- Author: mainly publishes their own content.
- Contributor: can prepare content with more limited permissions.
- Subscriber: has very limited permissions and can mainly manage their own user profile.

URL Management

WordPress allows readable *URL*s, commonly called permalinks (WordPress.org, 2018).
For example: https://example.com/?p=125
can be presented publicly as: https://example.com/articoli/come-funziona-http/.

The second *URL* does not imply the existence of a physical folder “/articoli/” or a file called “come-funziona-http” (WordPress.org, 2018).

WordPress interprets the *URL* and identifies the corresponding content, then generates the page through the theme (WordPress.org, 2018).

The fundamental concept is that a WordPress page should not be thought of as a single, ready-made HTML file.
A displayed page may be the result of combining numerous elements.
The page the user sees is therefore the result of processing different data, templates, code and settings.

Updates and Maintenance
A WordPress site is a software system and must be maintained over time (WordPress Developer Resources, 2026a, 2026c).

At least WordPress Core, Themes, Plugins, PHP, the Database, Backups and Security must be considered**.**

WordPress allows the Core to be updated and also supports update systems for plugins and themes. (WordPress Developer Resources, 2026c)
The official documentation recommends keeping these components up to date and making backups before important operations (WordPress Developer Resources, 2026c).

Security
The security of a WordPress site does not depend solely on the CMS (WordPress Developer Resources, 2026a).

It depends on the entire system:
- Server.
- WordPress core.
- Plugins.
- Themes.
- Passwords.
- Permissions.
- HTTPS.
- Backups.
- Configuration.

A vulnerable plugin, a weak password or outdated software can compromise the site’s security even when WordPress Core is up to date (WordPress Developer Resources, 2026a).

WordPress has a team dedicated to Core security, but the official documentation emphasizes that an installation's security also requires proper configuration and maintenance practices (WordPress Developer Resources, 2026a).

For this reason, it is important to:

- Keep software up to date.
- Use only reliable and necessary plugins and themes.
- Make backups.
- Use *HTTPS*.
- Manage users and permissions properly.
- Use secure credentials.

Supplementary material

Exercises

Designing the CMS for an Online Magazine
A school wants to create an online magazine in which several teachers and students can publish articles.
Each article must contain:

- Title.
- Author.
- Date.
- Text.
- Main image.
- Category.
- Tags.

The site must also allow users to:

- Prepare articles without publishing them immediately.
- Have content reviewed by a teacher.
- Schedule future publication.
- Use the same visual structure for all articles.
- Change colors and typography in the future without manually editing each article.
- Manage hundreds of images.
- Have an Italian and an English version.
- Add a newsletter later.

Conceptually design the system.

- Explain why a CMS is more appropriate than manually editing individual HTML files.
- State which information you would store as content and which instead belongs to presentation.
- Design a possible article template, stating which elements it should contain.
- Explain why modifying the template makes it possible to change the structure of many articles without changing the data of each content item.
- Describe a possible editorial lifecycle for an article from creation to publication, including at least a draft and a review stage.
- Assign appropriate roles to the student, the teacher in charge and the site administrator, indicating which operations each should be able to perform.
- Explain how you would organize articles using categories and tags.
- State which metadata you would store for images in the media library.
- Explain which elements must be considered in multilingual management beyond simply translating the text.
- The school wants to add a newsletter. Explain why this feature could be added through an extension rather than by modifying the basic system directly.
- Explain which risks may increase when a large number of plugins or extensions are installed.

- Finally, reconstruct the possible process followed when a visitor opens an article: Browser → request → CMS → database → template → HTML generation → response → browser.

Building a WordPress Site Without Confusing Content and Design

You need to create a website about cinema using WordPress.
The site must contain:

- Home.
- About Us.
- Contact.
- A section of in-depth articles.
- Profiles dedicated to films.
- Profiles dedicated to directors.
- Categories such as “Italian Cinema”, “Animation” and “Documentary”.
- Tags such as “Kubrick”, “Science Fiction”, “Cinematography” and “Editing”.
- A common layout for all film profiles.
- A contact form.
- An Italian and an English version.

Analyze how you would organize the project using WordPress.

- Decide which content you would create as pages and which as posts, explaining your choice.
- Assess whether “Films” and “Directors” should be ordinary pages or Custom Post Types and justify your choice.
- Explain the difference between a Post Type and a taxonomy using the proposed case.
- Design a possible template for a Film profile including title, director, year, running time, poster, genre and synopsis.
- Explain why the data for two different films can use the same template without being stored directly within it.
- Imagine building a card consisting of an image, title, short description and button. Explain why it could be useful to turn it into a pattern.
- Distinguish the roles of the Block Editor, templates, patterns and themes.
- The client decides to completely change the typography, colors, header and footer. State which component should mainly be modified and explain why the content can remain unchanged.
- The site needs a contact form and multilingual support. State what role plugins could play.
- Explain the difference between WordPress Core, a theme and a plugin.
- A user visits ”https://example.com/film/2001-odissea-nello-spazio/”. Explain why a physical file called 2001-odissea-nello-spazio.html does not necessarily have to exist on the server.
- 12. Reconstruct the process: Browser → Web server → WordPress → database → content retrieval → template/theme → HTML → browser.
- A plugin used by the site has not been updated for two years and has a vulnerability. Explain why keeping only WordPress Core up to date is not enough to consider the site secure.

- State at least four elements of the entire system that must be considered in the maintenance and security of a WordPress site.

Bibliography

Barker, D. (2016). *Web content management: Systems, features, and best practices*. O'Reilly Media. https://books.google.com/books/about/Web_Content_Management.html?id=x6_NCwAAQBAJ

Contentful. (2024, May 2). *Headless CMS explained in one minute*. https://www.contentful.com/headless-cms/

Google. (n.d.). *SEO starter guide: The basics*. Retrieved September 2, 2026, from https://developers.google.com/search/docs/fundamentals/seo-starter-guide

WordPress.com. (2026, August 11). *WordPress.com vs. WordPress.org: What's the difference?* https://wordpress.com/support/com-vs-org/

WordPress.org. (2018, November 16). *Customize permalinks*. https://wordpress.org/documentation/article/customize-permalinks/

WordPress.org. (2026a). *Site Editor*. https://wordpress.org/documentation/article/site-editor/

WordPress.org. (2026b, July 31). *WordPress Block Editor*. https://wordpress.org/documentation/article/wordpress-block-editor/

WordPress.org. (n.d.-a). *Features*. Retrieved September 2, 2026, from https://wordpress.org/about/features/

WordPress.org. (n.d.-b). *License*. Retrieved September 2, 2026, from https://wordpress.org/about/license/

WordPress.org. (n.d.-c). *Media Library screen*. Retrieved September 2, 2026, from https://wordpress.org/documentation/article/media-library-screen/

WordPress.org. (n.d.-d). *Requirements*. Retrieved September 2, 2026, from https://wordpress.org/about/requirements/

WordPress.org. (n.d.-e). *Roles and capabilities*. Retrieved September 2, 2026, from https://wordpress.org/documentation/article/roles-and-capabilities/

WordPress Developer Resources. (2023a, December 14). *Custom Post Types*. https://developer.wordpress.org/plugins/post-types/

WordPress Developer Resources. (2023b, December 14). *Introduction to Plugin Development*. https://developer.wordpress.org/plugins/intro/

WordPress Developer Resources. (2023c, December 14). *Taxonomies*. https://developer.wordpress.org/plugins/taxonomies/

WordPress Developer Resources. (2024, January 16). *REST API Handbook*. https://developer.wordpress.org/rest-api/

WordPress Developer Resources. (2026a). *Hardening WordPress*. https://developer.wordpress.org/advanced-administration/security/hardening/

WordPress Developer Resources. (2026b, July 17). *Patterns*. https://developer.wordpress.org/block-editor/reference-guides/block-api/block-patterns/

WordPress Developer Resources. (2026c). *Upgrading WordPress*. https://developer.wordpress.org/advanced-administration/upgrade/upgrading/

WordPress Developer Resources. (n.d.-a). *Templates*. Retrieved September 2, 2026, from https://developer.wordpress.org/themes/templates/templates/

WordPress Developer Resources. (n.d.-b). *Theme Handbook*. Retrieved September 2, 2026, from https://developer.wordpress.org/themes/

W3C Web Accessibility Initiative. (n.d.). *Images tutorial*. Retrieved September 2, 2026, from https://www.w3.org/WAI/tutorials/images/

Further reading

- Deane Barker, Web Content Management: Systems, Features, and Best Practices, O'Reilly Media, 2016 — Probably the most comprehensive theoretical reference for this handout. It examines CMS architecture, content modeling, editorial workflows, publishing, asset management, content reuse, and system implementation. Its platform-independent approach makes it particularly useful for understanding what distinguishes a CMS from an ordinary website and why managing content requires more than simply providing a graphical editor.
- Matthew MacDonald, WordPress: The Missing Manual, 3rd ed., O'Reilly Media, 2020 — One of the most accessible practical introductions to WordPress installation, administration, posts, pages, categories, themes, plugins, media management, and website maintenance. It explains how the different components work together to create and manage a complete website. It is particularly suitable for students learning WordPress for the first time, although some interface details predate the latest developments in block themes and site editing.
- Lisa Sabin-Wilson, WordPress All-in-One For Dummies, 5th ed., Wiley, 2024 — A comprehensive practical guide to building, configuring, administering, and maintaining WordPress websites. It examines themes, plugins, content management, customization, website security, and optimization. Its progressive approach makes it particularly useful for moving from basic publishing activities to the broader responsibilities involved in managing a WordPress installation.
- Tammie Lister, WordPress Styling with Blocks, Patterns, Templates, and Themes, Packt Publishing, 2024 — Probably the most directly relevant book for the chapter's treatment of modern WordPress design tools. It focuses on the Block Editor, reusable patterns, templates, template parts, block themes, global styles, and the Site Editor. It explains how WordPress enables designers to create consistent visual systems through reusable components without necessarily modifying PHP or CSS directly. Particularly valuable for understanding the distinction between editing individual content and controlling the overall design of a website.
- Carrie Hane and Mike Atherton, Designing Connected Content: Plan and Model Digital Products for Today and Tomorrow, New Riders, 2018 — An excellent complement for structured content, content modeling, information architecture, content reuse, and CMS design. The authors explain how digital content can be modeled independently of its final presentation, allowing the same information to support different interfaces and communication channels. It is particularly relevant to the handout's explanation of Headless CMS architecture, in which content management and presentation are separated through APIs.

Editorial notes

In producing these materials, I have used generative artificial intelligence tools to support the writing process, particularly to improve the form of the text, reorganize content, refine wording, and speed up certain editorial tasks.

As I work independently on the production of these materials, I try to automate any activities that can reasonably be automated, so that I can devote more time to research, design, and content development.

However, artificial intelligence does not determine the content of this work: the choice of topics, structure, ideas, interpretations, examples, and teaching approach are developed by me. AI is therefore used primarily as a tool to support production and formal revision, while authorship and responsibility for the design and content remain mine.

Unless otherwise indicated, this material is distributed under the Creative Commons Attribution–NonCommercial–ShareAlike 4.0 International license (CC BY-NC-SA 4.0).

You are therefore free to share, redistribute, adapt, and create derivative works based on this material, provided that appropriate credit is given, the material is not used for commercial purposes, and any modified or derivative versions are distributed under the same license.

