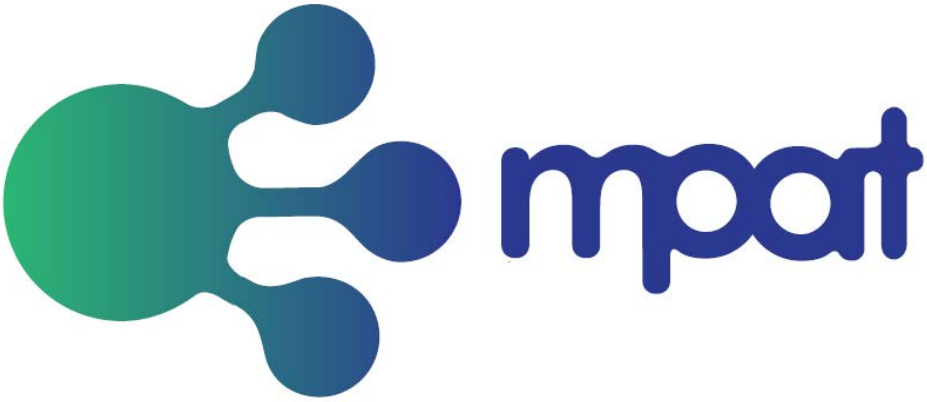


State-of-the-art and Requirements

Deliverable D4.1

MPAT File ID:	State-of-the-art and requirements.docx
Version:	1.0
Deliverable number:	D4.1
Authors:	FOKUS, IRT, ULANC, FINCONS, IMT
Contributors:	Miggi Zwicklbauer (FOKUS), Jean-Claude Dufourd (IMT), Matthew Broadband (LANC), Jamie Jellicoe (ULANC), Stefano Miccoli (FINCONS), Fabian Schiller (IRT)
Internal reviewers:	Benedikt Vogel (IRT)

Work Package:	WP4
Task:	T4.1
Nature:	R
Dissemination:	PU
Status:	Final
Delivery date:	07.06.2016

Version and controls:

Version	Date	Reason for change	Editor
0.1	11.04.2016	Set-up document and structure and added chapter “Introduction”	Fabian Schiller
0.2	02.05.2016	Added chapter “Joomla”	Fabian Schiller
0.3	12.05.2016	Added chapter “System built from scratch”	Matthew Broadbent, Jamie Jellicoe
0.4	02.06.2016	Added chapter “WordPress”	Miggi Zwicklbauer
0.5	30.05.2016	Added chapter “Drupal”	Stefano Miccoli
0.6	31.05.2016	Added chapter “Requirements from scenarios”	Jean-Claude Dufourd
0.7	03.06.2016	Added „Overall conclusion“	Fabian Schiller
1.0	07.06.2016	Internal Review and finalized document	Benedikt Vogel, Fabian Schiller

Acknowledgement: The research leading to these results has received funding from the European Union's Horizon 2020 Programme (H2020-ICT-2015, call ICT-19-2015) under grant agreement n° 687921.

Table of Contents

1 Introduction

This deliverable should give an overview of the initial set of requirements derived from the scenarios in WP2. Furthermore a detailed state-of-the-art review in respect to already available content-management-systems (CMS) and their usage for the designated MPAT system is conducted.

The deliverable starts off with describing the process of how the consortium analysed the scenarios and derived use cases from which the basic technical functionalities were deducted. As a second step the required MPAT modules are specified, which sets the overall concept and which will be used as a starting point for the initial development of the core system. Followed by a CMS analysis to find out which software platform is most appropriate for the scope of the project. It compares the three market-leading systems in respect to the applicability for the MPAT framework and also takes the building of a CMS from scratch into account. Finally a conclusion is drawn which solution for building the MPAT architecture is the most viable.

2 Requirements from scenarios

2.1 User Scenarios

A first set of scenarios was agreed and described in document D2.1 Initial MPAT Scenarios. At the WP2 meeting in Finland we decided to focus for the first year on eight of the most important scenarios described in D2.1. After a simple vote, the chosen list was:

- Added/Additional Information
- Social Media Feed
- Selected camera view / 360° video
- Move interactivity to the Companion Screen
- Play Along
- Betting Game
- Live stream Overview / Commenting the show / Event live streaming with social media
- Interacting with commercials

Two more scenarios were added later as a special request by partners who were already committed to working on those two:

- Ad insertion/replacement
- Polling / voting

2.2 Components of User Scenarios

We have also gathered a set of ideas for useful modules that partners were willing to work on. The list of modules is:

- Screen Overlay / Pop-up: this module deals with situations where information needs to be conveyed to the viewer as a dialog that pops up in front of what is on the screen, until the viewer closes it.

- Device Pairing & Communication: this module deals with the pairing of companion screens with the TV, and then the communication between the companion screen and the TV, including on HbbTV v1.x devices which do not have support for it.

- Bookmark functionality: this module covers bookmarks, reading list, favourites and other such names, i.e. the ability of the viewer to have a link to some content stored for later perusal. The key functionality is the persistent storage for authenticated users.

- Video player (Livestream, Broadcast, VoD): this module deals with controls appropriate for each version of video, e.g. for livestream and broadcast there are no forward and backward buttons, no interactive progress bar.

- Social Media Feed: this generic module deals with presenting a stream of rich text information filtered from social media sources.

- Notification: this module deals with the event notification of the viewer. A viewer can choose from a list of events which to be notified about. The system may choose to notify the viewer about program-related events (recommendations).

- Video List Menu: this module deals with presenting many links to videos, e.g. in a VoD store.

- Interactive view (panning,scroll/360 view): this module deals with the interaction required of consuming a 360° video; the viewer sees a particular point of view, and may interact with the TV to change this point of view, e.g. by clikcing on the arrows of the remote control to see further left or right.

- 2nd Screen application: this module deals with the launching of the companion screen application on a paired device, as well as moving content from the TV screen to the companion screen and back.

- Single-device multi-user management: this module deals with the fact that a TV is used by multiple viewers, and gathers information to inform the server on who could be currently viewing, by any means, e.g. which companion screen is currently interacting with the TV.

- Festival Schedule (EPG like): this module deals with the presentation of EPG-like information, and how to interact with it.

- User/device identification through network information: this module is related to device pairing and communication and to single-device multi-user management.

- Display results/scores on HbbTV / performance statistics from previous games: this module deal with the presentation of sports information on the TV.

- Invite/select players/competitors / squad building interface: This module deals with situations where a team of viewers needs to be assembled for a game, for common viewing, etc. The module deals with gathering candidates from e.g. social networks, with presenting the candidates and for interacting to choose the team.

- User Authentication and User Data Management: this module deals with situations where users need to be authenticated, e.g. in a VoD store, and to keep track of what the user has done (list of movies already bought)

- List selector (used to be “check box for selecting players” in earlier deliverables): this module is a more generic version of above modules.

- Polling: this module deals with presenting polls, interacting and sending the voting to the server.

- Application launcher: this module deals with the presentation of the red button some time during a longer period, interacting to unfold a launcher menu, presentation and interaction with the launcher menu, usually a flat carousel of icons and text.

We have then determined which functionality was required by which scenario, to deduct the development priority. The table on the next page was used to identify the overlapping of the modules and carving out the synergies between the scenarios.

Use Cases / Required functionality	Additonal info	Social Media Feed	Selected Camera View / 360° Video	Move interacity to CS	Play along	Betting Game	Interacting with ads	Livestream overview	Ad insertion / replacement	Polling / Voting
Screen Overlay/Pop-up	✓	optional		✓	✓	✓	✓	optional		✓
Device Pairing Communication	✓			✓	✓		✓		optional	
Bookmark functionality	optional						✓	✓	✓	
Video player (VoD, Live, Broadcast)		✓	✓					✓	✓	
Social Media Feed		✓						✓		
Notification	✓		✓	✓	✓		✓	✓		
Video List Menu			✓					✓		
Interactive view (panning,scroll/360 view)			✓					optional		
2nd Screen application	✓			✓	✓		optional			
Single-device multi-user management	optional			✓	✓					
Festival Schedule								✓		
User/device ident. through network info.				✓	✓		✓		✓	
Display results/scores					✓	✓				✓
Invite/select players/competitors					✓	✓				
User Auth. & Us. Data Mgmt					✓	✓				
Performance statistics					optional	✓				
List selector					optional	✓				
Polling										✓
Application launcher		✓	✓			✓			optional	

Table 1: Scenarios vs. functionality

2.3 Components of Content Creator Scenarios

We have also worked on Content Creator Scenarios. From these, we have identified features that do not derive directly from User Scenarios, or at least not obviously.

Event support

Event support is different from time support below. Event support is about the ability of the author to plan for the application to react to an HbbTV event, not planned at a specific time. However, user experience may end up as similar to “time support” in the editor.

Time support

The authoring system from which MPAT is inspired, HAT, does not have time support in the HbbTV application. The HbbTV application designed with HAT is constant over the lifecycle of the TV show. MPAT intends to add “time support” in different ways:

- Some elements of the MPAT application can be available at a certain time in the show, or from a certain time and for a certain duration, all decided by the application author. This can be achieved simply with browser-related time, or with HbbTV stream events.
- Trigger time and duration may need to be decided at the broadcast time. This requires the support of HbbTV stream events.
- Editing time related elements will be easier if a timeline of the broadcast video is presented to the author in the form of a band of thumbnails of key frames of the video, with markers where applications elements appear and disappear.

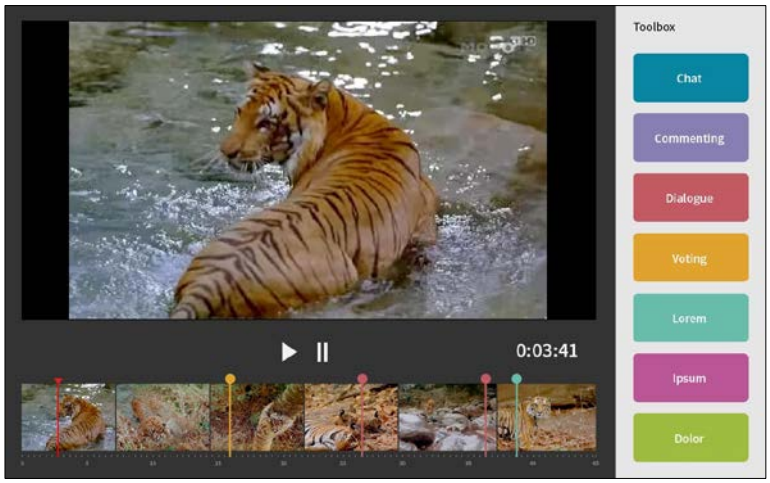


Figure 1: Video timeline

Playout

The playout is a mode of the authoring tool when the TV show is simulated on the author's machine. If the video already exists (i.e. not a live show), the final video is played in the background. In case of live broadcast, an example video is played instead, or a background image is used. Interactivity is available with the keyboard instead of with the remote control. Time-dependent events can be simulated, but when relating to a broadcast stream external hardware for proper testing on a TV set will be required (e.g. when dealing with stream events which have to be signalled in the broadcast stream).

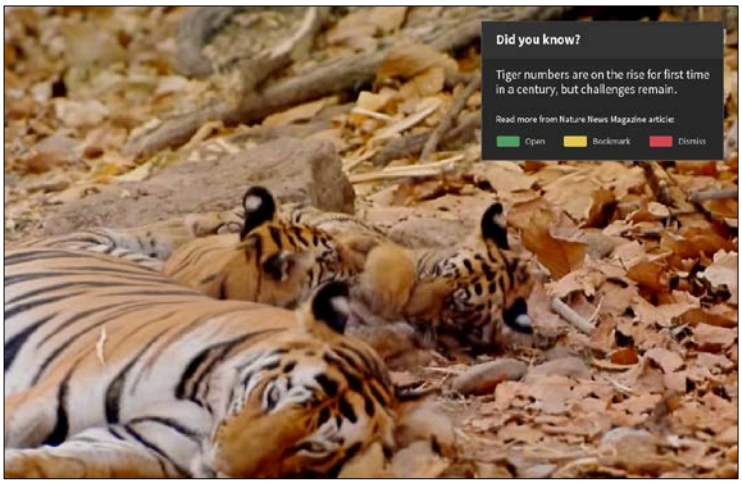


Figure 2: Example playout with HbbTV dialog

After the consortium had a closer look at the scenarios from WP2 and decided on a technical concept, it was discussed on how to achieve these goals and by which means. The initial idea of MPAT, as already stated in the description of work, was based on the solution of a content-management system. The next two chapters describe, if it is more feasible to build MPAT on an already existing CMS or to build a new system from scratch.

3 Review of existing CMS solutions

The following section focuses on a state-of-the-art analysis regarding different CMS solutions which are capable of being a suitable system to build the MPAT platform. The evaluation will consider technical aspects as well as business-oriented aspects. The technical oriented analyses will ask whether the existing solution can speed up the development of MPAT while still meeting the requirements for keeping the whole system flexible for extension. Three of the most popular CMS solutions were selected for an evaluation, namely:

- WordPress
- Joomla
- Drupal

The analysis starts off with the most popular and widespread CMS, which is WordPress.

3.1 WordPress

WordPress is an open source Content Management System (CMS) to build websites, blogs or applications. The web software is used by over 60 million users and is built by hundreds of volunteers and includes thousands of themes and plugins in their marketplace to generate personalized sites. WordPress is also the leader of CMS (April 2016) worldwide with 59,3%.

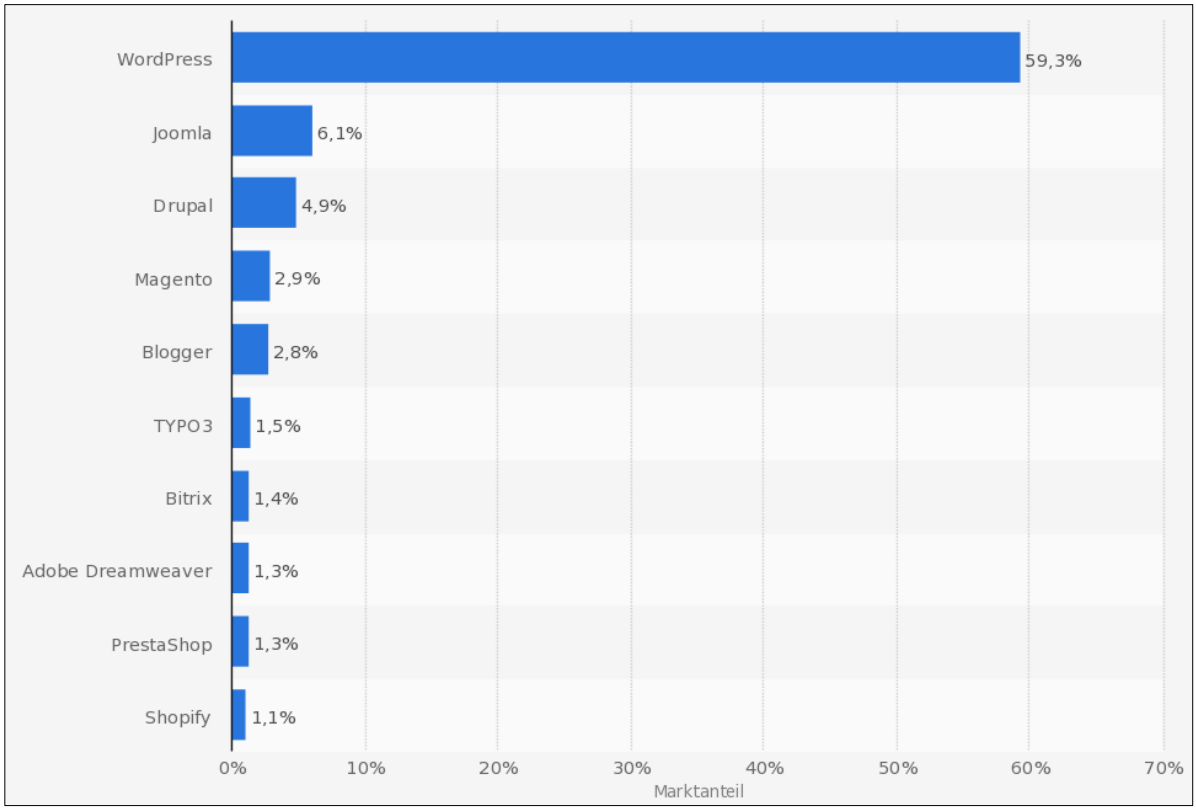


Figure 3: Market Share of Content Management Systems in April 2016 [CMS16]

Also 26,4% of all sites in the web are built with WordPress. Still, for 55,6% of the websites the underlying CMS can not be automatically detected. [W3T]

Architecture

The following diagram and below describe the Wordpress architecture and its components in detail:

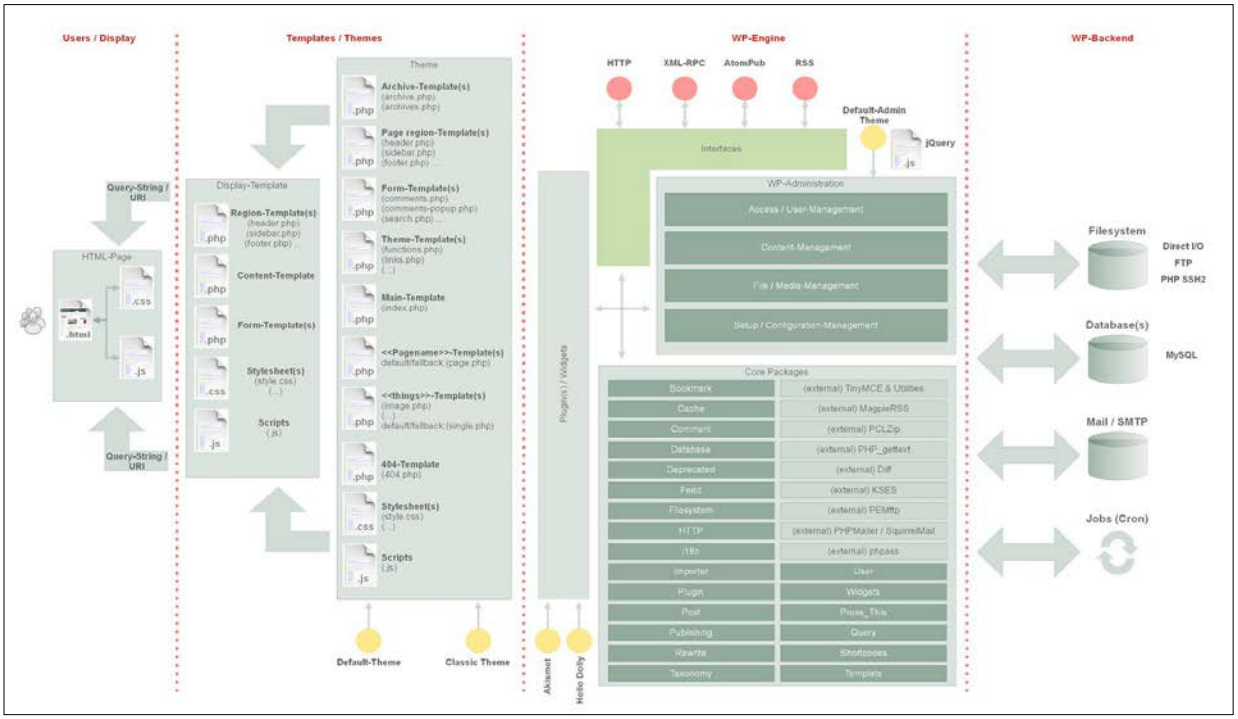


Figure 4: WordPress architecture diagram [WPA16]

WP-Backend - holds the MySQL database, the File System and the Mail/SMTP

WP-Engine – includes the core elements of WordPress: WP-Administration, Core Packages, Plugin(s) and Widgets, Interfaces and jQuery. By the installation of WordPress an admin-default theme, as well as two plugins will be installed automatically.

Themes / Templates – handle the construction of a website, blog or application. A theme includes various templates that are responsible for the design of a page. One theme can have 1-to-n templates. Also WordPress can have more than one theme installed, but only one theme can be activating in one site.

User / Display – displays the generated website, blog or application. This can be a HTML-Page with HTML, CSS and JavaScript.

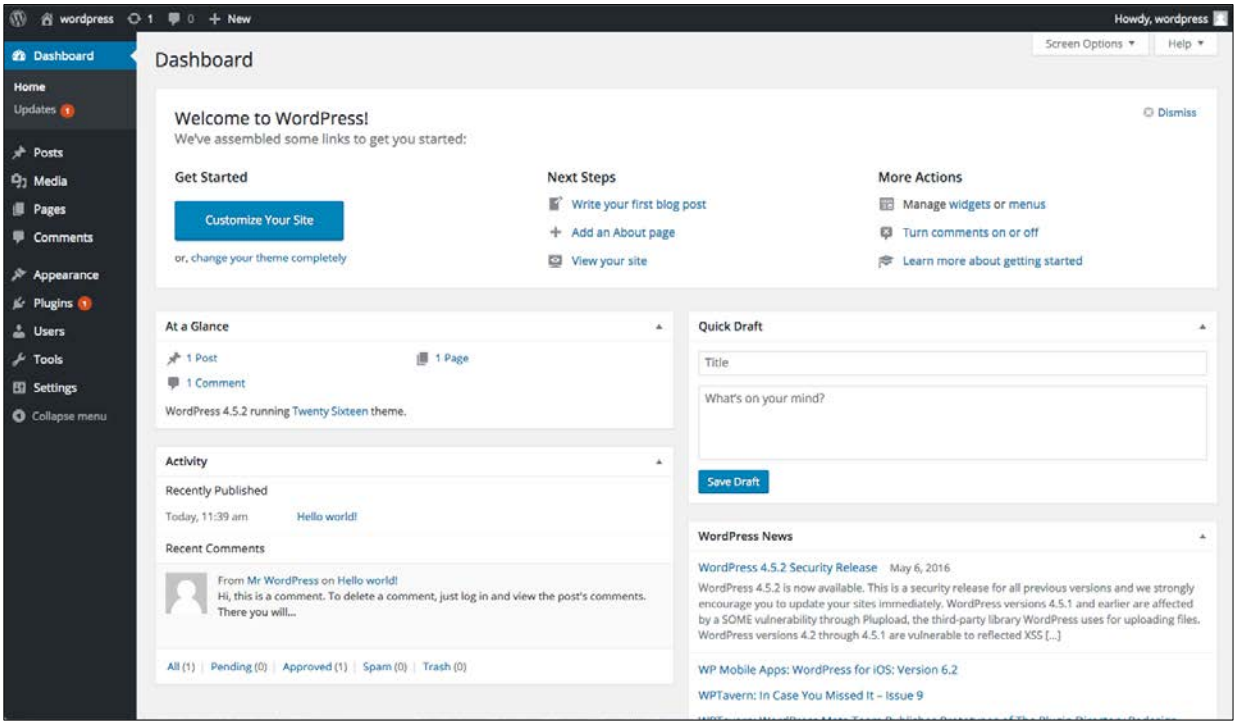
Preparation aspects

The WordPress system built on MySQL and PHP and is licensed under the GPLv2 (or later). WordPress recommends a PHP version of 5.6 or greater and a MySQL version in 5.6 or greater. Another possibility is running WordPress with MariaDB version 10.0 or greater as the database. The newest WordPress version is Coleman in 4.5.2 (18.05.2016).

Core System

The WordPress Core is developed by the WordPress Core Team which consist of the WP co-founder Matt Mullenweg, five lead developers and eight core developers with permanent commit access. Since WordPress is an Open Source project, external developers can contribute to the core. The Core System can run on different devices. A responsive web application runs on every modern browser. The iOS and Android applications handle the native mobile devices.

By installing WordPress on a web server a default setup with the core features and some core templates and plugins are already there. Also a sample page, post and comment function will be provided by the



installation.

Figure 5: Backend view of Wordpress

Basic modules

The modules included in the WordPress core system are the following [COR16]:

Posts – were designed for blogs and news websites. A post is a short page that will be displayed in the blogroll

Media – this is the repository for every media file like images and videos

Pages – page is built by a template, here the actually textual content will be created

Comments – this module can be active or not. The comments can be set under every page or post

Appearance – handles the themes, customizer, menu and editor for every activate theme

Plugins – in the core system WordPress pre-installed two not activated plugins

Users – WordPress can deal with users in different user roles. The user roles can be

- Subscriber
- Contributor
- Author
- Editor
- Administrator

Settings – General Settings of the WordPress Core

Templates – Layouts for Pages

Customizer – Styles for Pages

Extensibility, adaptability and flexibility

WordPress as an Open Source project says: ”With WordPress, you can create any type of website you want: a personal blog or website, a photo blog, a business website, a professional portfolio, a government website, a magazine or news website, an online community, even a network of websites. You can make your website beautiful with themes, and extend it with plugins. You can even build your very own application.“ [WPF16]

WordPress allows to build themes for different web content. For MPAT, we require that it is also possible to build HbbTV related applications by developing an HbbTV conform theme. Also Plugins and Templates can be developed to create an HbbTV component. WP is complete, extendable, adaptive and flexible for the developing process. It provides also a multisite feature, which can handle more sites in the same WordPress instance.

User management

WordPress provides a user management by the concept of user roles, designed to give the site owner the ability to control what users can and cannot do within the site. The site owner manages the user access to tasks creating pages, write and edit posts, etc. By the installation of WordPress, an Administrator account is automatically created. There are in total the six following user roles:

- Super Admin – somebody with access to the site network administration features and all other features.
- Administrator – somebody who has access to all the administration features within a single site.
- Editor – somebody who can publish and manage posts including the posts of other users.
- Author – somebody who can publish and manage their own posts.
- Contributor – somebody who can write and manage their own posts but cannot publish them.
- Subscriber – somebody who can only manage their profile.

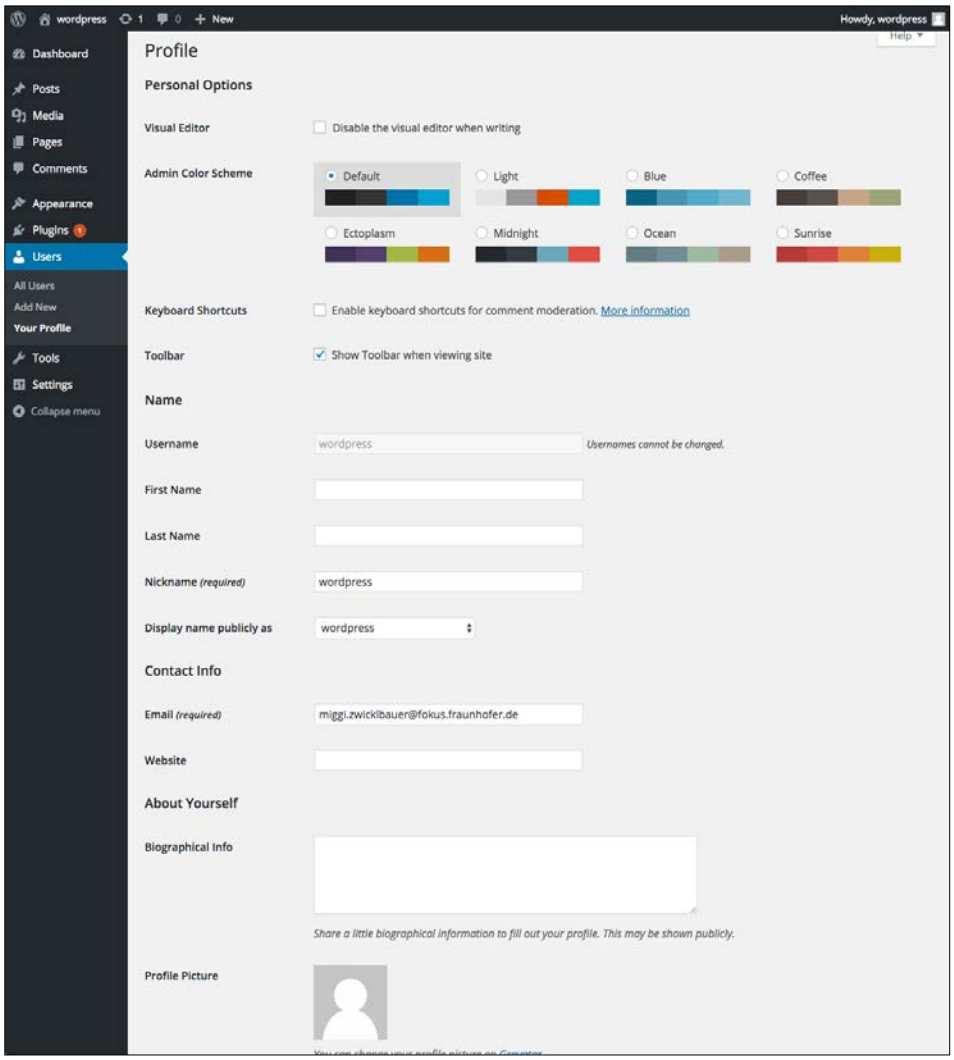


Figure 6: User profile view in WordPress

Usability

Since WordPress is a web based CMS, the user only needs an internet connection to do changes. With the help of native applications of the CMS the user can also edit the content via smartphone or tablet.

WordPress got a very simple and intuitive design on the editor view (web solution and native app). Depending on the active theme, the backend can extend or minimized by only displaying necessary options to the user. Also with the benefit of jQuery e.g. Drag & Drop functions can be easy handled to create an application.

Performance

The performance of the WordPress backend depends on the server where WP is installed. The most important thing that is responsible for the performance of backend and frontend are themes. Depending on a theme the editor (backend) can be manipulated by add extra non WordPress Core functionalities, that may be boost up or down the performance of WordPress. Such as Themes Plugins can have effects of the performance. There are a lot of Plugins that improve performance like e.g. minifying JavaScript code.

Also the updating of the WordPress System is helpful to optimize performance of the CMS.

Performance mostly depends on active Theme and Plugins. WordPress Core System should be updated. Plugins for performance optimization:

- Code minimizer
- Media Management

Security and privacy

As WordPress is the most popular CMS with over 59% market share worldwide the security and privacy is one of the most important topics in the development of WP. A WP Security Team (up to 25 experts) works to identify and resolve security issues in the core software available for distribution and installation at WordPress.org, as well as recommending and documenting security best practices for third-party plugin and theme authors. This work happens in collaboration with the WordPress Core Leadership Team and backed by the WordPress global community.

The WordPress release cycle lasts around 4 months from the initial scoping meeting to launch of a new version.

Licensing

WordPress is released under the GPLv2 (or later) license from the Free Software Foundation. Part of this license outlines requirements for derivative works, such as plugins or themes. WordPress also points out, that there are some „grey areas“ in their licence model regarding the implementation of plugins and themes and thus inherit the GPL license.

Reliability and recoverability

WordPress provides a lot of **Backup Plugins** that handle the recoverability of the database including the content. Next to the Plugins that can be used for Backups there are two more options to recover the WordPress content:

Manual Backups: Can be done via phpMyAdmin for the database and with the help of FTP for the content. This kind of backups can be useful for smaller and non-professional websites.

WordPress Backup Services: There are a lot of Companies at the market that deal with Backup solutions for WordPress. There are two different kinds of backups. The normal backup functionality and a migration service. Migration Services are helpful to migrate the WP into another server or domain.

3.1.13 Summary

When analysing WordPress for its capabilities to be used as a platform for the MPAT system, the following conclusion can be drawn:

Pros:

- Easy-to-use for non-developers and users
- Most used content-management-system
- Biggest marketplace with themes and plugins
- Easy to expand the functionality
- Biggest community of developers and designers
- Growing market of business models around that ecosystem

Cons:

- Limited to MySQL or MariaDB as Database
- No integrated export function for generated websites/blogs/applications

3.2 Joomla

Another popular CMS is Joomla, which started in 2005 and is mostly targeting at middle sized to large websites. With an estimated 2.8 million websites, the CMS ranks as second in popularity behind Wordpress, but not in close distance to it. Famous companies such as Porsche, Danone and Nickelodeon use it for their web appearance.

Architecture

The core system of Joomla is a Model-View-Controller web application, which makes use of various system layers as shown in the figure below:

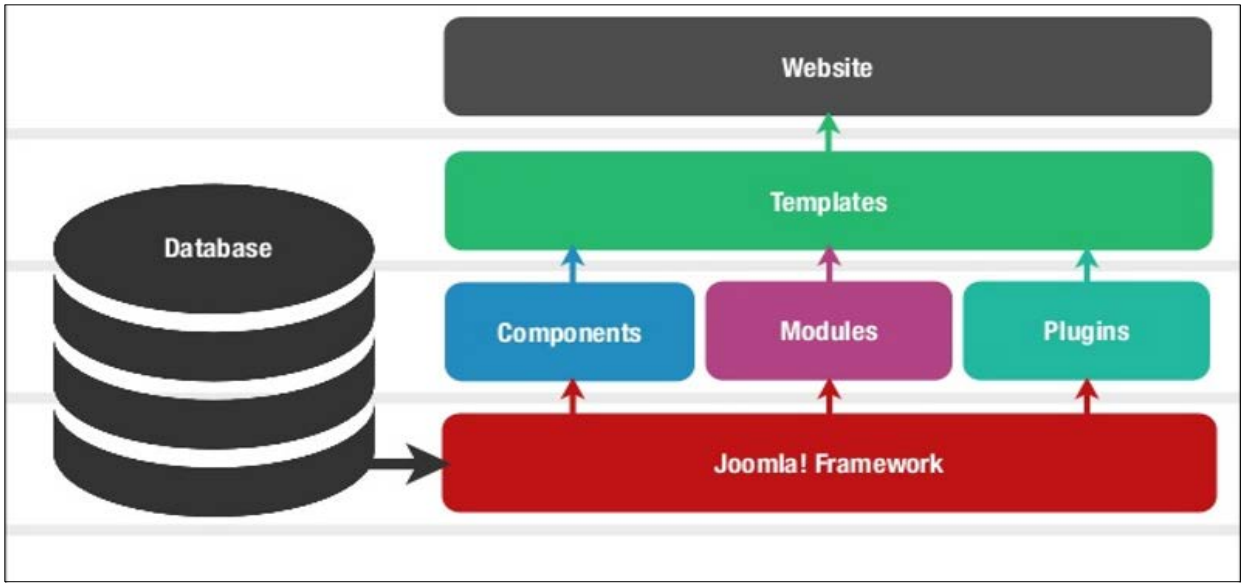


Figure 7: Architecture overview of Joomla [JOA14]

The wording for each layer in the system is often quite similar compared to other CMS, but sometimes has a different mind-set to it, therefore these layers shall be described in more detail [JOA14]:

Database − Database is a collection of data and can be stored, manipulated and organized in a particular manner. The database stores the user information, content and many more required data of the site. It is used to store the administrative information to manage the Joomla site. Using Joomla database layer, it ensures flexibility and compatibility for extension.

Joomla Framework − Framework is a collection of open source software, where the Joomla CMS is built. It is developed to break the framework into single modular packages which helps each package to develop more easily.

Components − Components are considered as mini applications. It consists of two parts i.e. Administrator and Site. Whenever a page gets loaded, component is been called to render the body of main page. The Administrator part manages different aspects of the component and the site part helps in rendering the pages when request is made by site visitor. Components are main functional unit of Joomla.

Modules − Modules is an extension which is used to render the pages in Joomla. It is also used to display the new data from the component. It frequently looks like boxes such as login module. In Joomla administrator the modules are managed by the module manager. It displays the new content and images when module is linked to Joomla components.

Plugin − This is also a kind of Joomla extension, it is very flexible and powerful for extending the framework. It contains a bit of codes that is used to execute the particular event trigger. It is commonly used to format the output of a component or module when a page is built. The plugin function which are associated with event are executed in a sequence when a particular event occurs.

Templates − Template determines the look of the Joomla website. There are two types of templates used i.e. Front-end and Back-end. The Back-end template is used to control the functions by the administrator where-as the Front-end template is a way to present the website to users. Templates are easy to build or customize your site. It provides maximum flexibility to style your site.

Web Browser − It is a server where the user interacts. It delivers the web pages to the client. The HTTP (Hyper Text Transfer Protocol) is used to communicate between the client and the server.

Preparation aspects

Joomla is running on Apache- and Microsoft IIS-web servers with PHP from version 5.2 on and needs a MySQL installation starting from version 4.1, which the one-file-installer covers. After downloading the installation package the user has to copy the related files to his designated web space. The installation routine guides the user through the setup process by asking for the required access rights on the system (administrative user rights are mandatory) as well as setting up the MYSQL database. On the whole the process of setting up the core system takes not much longer than 10 minutes if the necessary information and user rights are available.

Core System

Joomla structures its backend into “segments” and “categories”. The latter matches a sub-page structure, in which single posts can be created out of text and pictures with the help of a WYSIWYG-editor called TinyMCE. The editor can also be replaced by any other 3rd party editor if necessary. Before publishing the user must upload any media to the CMS gallery and from there the editor gets access to the available picture and other media. Components represent larger structures, as for example a contact database, voting systems or even a complete community of users. Multiple pages are made available to the front end via a separate menu entry in the backend. Via components, modules and plugins the function range can be extended.

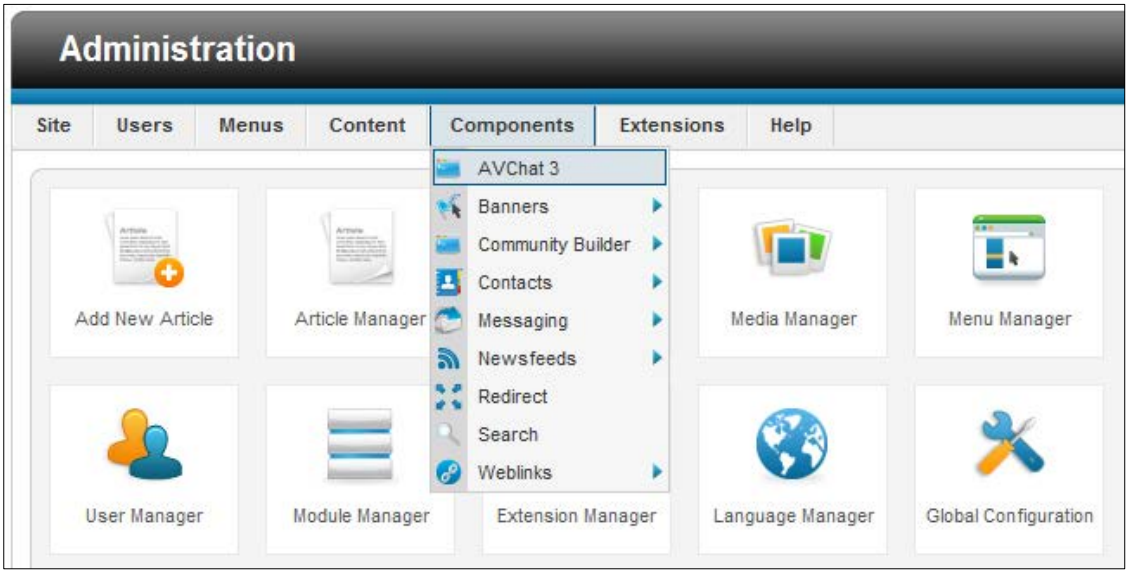


Figure 8: Administration view of the Joomla

Basic modules

The modules included in the Joomla core system are the following [COR16]:

Templates - Change the site design template (look and feel)
Modules - Install third party programs or functionalities
Multiple-user - Multi-level permission user content creation and editing
Multiple-level menu system - Main menus, sub-menus
Pages - Add / edit text, image, and other media content
Polls - Provides the capabilities to capture votes on different topics in the form of multiple choice questions
Search - Provides site-wide keyword searching
Article - Provides the capabilities to create and archive articles
News Feed - Provides syndicated content (RSS, RDF, and Atom feeds)
Banner - Provides banner advertising functionality
Contacts - Provides contact management capabilities
Web links - Provides management controls for controlling Web Links

Like every CMS, a module for writing and editing text and images is already an integral part of the Joomla core system and most of its features would be reusable for MPAT. However there's a limited support of editing video in the sense that is needed for broadcast specific requirements. But this is probably the case for all reviewed CMS solutions. Other features such as responsiveness are also included, but need to be expanded to work responsively with the HbbTV devices.

Extensibility, adaptability and flexibility

As already mentioned, Joomla offers five varied categories of extensions to the users for improving the site content. It includes plugins, components, modules, templates, and languages. However, each of these modules differs in functionalities and capabilities and the modules offer only small building blocks, which are often referred to as widgets in other CMS. With this the editor can arrange those assets with few clicks in the sidebar of the website. Typical modules are weather forecast or user statistics.

In Joomla's case, plugins are extensions, which can manipulate the system and function automatically in the background, e.g. authentication of a login or a search function. In the German webstore edition overall 650 components, modules and plugins are available. More than 4,200 in the English variant. As any other CMS, Joomla also offers a dedicated interface for the website-design, here the user can select from different design templates and implement them on a global basis. A deep manipulation tool for the design is not delivered and has to be adapted manually via HTML, CSS and even PHP. Because of its common and adhering certain norms, the various design templates look very similar. For example one or two vertical columns have to be created for menus and modules, which leads to a relatively static design. Apart from this the user can select from around 1,200 different templates via the Joomla platform, some of them are free and some have to be paid for. The user has to define a FTP-account for directly uploading add-ons, such as plug-ins [CDJ16].

Other extensions which will be needed in MPAT, such as managing multiple sites within one Joomla installation is also possible (e.g. Mighty Sites). The system can also be expanded with other functionalities needed in MPAT such as a social-media plugin.

But the advanced customization might be a problem for novice users, because if you want to use the advanced features of Joomla in full-fledged way then you sometimes need to take assistance from a programmer or developer.

User management

Joomla has strong content management capabilities, designed as an enterprise grade Content Management System, Joomla offers a wide range of content management capabilities, which allows to handle extensive bulk of content conveniently.

The user management handling is clearly arranged and distinguishes between three user groups for front- and backend. Neither user groups nor rights management can be edited in very much detail. A further deep intervention of those assets have to be dealt with in a more sophisticated add-on. The user of the backend also can communicate via an own chat system and also a mass-mail function is included where one user can send messages to selected or all other users. The ACL (Access Control Line) is a set of permissions granted to particular set of users for accessing specific pages, which is offered by almost all enterprise-grade CMS solutions. However, Joomla lacks this functionality and developers can use this feature only if they have version 1.6 or onwards installed on their computers.

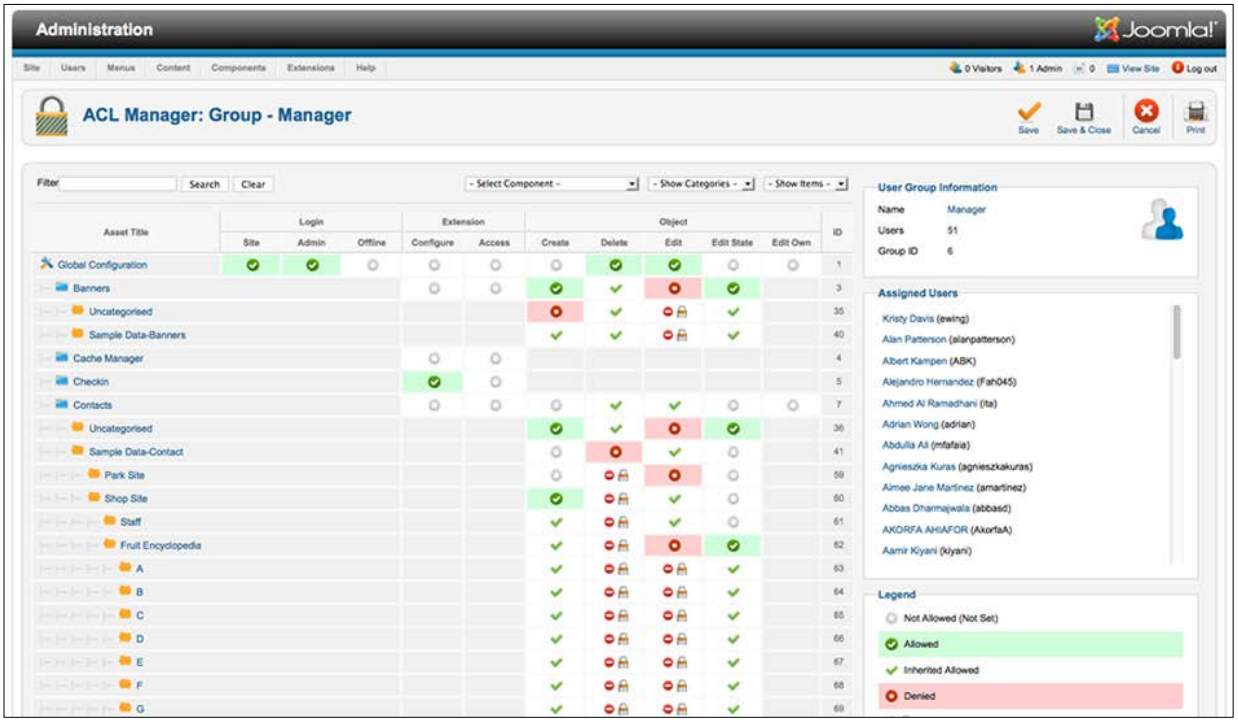


Figure 9: User management view in Joomla

Usability

The User/Editor Interface of Joomla credits to its popularity. Although a little learning is required to figure out the backend appropriately but when compared to its rivals, Joomla with flexible, powerful and polished UI, which makes easy for the new publishers also to publish the content easily. For example the options in the media manager are clearly structured and there's also an overview of the currently used modules, which can be edited with only one mouse-click.

Accessibility

The web community has moved a great deal forward in terms of accessibility and people have increasingly had an open ear on accessibility issues. Due to several new features of HMTL5 the following features are supported in Joomla:

- Separating content and presentation (layout and formatting with CSS)
- Semantic and logical structures
- Careful coloring
- Consideration of minimal contrast ratios
- Variable font sizes
- Fluid or elastic layouts
- No images of text
- No transparent backgrounds for images
- Making sure that navigational elements are perceivable
- Making sure events are not mouse-triggered, but can be triggered also with a keyboard
- Highlighting focus (keyboard)

Performance

Joomla itself is not slow, one needs to tweak settings and use the available inbuilt options to boost up performance. If Joomla developers were to optimize Joomla performance by default, then it would be difficult for the user to customize Joomla and much more difficult to make functional changes. It's up to the application developer to get the maximum out of Joomla by tweaking settings and optimizing Joomla framework. Default Joomla has lot of potential to be optimized for speed and performance [JOS16]

- Speed and performance of Joomla is largely dependent on the choice of template, followed by add-ons components, modules and plugins.
- The number of CSS images, Javascripts (size)
- Activation of inbuilt Gzip and Cache system does speed up the Joomla site
- Cleaning up of CSS codes can result in a highly boosted Joomla performance in terms of decreased http requests and total page size
- Compression and modification of CSS / JS files, similarly compression and optimization of pics / images is the key to speed up default Joomla
- As a demonstration we called two CSS files with one line of code resulting in 1 less http request

Security and privacy

The security of a site is a major factor to consider while choosing a CMS. The high risks in the current digital world which mainly attributed to hacking and cracking of the site are a concern to all. The absence of regular code snippets and downloadable security plugins makes Joomla less secure. It has a general setting which is more prone to attacks. The security forums are general and therefore exposure of security details of the site.

Also the availability of plug-in will determine to interpret and the security of a site. A secure site should have the most recent security plugins which should allow compatibility with previous versions for easier coding.

Licensing

Joomla is open-source software which is available under the GPLv2 license. Meaning the usage is free, the code can be altered in any kind, it can be redistributed freely without any limitation and it is also possible to sell software code which is built with or around Joomla.

Therefore the system can be used for commercial and private purposes as long as the GPL license is passed on, allowing other developers the same four mentioned rights for their creations. This would be suitable for the purpose of the MPAT platform in order to establish an open-source ecosystem with a market-place for plugin and theme developers including selling themes and plugins.

Reliability and recoverability

Joomla has a weaker developer community than Wordpress, which continuously launches free of cost open source plugins and extensions. The number of support articles for Joomla is less than a thousand. Meaning the forums and documentations found in the web are not up to the vast fund which is available for WordPress.

3.2.1 Summary

When analysing Joomla for its capabilities to be used as a platform for the MPAT system, the following conclusion can be drawn:

Pros:

- Editing content and structure on a Joomla site is very intuitive and can be compared to editing a Word document
- Joomla has a handful of available extensions for e-commerce, which allows to manage commercial products and content all in one place, which would be handy for building an MPAT ecosystem
- The learning curve for building a site is relatively low, allowing a non-developer to understand and utilize the website in a short amount of time

Cons:

- User management is limited and does not allow granting access for certain pages to specific users
- Unlike WordPress, Joomla has a much more limited marketplace for additional modules and add-ons
- Joomla's SDK is more complex than other CMS, making it difficult to incorporate own custom designs without developer experience and build a MPAT ecosystem
- Not many code snippets or documentation for developers available, therefore it's much more harder to target 3rd party developers

3.3 Drupal

Drupal is an open source CMS, released under GPL license, maintained by a community of over 100000 users. Last major version, namely Drupal 8, was released in November 2015. One of the main improvements is the adoption of some components of the Symfony framework, a well-known PHP framework which is widely adopted in professional development of web applications.

Looking at usage rate, Drupal ranks as the third most used CMS in the web, behind Wordpress and Joomla. Since the main target is enterprise and community sites, Weather.com with its 100 million people per month audience is a perfect example of Drupal scalability, while Telegraph Travel Guides App represents the potential of the CMS as a web service for mobile apps.

To date, over 1000 modules are actively maintained and already compatible with Drupal 8 and the number is set to grow in the next months.

Architecture

Drupal uses an abstract management of web content and functionalities, using separated layers it makes the content management more flexible.

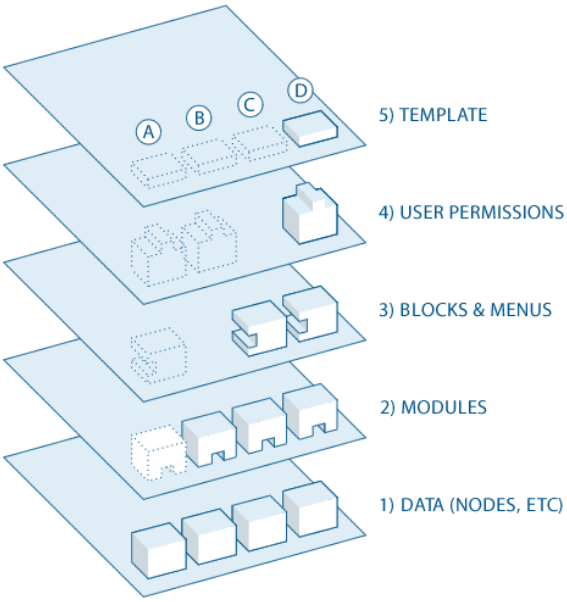


Figure 10: Architecture overview of Drupal [DRU16]

Data - Data layer represents the whole set of contents of a Drupal instance. From articles to users, every kind of information that require to be persisted in the database is a part of the data layer.

Modules - Modules represent functionalities which are available in Drupal. The trend in last releases is to move functionalities like comments, file, forum and entities from core to modules to let the user to activate only needed features. Like other CMSs, there is an official repository for contributed modules created by the Drupal community, while the official documentation guides users through the creation of custom ones [DOR16].

Blocks and menus - Blocks provide output both from the data layer and from custom logic from modules. They can be placed in different areas of the theme and rules can be defined to regulate visibility on a context basis. After the adoption of Symfony routing component in Drupal 8, Menu system define only the collection of resources provided by modules and navigable on the website.

User permissions - In between of the site structure and its visual representation there is the user management system. Like other CMS it defines growing permissions for common roles like anonymous, authenticated, editor and administration users for both frontend and backend parts of the web application. Drupal also includes APIs to define custom roles and permissions.

Templates - On the top layer reside the site theme, a set of templates, stylesheets and scripts which define the visual representation of CMS contents. Since version 8, Drupal use Twig as template engine with the aim to achieve a better separation between logic and presentation levels. Compared to pure php files, it has a simplified syntax, helper functions and filters and multiple inheritance of templates [TTE16].

Preparation aspects

While Apache is the recommended web server, since it is the most used, Drupal 8 officially supports also Nginx and Microsoft IIS and theoretically every web server with proper PHP support. Similarly databases with official support are MySQL, MariaDB, PostgreSQL and SQLite, while Microsoft SQL Server and Oracle require additional modules to be used. The integration of Symfony components in Drupal 8 core implied also to increase PHP requirements from 5.2.5 to 5.5.9. [DSR16].

Configuration of a new Drupal environment is pretty straightforward, similarly to other CMS. After setting up web server host and creating the database, Drupal proposes an interactive wizard to define site parameters like system language, database credentials and first administrator user identity. During setup user can choose between two different configurations: minimal, which activate only basic core modules, and traditional, which enable more modules to improve usability and features. In both cases, modules can be activated and deactivated at a later time. In the end, the wizard performs some checks over project files permissions recommending eventual changes to enhance security.

Attention to professional development could be also seen in ways users can setup their instance. Besides standard archive-based installation, user can fetch drupal core from official GIT repositories, composer packages or via Drush and Console, two CLI interfaces, the former derived from previous versions, the latter built on top of Symfony Console component.

Core System

Standard profile is recommended during previously described wizard, since it configures common features like articles and images and preload the standard backend theme.

When enabled users are logged in, Drupal exposes an adaptive administration toolbar with quick links to the main backend sections.

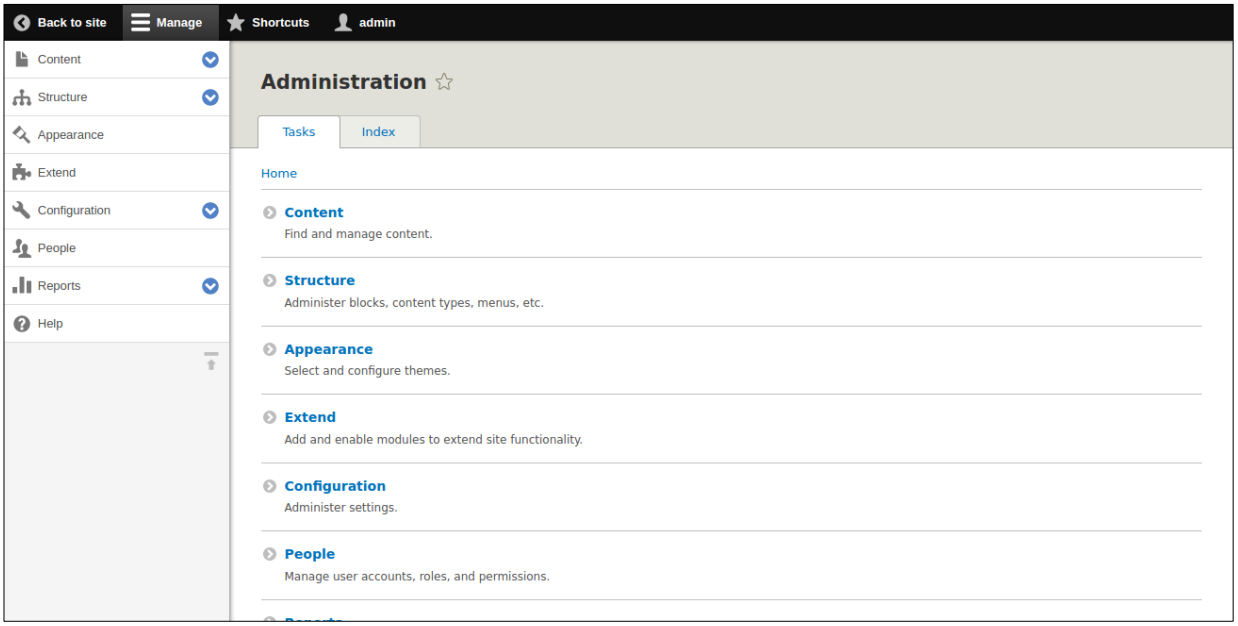


Figure 11: Administration view of Drupal

- Content: every content of the site is listed in this section. In standard instances, Drupal has four main content types: articles, pages, files and comments. Since latest major release, namely Drupal 8, CKEditor a WYSIWYG editor is enabled by default, but other editors like tinyMCE can be installed. To improve even further the content editing process, inline editing feature of CKEditor was also implemented to let editors change contents directly from the frontend.

- Structure: Site structure configurations from core, contributed and custom modules are specified in this section. By default, following subsections are available to administrators:

    - Block layout: given all the site sidebars and regions, user can choose which blocks will be displayed inside them.
    - Comment types: users can manage form and display settings and relation to content types for comments inside Drupal site. This is an example of structure settings defined by a module (comment module from Drupal core).
    - Contact forms: used to create and manage forms to collect data from site users.
    - Content types: Create and manage fields, forms, and display settings for site's content.
    - Display modes: display modes define the way contents and forms are displayed in the frontend, in this section users can configure and create these properties.
    - Menus Manage: both frontend and backend menus can be edited in this section, adding and removing links and even changing the menu items hierarchy.
    - Taxonomy: Manage Tagging, categorization, and classification of content.
    - Views: Views used to be a contributed module, now included in Drupal core, which define custom lists of contents. In this section users can define and manage them through an exhaustive UI

- Appearance: this section allow users to assign themes to frontend and backend interfaces choosing from available ones. It is also possible to install new themes from external URL or archive upload, but since Drupal lacks of an integration with official marketplace, the whole experience is less

intuitive when compared to other CMS like Wordpress. Other options control the default display settings for the entire site, across all themes, unless they have been overridden by a specific theme.

- Extend: this section is dedicated to modules which extends Drupal core. They are organized in different categories based on their nature. Similarly to themes, new modules can be installed, even if the lack of integration with official marketplace makes the whole process a little rough. To be able to install themes and plugins from the backend, Drupal requires FTP access to installation server.

- Configuration: in this section administrators can manage settings about user accounts, text formats and editors, performance such as caching and bandwidth optimization, site settings like name, email address, error page. Also configuration from modules which not define custom backend sections are shown here.

- People: this section allows administrators to manage users, roles and permissions; register new users with a role. It is described deeper in “user management” subchapter in this document.

- Reports: different kinds of report are available such as recent log messages, authentication errors, top search phrases or themes and modules which need to be updated.

Extensibility, adaptability and flexibility

To improve flexibility, several core components have been redesigned as core modules in the last major releases. In that way users can enable only required feature during first configuration, excluding components that are not needed like comments or polls.

Like many other CMS, Drupal provides different ways to develop web sites, depending on the nature of the customization.

Themes are used to alter the appearance of the site. They can include template files, which define html representation to data coming from modules and database, and other assets like stylesheets and JavaScript.

Modules allow users to add functionalities and alter the behaviour of Drupal with different methods:

- Hooks: Specially-named functions that a module defines, which are discovered and called at specific times, usually to alter behaviours defined in core and other modules.

- Plugins: Classes defined to add micro functionalities which are discovered and instantiated by drupal core at specific times during request lifecycle.

- Entities: Special plugins that define new types of content or configuration in Drupal

- Services: Implementation of the Service pattern built on top of Symfony Services component, they are classes that perform basic operations which are available to other modules. For example, operations like accessing the database or sending an e-mail are performed through dedicated core services.

- Routing: Another method inherited from Symfony, expose functions to define and alter "routes", which are URLs that Drupal responds to, or alter behaviour of existing routes with event listener classes.

- Events: modules can register event subscribers, a specific kind of method which is executed by the core system when related event is dispatched during request lifecycle.

At last, installation profiles combine core Drupal, contributed modules, themes, and pre-defined configuration into one download. They provide specific site features and functions for a specific purpose or type of site, enabling users to quickly set up a complex, use-specific site in fewer steps than if installing and configuring elements individually. In the final phase of MPAT development, a specific installation profile could be defined to simplify the configuration of MPAT ecosystem in a production environment.

User management

User management in Drupal is achieved through roles and permissions. Every user has at least one role, and every role has one or many permissions. Out of the box, Drupal defines two types of users, those who are logged in and those who are not. For that reason two roles are defined, authenticated and anonymous. The exception is the user created during installation. He is also called as the superadmin and he is the only one that has permanent/non-removable access to everything in a Drupal site. Custom user roles and related can be defined both programmatically in modules and visually in People section in backend.

People section in the backend offers three distinct pages for users, roles and permissions. The latter is the most complex one. It organize information in a grid, displaying permissions in rows and roles in columns. Users can then assign or revoke permissions by interacting with checkboxes.

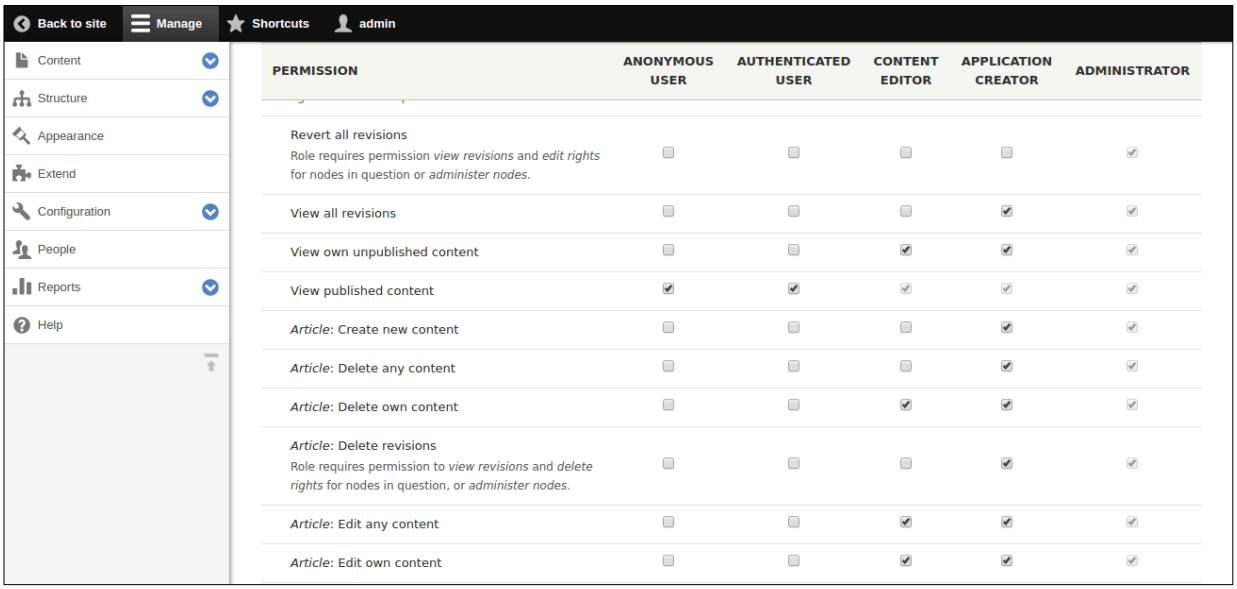


Figure 12: User management view of Drupal

What is actually missing in Drupal is the implementation of access control lists on content and user basis, to restrict permissions of specific users to specific contents, anyhow several contributed modules are designed for that purpose.

One important feature, especially from the extensibility perspective, is represented by the authentications providers. They allow developers to define authentication solutions, other than standard login-based one.

Contributed modules already exist to implement standard OAuth2. At last, authentication providers integrate with REST API to protect exposed resources when developing RESTful web services with Drupal.

Usability

Even though Drupal 8 addressed usability issues of previous versions, its backend is still less intuitive compared to other CMS like Wordpress mostly in the media management. If manual image editing is a concern, users have to install contributed modules to achieve even simple actions like crop and resize, since admin UI handles only the relation between assets and contents, however Drupal provides functionality to define image styles for automatic crop and resize. If the project requires deep customization of admin UI, developers can define a custom backend theme and assign it to the Drupal instance. Finally, in-place editing let users to edit articles even from the frontend, to have a clear idea of how they will be presented to final users.

Accessibility

Drupal 8 ships with extensive support for accessibility standards, and not only for colour contrast and font sizes. Semantic HTML5 helps to create interactions.

It marks content regions with descriptive attributes, thanks to WAI-ARIA techniques and HTML5 semantic mark-up. Navigation was also improved by identifying menus, banners, and ancillary content areas, and letting keyboard users move through them more easily. Drupal 8 provides a redesigned admin toolbar that adapts to different screen widths and exposes properties to screen readers to let every user to manage Drupal sites.

Performance

The adoption of different Symfony components brought lot of improvements in development, deployment and design process of web apps. But additional abstraction layers like HTTPKernel made Drupal core little worse than previous releases when looking at raw performance while rendering uncached pages.

Anyhow, Drupal 8 introduced some improvements regarding performances of production-ready applications. The most powerful one is the new dynamic page cache system. By adding cacheability metadata, developers and designers can define different rules to cache different parts of the page both for anonymous and authenticated users. Furthermore, Drupal defines APIs to integrate to existing reverse proxies like Varnish to achieve proper cache expiration of pages which include dynamic parts.

While still in a developing process, Drupal 8.1 introduced the support to Facebook BigPipe technology as a core module called Big Pipe. When dynamic page cache is enabled, default behaviour, and a page has to be rendered for the first time, the system substitutes dynamic parts (e.g. user related info) with placeholders and cache the result. Then, according to traditional strategy, every time the page is requested, it renders dynamic parts, merge them in the cached page instead of the placeholders and send rendered page to the client. BigPipe introduces a new strategy: cached main part of the page, is sent to the client as soon as it is requested, while dynamic parts start to be rendered. When ready, rendered blocks are streamed to the browser. Finally, a client-side component merge them in the page instead of placeholders. Even if total page load time does not differ too much between the two strategies, perceived one can be far less in BigPipe way. Considering that, official support to BigPipe is a very convenient feature when developing dynamic web apps which involve profiled users [PCR16].

Also, fairly common in the CMS landscape, Drupal implements CSS and JS files compression and minification and JS assets inclusion from the bottom.

Security and privacy

Architecture changes and core reengineering might lay Drupal 8 open to more bugs, when compared to a more "incremental" release strategy adopted by other CMS. To date, only one security bulletin reported 8.x vulnerabilities from "less critical" to "moderately critical" severity levels [DSB16].

Security features like CSRF, filters applied to WYSIWYG inserted contents, hashing of user passwords and session IDs and auto escape of theme code via Twig engine are included in Drupal core to strengthen security.

Anyhow, like every extendable CMS, a major role in application security is acted by contributed and custom modules. Developers have to pay close attention when choosing which modules are needed in their applications considering different aspects like: modules stability, closed/reported bugs ratio, continuous support from the developer, and the nature of the developer (renowned agencies instead of unknown users).

Finally, a special mention should go to the rewarded security bug bounty program. It pledges money rewards to users who discovers security issues such as XSS, SQL Injection, CSRF, Access Bypass, etc. and submit them to a specifically created platform. It is encouraging for Drupal 8 security, since in the last years, such kind of program has proved effective in other software projects [DSP16].

Licensing

Drupal and all contributed files hosted on Drupal.org are licensed under the GNU General Public License, version 2 or later. That means that users are free to download, reuse, modify, and distribute any files hosted in Drupal.org's Git repositories under the terms of either the GPL version 2 or version 3, and to run Drupal in combination with any code with any license that is compatible with either versions 2 or 3 [DLF16].

Summary

When analyzing Drupal for its capabilities to be used as a platform for the MPAT system, the following conclusion can be drawn:

Pros:

- Dynamic cache system provided by Drupal core and integration with systems like memcache and varnish provided by contributed modules, ensure high reliability and scalability to configured drupal instances.
- Adoption of Symfony framework components brought several improvements in development and deployment workflows, mostly appreciable by advanced developers.
- Authentication providers and RESTful API ensure consistent and secure management of resources in complex web applications.
- WYSIWYG and in-place editing components provide intuitive interfaces to content editors.
- Entity system allow users to define complex content types choosing from lot of field types both from core and contributed modules.

Cons:

- Since Drupal is built on top of the Symfony framework, or at least part of it, and is developed as a highly structured CMS with several abstraction layers, the learning curve might be steep, especially for unskilled developers.
- For the same reasons of the first point, raw performances might be limited compared to procedural-oriented CMS like WordPress when not properly configured.
- Out of the box, Drupal has limited capabilities in media management. Apart from lack of video management as Broadcasters might expect, which is supposedly an issue for all CMS, Drupal requires additional modules also to achieve simple manual editing of images.
- Core reengineering in last major release, required contributed modules to be refactored too. To date lot of modules are still in beta status, someone even in alpha, and so not stable enough for production-ready applications.

3.4 Selecting an appropriate CMS

After the observation of the candidates on its own, a fair comparison of the three has to be conducted. Before drawing a final conclusion, each system shall be characterised by its main advantages, including their main target group [CPA16]:

WordPress: Ideal for fairly simple websites, such as everyday blogging and news sites; easy-to-use and an easy-to-manage backend; add-ons make it easy to expand the functionality of the site; best for beginners

Joomla: Allows to build a site with more content and structure flexibility than WordPress, but still with fairly easy and intuitive usage. Supports specially E-commerce and social networking

Drupal: For complex, advanced and versatile sites; for sites that require date organization; for community platform sites and online stores;

The following table shall give an overview of the most significant features of each CMS, which leads, taking into account the propositions of the previous sections, to a conclusion of which CMS is the most appropriate for the needs of MPAT.

Comparison of CMS systems for MPAT		Drupal™	Joomla!™	WordPress
General	Number of websites	0.75 million	2.8 million	74 million
	User downloads	> 15 million	> 30 million	> 140 million
	Website usage share in %			
	Market share in %	4.9	6.1	59.3
	Number of free themes	1800+	900+	2000+
	Number of free plugins	24,000+	7,000+	27,000 +
	Primary target sites	All-round solution	E-commerce, social networking sites	Blogs, corporate websites, small-medium sized websites
Business	License	GPL	GPL	GPL
	Average Setup and Customization Cost in $*	5,000 - 50,000	2,000 - 20,000	250 - 15,000
	Average Monthly Maintenance Cost in $*	1,000 - 2,000	500	250
Preparation	One Click installation	✔	✔	✔
	Platform	PHP	PHP	PHP
	Supported databases	MariaDB, Microsoft SQL Server,	MySQL, PostgreSQL, MSSQL, SQLite	MY SQL, Maria DB
	Supported web servers	Apache	Apache, Nginx, Microsoft IIS	Apache

Table 2: CMS comparison table [WJD16], [CMS16], [CDJ16], [WIK16]

In the crowded market of CMS systems it seems that all the systems are to be measured with WordPress as its most dominant player. But is it really living up to its expectations and more importantly does it qualify for the usage as a platform for the MPAT system?

First, let's identify the similarities between the three candidates. The licenses which are tied to the systems are basically identical allowing the license holder to freely copy, alter and distribute the source code. However the principle of a copy left, must be conceded to the altered version, meaning to pass on the same rights as the original inherits. Further the underlying technology or used frameworks such as PHP, Apache and MySQL are more or less the basis for all the enlisted CMSs. After setting up the necessary web hosting services, the installation of the systems is in each case easy to handle with the help of a one-click-installation in less than 15 minutes. Further the costs for the maintenance are relatively low and normally don't exceed 1000€ a month (for hardware costs, internet access, administration, etc.) due to having not to pay any license fees. When talking about security, one has also to mention the update frequency, which is also rather short, ranging from 36 to 51 days.

Now, let's have a look at the differences among the candidates. The analysis shows that Wordpress is the most widespread solution with the biggest market share and community and therefore being well documented and easy accessible for programmers of 3rd party components. The re-usage of already existing modules for the means of MPAT is also a major benefit and therefore covers the modular approach of the project, which is also true for Drupal and Joomla. But it has the most free plugins and themes of all CMS platforms, also guaranteeing that there's a large developer base which can be addressed by the MPAT project to develop a new ecosystem on the basis of WordPress with little period of adjustment. Further the easy-to-handle backend interface for non-developers resp. editors is also a major advantage of WordPress. Of course, there exist also reasons not to use WordPress for the project, as there's no integrated export function for generated applications for example. But in the end every CMS needs to be expanded to suit the purpose of MPAT.

4 Efforts to Design a System from Scratch

In this section, we describe the efforts and techniques necessary to design and develop a Content Management System (CMS) from scratch. This assumes the availability of technologies that can be used to aid the implementation of system. However, it does not reuse any of the underlying frameworks found in existing CMSs.

4.1 Architecture

Fundamentally, a CMS is responsible for storing content, enabling the management of this content, and facilitating the presentation of this content in some way. This fits very well with the Model-View-Controller (MVC) concept. Using MVC provides a distinct decoupling of these constituent elements to allow for cleaner implementations in which logical separation is enforced.

In MVC, the *model* manages and stores the data used in the application (in this case, the main example would be the content). The *view* is the representation of this information, and is updated to reflect changes made in the *model*. In the case of a CMS, the view will likely be the administration interface, which presents the available content, and facilitates the management of such. This is done by issuing commands to the *controller,* which in turn manipulates the *model.* These three aspects interact with each other in a cyclical, yet well-defined, way to form the MVC concept.

4.2 Realising the Architecture

Using the MVC model in the context of a CMS has some clear benefits for modularity and separation of concerns. There are a number of existing frameworks that aim to assist developers in using MVC in their applications. These are available in many different languages, including many designed for use on the web. An example of this is CakePHP, which would be an ideal candidate for realising the backend CMS discussed in the section.

CakePHP is a compelling choice because of a number of factors. This includes the availability of detailed documentation, a large community, and a regular release cycle. Ultimately, it provides the boilerplate code for which to build the CMS. It also provides a number of other functions by default, which are outlined in the following sections.

It is worth noting that the design and implementation of such a system, regardless of the availability of tools designed to aid the process, will be a time and capital intensive process. Developing CMSs, especially if they are to be used in production environments, is an endeavour which requires entire teams of people engaged in development for many months. This goes beyond pure development too; there is a need to extensively test and ship software, as well as document and support the tools. This is an equally expensive process that must not be overlooked.

4.3 Extensibility, Adaptability and Flexibility

By developing in an MVC style, a separation of concerns is enforced. This helps in the building of alternative modules, which can be swapped out to enable certain features to be exposed. However, it requires the intentional design of these features, especially if the modularity is required *within* one of the MVC elements. For example, if a system is desired whereby functional components can be exchanged, added and removed within the *model* element, then there needs to be a framework around this. This includes the definition of well-documented interfaces which ensure a level of interoperability. The CMS would need to be designed from the ground-up with this in mind; it would be difficult to retrospectively add this functionality after the matter.

4.4 User Management

User management is a requirement of many CMSs. Following our previous recommendation for the use of CakePHP, where user management is already integrated, and users can have page specific roles. Every time a page is loaded, these permissions are checked before any operation can be completed. Groups of pages or individual pages can then admit or exclude certain users or groups of users.

Although user management is a core part of CakePHP, it does not in itself provide an interface to this. It would therefore be the responsibility of the designers to create the interfaces necessary to build the interfaces to administer the users. So it would be down to the consortium to design the UI for this tool. Alternatively, existing user management systems such as OAuth can be used.

4.5 Usability and Accessibility

If a CMS was to be created from scratch, then it would be possible to include all of the modern accessibility and usability standards. In most cases, these are now expected (rather than desired) for interfaces. These standards and good practices are well documented, yet still require implementation. This may be a straightforward process in many cases, but to ensure full coverage would be a time consuming and laborious process.

4.6 Performance

A scratch-built CMS could be designed from the ground up to be performant. However, there are always trade-offs in ensuring this, including an increase in resource consumption and increase in development time. However, with modern technologies and tools, a sufficient level of performance can be achieved with relative ease. The additional performance gains possible by designing a high-performance CMS would likely be less intangible and unnecessary for most users.

4.7 Security and Privacy

If a system is built from scratch, many 'known' security issues with popular CMS's would not be present. However, a bespoke system would not have undergone the rigorous testing that popular CMS's would have, making it much more vulnerable to targeted and zero-day attacks. These may occur if an attacker methodically goes through a specific site looking for flaws in security.

4.8 Reliability and Recoverability

There are several standard solutions to reliability, recoverability, and resilience. These could easily be implemented in conjunction with a custom-built solution, but are equally appropriate to all other existing CMSs.

For the databases we could use, for example, 2 hosts each with MySQL installed, running a master-master replication. The frontend, again, could be duplicated on different hosts (git worktree's from a remote repository), with HAProxy sat in front of them with a floating IP. This would provide resilience and reliability with recoverability only a git push away (maybe with a Docker build for the frontend servers)

4.9 Summary

The advantages to building a bespoke system are:

- It can be built to a specific set of requirements, exactly matching what is needed
- It is simpler to optimise, as there is not additional/unnecessary functionality
- Security through obscurity – known security flaws (present in popular CMSs) are not freely introduced

The disadvantages are:

- Building a platform is ultimately **much** more expensive, both in terms of financial outlay and time necessary
- Test coverage is likely to be smaller, leading to more bugs, unseen security flaws, etc,
- There is no existing community for supporting users
- It is not guaranteed that performance/security goals are met, or even match that of existing platforms – the chance of failure is **much** greater

5 Overall Conclusion

When weighing up the usage of an already existing CMS versus a solution developed from scratch, it has to be said that building a new system seems to be tempting, because it could fulfil all the specifications of MPAT. But in the end the effort to build such a system would exceed the scope of the project as stated in the previous section. Additionally it is not only the technical challenge which has to be dealt with, but more importantly the ecosystem around it, which has to be built from scratch. Attracting users, developers and building a community needs a lot of effort and time, while hopping on a train seems more feasible and guarantees that the underlying system is well tested and that performance and security goals can be met.

Therefore with an existing CMS, the focus can be set to the upmost goals of MPAT, which is building a multi-platform application toolkit. After a careful analysis the consortium finally decided to use Wordpress as platform for the MPAT system.

Glossary

CMS	Content-Management-System
DoW	Description of Work
EC	European Commission
HAT	HbbTV Application Toolkit
HbbTV	Hybrid Broadcast Broadband TV
IPR	Intellectual Property Rights
MVC	Model-View-Controller
MPAT	Multi-Platform Application Toolkit
QA	Quality Assurance
R&D	Research and Development
VoD	Video-on-Demand
WP	Work Package

Partner Short Names

Short Name	Name
FOKUS	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V. (DE)
IRT	Institut für Rundfunktechnik GmbH (DE)
RBB	Rundfunk Berlin-Brandenburg (DE)
ULANC	Lancaster University (UK)
MEDIASET	Reti Televisive Italiane S.p.A. (IT)
LEADIN	Leadin Oy (FI)
FINCONS	Fincons SpA (IT)
IMT	Institut Mines-Telecom (FR)

References

[CPA16] Content management system comparison - June 2016
http://makeawebsitehub.com/content-management-system-cms-comparison/

[WPA16] Wordpress architecture – accessed May 2016
http://codex.wordpress.org/images/2/20/WP_27_modules.JPG

[CMS16] Content Management Systems - April 2016
http://de.statista.com/statistik/daten/studie/320670/umfrage/marktanteile-der-content-management-systeme-cms-weltweit/

[W3T16] Content Management System analysis of W3Techs - June 2016
http://w3techs.com/technologies/overview/content_management/all

[WPF16] WordPress Features - accessed June 2016
https://wordpress.org/about/features/

[JOA14] Joomla CMS Architecture – July 2014
http://de.slideshare.net/dustinczysz/joomla-cms-architecture

[JOS16] Joomla Performance – June 2016
http://joomspot.net/how-to-speed-up-optimize-joomlas-performance.html

[COR16] Core modules – accessed March 2016
http://www.comentum.com/drupal-vs-joomla-cms-comparison.html

[DRU16] The Drupal Overview – accessed April 2016
https://www.drupal.org/getting-started/before/overview

[WJD16] Content Management System comparison - May 2016
http://websitesetup.org/cms-comparison-wordpress-vs-joomla-drupal/

[CDJ16] Wordpress vs. Joomla vs. Drupal comparison chart - April 2016
http://www.comentum.com/drupal-vs-joomla-cms-comparison.html

[WIK16] Wikipedia: List of content management systems – last updated June 2016
https://en.wikipedia.org/wiki/List_of_content_management_systems

[DSR16] Drupal 8 system requirements – March 2016
https://www.drupal.org/requirements

[PCR16] Performance comparison between D7 and D8 - March 2016
http://www.jeffgeerling.com/blog/2016/yes-drupal-8-slower-drupal-7-heres-why

[DLF16] Drupal licensing FAQ – accessed March 2016
https://www.drupal.org/about/licensing#q1

[DOR16] Drupal official repository for contributed modules - accessed March 2016
https://www.drupal.org/project/project_module

[TTE16] Twig template engine in D8 - accessed March 2016
https://www.drupal.org/theme-guide/8/twig

[DSB16] Drupal 8 security bulletin - accessed March 2016
https://www.drupal.org/SA-CORE-2016-001

[DSP16] Drupal security bug bounty program – accessed June 2016
https://www.drupal.org/drupal8-security-bounty

List of Figures

List of Tables