Methodology for market analysis

This is an openly published report based on deliverables of the Pilot Project Pilot "We4Authors" on Web accessibility for web authoring tools producers and communities (LC-00788801) lead by Funka in collaboration with CTIC and funded by the European Commission.

In the report, we use the term CMS (Content management System) as synonymous to authoring tool.

Table of contents

Sampling

The sample of the survey is a collection of public bodies and their corresponding (URL) websites from all the EU 28 Member States, including public institutions and administrations at any level – local, regional, national – that are affected by the Directive (EU) 2016/2102.

This systematic sampling will be based on accurate and a reliable source of information that is updated yearly, the eGovernment factsheets[1] published by the European Commission's National Interoperability Framework Observatory (NIFO). This observatory operates with the support of the ISA and ISA² Programmes, producing a PDF document for each Member State with updated indicators about the country. Every factsheet includes specific information about the eGovernment services, infrastructure, strategy, and the public stakeholders involved in the definition of policies, and provision of eGovernment services. These annual eGovernment Factsheets are issued in May.

These factsheets include a list of public actors segmented by thematic (i.e., tourism, industry, economy, legislation, etc.). These institutions, along with their websites, will compose the sample of the study.

The selection of the public bodies and URLs will be performed using automatic methods, combined with an expert final revision and curation that guarantees the quality of the sample. The process to identify and select the survey sample is shown as follows:

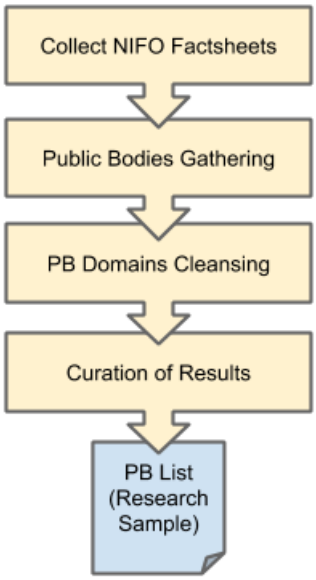

Fig. 1. Process for survey sample identification

[1] https://joinup.ec.europa.eu/collection/national-interoperability-framework-observatory-nifo/egovernment-factsheets-and-infographics

1. **Collect NIFO factsheets**. An expert will download all the EU28 Member State eGovernment factsheets from the NIFO's website. The output of this task is a list of PDFs, one per country.
2. **Public bodies gathering**. Every PDF will be analysed using a PDFx script. This tool will extract automatically the extract denomination of each public body listed on the documents and their URLs. The list of public bodies of each country will be stored in a CSV document. The output of this task is a set of CSV documents with the list of all public bodies detected in the factsheets. This list may include duplicates and references to other institutions that are out of the scope (e.g., European institutions, JoinUp, etc.). The final document will include a column with the name and code of the country, as well as a composed URL to use an external service in the analysis phase.
3. **Public Body domains cleansing**. Every CSV file with the list of public bodies per country will be refined, leaving a list of unique public bodies and their URLs. The output of the previous task (CSV files) is loaded on a Google spreadsheet (one tab per country). A script will clean the URLs obtained, removing all the invalid URLs: the external URLs (i.e., europa.eu, joinup.eu, etc.); duplicates; non-HTTP/HTTPS URLs (i.e., mailto, tel, etc.). This script will extract the domain and subdomain into a new column. This process will be repeated in each tab of the spreadsheet. The output is a spreadsheet with 28 tabs with a list of unique domains (URLs). It is expected to have an average of 80 public bodies per country
4. **Curation of results**. An expert will review every set of URLs, removing wrong URLs. All the domains that seem to be not related to the country or the scope of the project will be removed. The output of this task is the final sample of the survey, a curated spreadsheet with a list of domains (URLs) organised by countries in separate tabs.

As set out in the process, we can identify a list of public bodies and their URLs for every Member State from the original NIFO factsheets. Approximately, it is expected to get an average of 80 websites per country. All the identified organization's websites will be used as input for the analysis task.

The CSV documents with the sample for the research will have the following structure:

Property	Description
Id_country	ISO 3166[2] code for the country
Country	Country name
URL	Public body's URL extracted from the NIFO factsheets
URL_BW	Composed URL to use the external *BuildWith* interface

Table 1: Structure of the CSVs with the URL detected in the sampling phase

This sampling method enables scalability in case that other official sources of information are identified.

[2] https://www.iso.org/iso-3166-country-codes.html

Analysis of websites

Since the objective of this assessment is analysing the most used CMS in public administrations, the next step is collecting information about the technologies used to build the websites identified in the previous phase. This identification process is made through automatic means.

The tools and scripts used to identify the technologies of the websites are based on the detection of **software fingerprints**, or elements that enables the script to identify the concrete software producing an application or website portal. These fingerprints may be identified by analysing: HTTP headers, while listening the HTTP/S communications; cookies sent to the client; specific files linked to the document (e.g., style-sheets, JavaScript, images, icons, etc.); and HTML code. Every CMS include specific fingerprints that may identify it with a high probability.

The automatic CMS identification process is not always accurate because sometimes those characteristic elements are not found. Some organisations hide them (i.e., removal or obfuscation) with the objective of protect the system against potential attacks, minimising the vulnerability of the website. In order to solve this, this methodology will rely on several tools that analyse different aspects to gather as much information as possible.

The automatic analysis will be driven by a **Node.js application**, developed ad-hoc for this project. This application is fed with the CSVs documents produced in the sampling phase, containing the public bodies' URLs.

The phase of analysis includes the following tasks, performed using different tools:

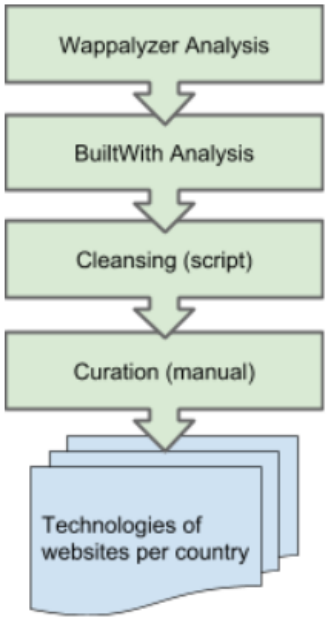

Fig. 2. Analysis phase process

Wappalyzer Analysis

Wappalyzer[3] is a cross-platform utility that uncovers the technologies used on websites. It detects content management systems, ecommerce platforms, web frameworks, server software, analytics tools, and others. Wappalyzer offers an API to perform the identification task.

This tool uses JSON as input/output document format. The input will be the collection of URLs extracted and curated during the sampling phase.

The script will be configured to analyse every single public body URL, following this process for each single URL:

1. The Node.js application runs the analysis taking the URL as input.
2. If the server returns a 30x HTTP code (e.g., 301, 302, 303 redirections) the script will try to get the content of the redirection. It will follow all the redirections until the server returns a 200 HTTP code (OK)
3. The Node.js script invokes the Wappalyzer API to crawl the website, **analysing up to ten documents** in the same domain, linked from the current document (URL). This step is important because some initial URLs are static landing pages and the CMS is in other location (a URL with the same domain but different path).
4. Wappalyzer will analyse all the URLs and it will return a comprehensive list of technologies (type, vendor, version, confidence, etc.) or “” in case nothing is detected.
5. All results are stored locally and dumped on a CSV file.


```
{
		"urls": ["http://administracionelectronica.gob.es/pae_Home",
"http://administracionelectronica.gob.es/pae_Home/?idioma=es", "http://administracionelectronica.gob.es/pae_Home/?idioma=ca",
"http://administracionelectronica.gob.es/pae_Home/?idioma=eu", "http://administracionelectronica.gob.es/pae_Home/?idioma=gl",
"http://administracionelectronica.gob.es/pae_Home/?idioma=ca_valencia", "http://administracionelectronica.gob.es/pae_Home/?idioma=en"],
		"applications": [{
				"name": "AddThis",
				"confidence": "100",
				"cms": "",
				"version": "",
				"icon": "AddThis.svg",
				"website": "http://www.addthis.com",
				"categories": [{
						"5": "Widgets"
				}]
		}, {
				"name": "IIS",
				"confidence": "100",
				"cms": "",
				"version": "5.0",
				"icon": "IIS.png",
				"website": "http://www.iis.net",
				"categories": [{
						"22": "Web Servers"
				}]
		}, {
				"name": "jQuery",
				"confidence": "100",
				"cms": "",
				"version": "1.11.1",
```


[3] https://www.wappalyzer.com

```
                                "icon": "jQuery.svg",
                                "website": "https://jquery.com",
                                "categories": [{
                                                "12": "JavaScript Frameworks"
                                }]
                },{
                                "name": "jQuery UI",
                                "confidence": "100",
                                "cms": "",
                                "version": "",
                                "icon": "jQuery UI.svg",
                                "website": "http://jqueryui.com",
                                "categories": [{
                                                "12": "JavaScript Frameworks"
                                }]
                },{
                                "name": "Windows Server",
                                "confidence": "0",
                                "cms": "",
                                "version": "",
                                "icon": "Microsoft.svg",
                                "website": "http://microsoft.com/windowsserver",
                                "categories": [{
                                                "28": "Operating Systems"
                                }]
                },{
                                "name": "Google Analytics",
                                "confidence": "100",
                                "cms": "",
                                "version": "",
                                "icon": "Google Analytics.svg",
                                "website": "http://google.com/analytics",
                                "categories": [{
                                                "10": "Analytics"
                                }]
                },{
                                "name": "Piwik",
                                "confidence": "100",
                                "cms": "",
                                "version": "",
                                "icon": "Piwik.png",
                                "website": "http://piwik.org",
                                "categories": [{
                                                "10": "Analytics"
                                }]
                },{
                                "name": "Twitter",
                                "confidence": "100",
                                "cms": "",
                                "version": "",
                                "icon": "Twitter.svg",
                                "website": "http://twitter.com",
                                "categories": [{
                                                "5": "Widgets"
                                }]
                }],
                "meta": {
                                "language": "en"
                }
}
```


Sample of code generated by Wappalyzer after the analysis of a website URL

BuiltWith Analysis

Additionally, the script will use the same URLs as Wappalyzer in order to perform another exhaustive assessment with BuiltWith.

BuiltWith[4] is a tool that enables analysing the web technologies of specific websites. This tool offers an API to launch queries, producing complete reports in HTML and JSON. As in the previous step with Wappalyzer, there is a specific section for recognised CMSs. The value returned by this tool, is stored in the same CSV with all the URLs for every country.

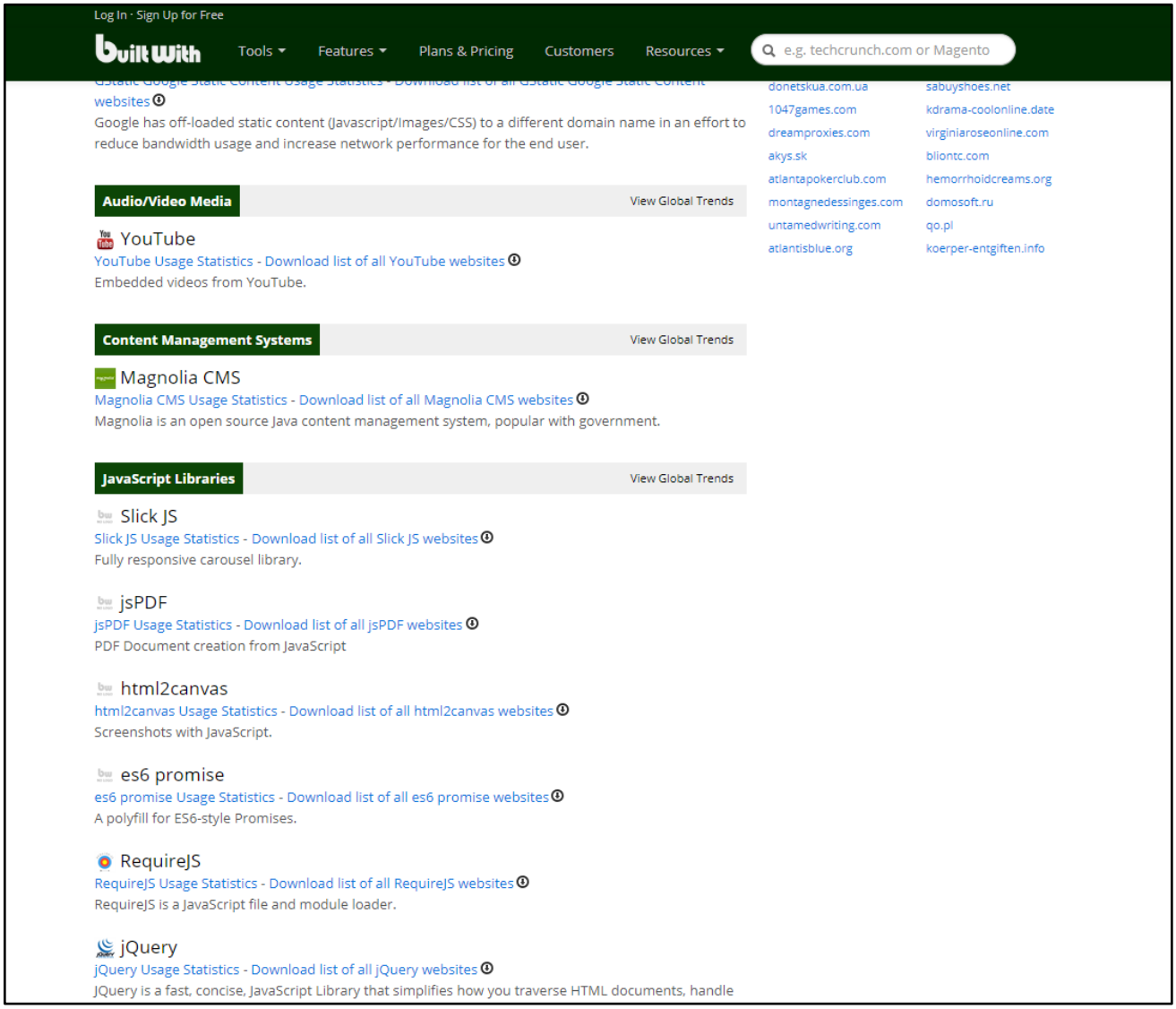

Fig. 3. Screenshot with BuiltWith results

This tool is invoked by the Node.js as next step of the analysis process.

[4] https://builtwith.com

Cleansing and curation

This stage will use the results generated by the Node.js application in the previous step. The resulting data will be stored on a CSV document with the following structure:

Column	Description
Id_country	ISO 3166 code for the country
Country	Country Name
URI	URL after redirection (in case server returns a 30x HTTP code)
DOMAIN	Domain extracted to avoid duplicates in public body list
APIWAP	Wappalyzer result. This is a JSON document containing all the technologies detected, also the name and version of the CMS and the level of confidence, if detected
CMS1	CMS name recognised by the Wappalyzer tool
CMS2	CMS name recognised by the BuidWith tool
CMS_CANDIDATE	The most probable CMS according to CMS1 and CMS2. The algorithm is as follows: If CMS1 is detected, CMS_CANDIDATE = CMS1 If CMS1 is empty and CMS2 exists, CMS_CANDIDATE = CMS2 Otherwise, CMS_CANDIDATE = “” (empty)
CMS_NORMALISED	Names of the detected CMS normalised. This normalization is needed for a subsequent aggregation by name. It implies matching all the variations of nomenclature, mapping them in a common capitalized term (i.e., “Joomla” and “Joomla!” Are normalized as “JOOMLA”).

Table 2: Structure of the CSV with the results of the analysis

Due to the redirections performed when the servers return 30x HTTP codes, the list of results will be loaded into a Google Spreadsheet. There, a cleansing script will be executed, removing duplicate entries, as well as those that experienced connection errors.

The final output will be stored in the same spreadsheet.

Reporting and final selection

After the analysis phase, data will be aggregated and reported using tabular and visual representations. The reporting phase aims at creating a comparative analysis of the identified CMS through visual representations to help experts to select a final subset of the most representatives CMSs for the study.

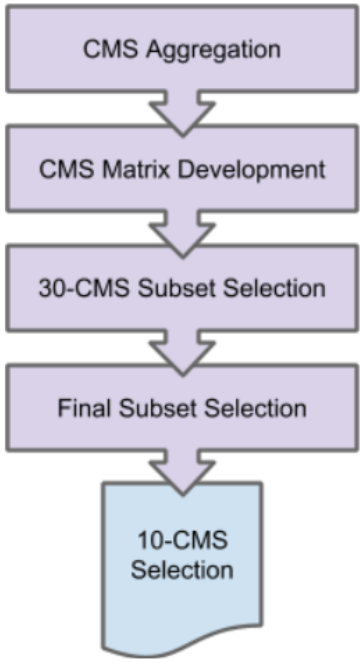

Fig. 4. Process of CMS selection

This final process includes the following tasks:

1. **CMS Aggregation**. The resulting CMS lists in the spreadsheet will be aggregated by CMS vendor name and country. The output of this task will be list of the most popular CMSs in every country, reported in descending order by number of occurrences.
2. **CMS Matrix Development**. A matrix of CMSs and visualization of similar aspects, clustering features using different metrics and indicators, being focused on accessibility aspects. This matrix, developed on a spreadsheet is filled in by the expert team. See more details about the matrix columns in the table below. Since there may be several technologies included in the matrix that are either not real CMS or not interesting for the study, the matrix will be trimmed and only the relevant CMSs will be included as the output of this task.
3. **30-CMS Subset Selection**. From the matrix, a group of experts will select a subset of 30 CMS. The expert criteria for that selection will be based on the key indicators of the matrix. Selecting the most representatives, aiming to cover as many technologies and platform as possible.
4. **Final CMS Subset**. After analysing the outcome of the previous step, a group of experts will select a subset of ten (10) CMS. The final decisions regarding the selection of the CMS list will be done taking into consideration the best technical opportunities and relevance in the context of this project.

The main technological features to be represented in the matrix are:

Feature	Description
Is CMS	Flag that indicates if the software is a real CMS or not. This is used when the automatic tools gather information about the software that is not a real CMS (e.g., Google Search Engine is considered as a CMS to Wappalyzer).
Application Server	The system needed to run the CMS server.
License	License of the CMS. The most valuable option is that the CMS is under a non-restrictive license.
Operating System	Platform required for the clients to access the CMS. Multi-platform support would be the most valuable option.
Programming Language	The programming language in which the CMS is coded. The most valuable option is any of the open and well community-supported programming languages.
Friendly URLs	Support of customizable URLs. This is useful to increase usability of the website.
WYSIWYG Editor	Support of a graphic editor that make the edition process easy.
Multilingual support	Support of multilingual features.
Built in applications	Relevant libraries and applications that the CMS includes.
Last update	Date of latest software update.

Table 3: Features of the CMS Matrix

I. References

- *NIFO's Factsheets: https://joinup.ec.europa.eu/collection/national-interoperability-framework-observatory-nifo/egovernment-factsheets-and-infographics*
- *ISO 3166 Country Codes: https://www.iso.org/iso-3166-country-codes.html*