Developing a RESTful API for a Web Accessibility Evaluation Tool

Cristian Timbi-Sisalima[1], José R. Hilera[2], Salvador Otón[2] and Paola Ingavelez[1]
[1]*Grupo de Investigación en Inteligencia Artificial y Tecnologías de Asistencia, Universidad Politécnica Salesiana, Cuenca, Ecuador*
[2]*Departamento de Ciencias de la Computación, Universidad de Alcalá, Alcalá de Henares, Spain*

Keywords: HCI, RESTful API, Testing, RESTful, WCAG, Web Accessibility, Enterprise Web Information System, Web Service.

Abstract: Nowadays, the evaluation of the accessibility of Enterprise Web Information Systems is based on the Web Content Accessibility Guidelines (WCAG 2.0), created by the World Wide Web Consortium (W3C) in 2008 and adopted in 2012 by the International Organization of Standardization (ISO) as ISO/IEC 40500 standard. Web accessibility evaluation tools are software programs or online services that help developers determine if web content meets the accessibility guidelines. This paper shows the extension of an existing online tool, with the aim of exposing its functionality also as an Application Program Interface (API). The original tool is used through a Web user interface, and this extension exposes functionality through web services based on REST technology, so that developers can invoke them from their own applications for accessibility testing.

1 INTRODUCTION

Web accessibility is the feature of a web application or web page that determines how it can be used by people with the widest possible range of capabilities.

The more accessible a website is, the greater the variety of users who can use it, including those who present some kind of disability. This would apply, for example, to users with visual disabilities, such as blindness, low vision or problems with color perception. An accessible Web application would be one that met requirements to ensure an adequate level of contrast, or that there are always text alternatives to visual information that can be reproduced by assistive technologies (AT), such as screen readers, used by blind or partially sighted users.

To create accessible websites must apply the paradigm of design for all, a concept that refers to design websites in a universal way, so nobody is excluded from use (Roig-Vila et al., 2014).

In 2008, the World Wide Web Consortium (W3C) published the Web Content Accessibility Guidelines (WCAG 2.0), adopted in 2012 by the International Organization for Standardization (ISO), as standard ISO/IEC 40500 (ISO, 2012). This standard establishes the requirements to be met by a Web application to be accessible (W3C, 2008).

WCAG is important, not only for being converted into standard ISO, but also because many countries have created laws based on this standard, that establish that Enterprise Web Information Systems of strategic companies (such as Banks, energy firms, etc.), and public services offered through the Web must be accessible (W3C, 2006).

WCAG 2.0 defines 61 accessibility success criteria (requirements) to be satisfied by web applications. The success criteria are grouped into 12 guidelines and 4 principles (Table 1). In addition, three levels of conformance are established (A, AA and AAA) for websites, depending on the success criteria met. To get level A, 25 criteria have to be met. To get level AA, besides the aforementioned, 13 more criteria have to be met. Level AAA is got when all 61 criteria are met.

In order to provide guidance for web developers and evaluators on meeting WCAG 2.0, the W3C has published a series of documents to support this standard. One of the most important includes guidance in form of a list of useful techniques to satisfy success criteria, and a repertory of common failures and how to avoid them. For each technique and failure, a repertory of code examples, resources, and tests are offered (W3C, 2015).

Table 1: Organization of WCAG 2.0.

Principles	Guidelines and Success Criteria (n)
1. Perceivable	1.1 Provide text alternatives (1). 1.2 Provide alternatives for time-based media (9). 1.3 Create adaptable content (3). 1.4 Make content distinguishable (9).
2. Operable	2.1 Make all functionality available from a keyboard (3). 2.2 Provide users enough time to read and use content (5). 2.3 Do not design content in a way that cause seizures (2). 2.4 Provide ways to help users navigate (10).
3. Understandable	3.1 Make text content readable and understandable (6). 3.2 Make Web pages appear and operate in predictable ways (5). 3.3 Help users avoid and correct mistakes (6).
4. Robust	4.1 Maximize compatibility with user agents (2).

There are available many web accessibility evaluation tools in form of software programs or online services that help determine if a Web site is accessible (W3C, 2014). These tools can significantly reduce the time and effort required to carry out evaluations, and can assist developers in preventing accessibility barriers, repairing encountered barriers, and improving the overall quality of Web sites. But not all success criteria can be automatically evaluated, many of them need to be evaluated manually. So, Web accessibility evaluation tools can be useful to determine the conformance of Web sites to accessibility checks which can be executed automatically, and assist reviewers in performing accessibility checks which need to be evaluated manually.

In the following section of this paper, an online evaluation tool developed at the Salesian Polytechnic University of Ecuador (UPS) is presented. In section 3, an extension of this tool to expose their functionality through an Application Program Interface (API) based on Web services is described. Finally, in section 4 some conclusions about the work carried out are discussed.

2 WEB ACCESSIBILITY EVALUATION TOOL

As mentioned in previous section there are accessibility evaluation tools that support WCAG 2.0 standard. These tools usually check compliance with the success criteria using the techniques suggested by the W3C. But not all techniques can be checked automatically. Three types of techniques can be considered:

- Programmable, or techniques which can be determined automatically.
- Manual, whose verification is subjective and requires a judgment of an expert.
- Mixed, or techniques which has a percentage of manual checking and another that can be supported automatically by software.

An evaluation tool in a basic way receives a request for content to be analysed, usually by means of a URL or HTML code, and then proceed to verify compliance with WCAG techniques or similar rules, and shows the result of verification of these techniques. The W3C maintain a web page with information and links to use many evaluation tools (W3C, 2014).

Although not yet been included in the list maintained by the W3C, the UPS has developed an evaluation tool called OWA that is now available in http://observatorioweb.ups.edu.ec. It is a supplement to a project entitled Web Accessibility Observatory Ecuador, working in coordination with the National Council for Equal Ecuador Disabilities (CONADIS).

It is a tool that combines server technologies Java JEE and JavaScript. Figure 1 shows the two main modules of the software installed in the web server.

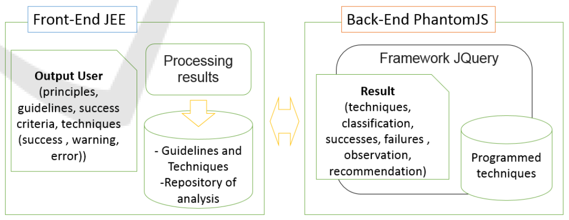
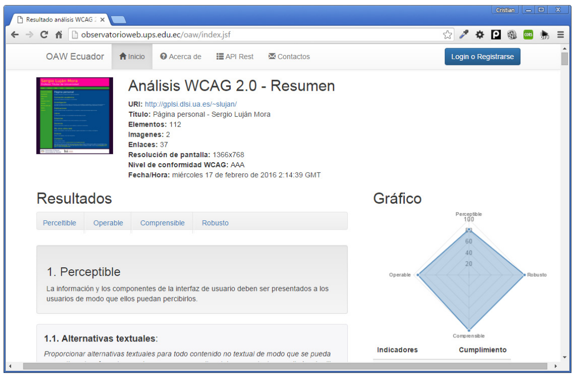

Front-End JEE
Output User (principles, guidelines, success criteria, techniques (success , warning, error))
Processing results
- Guidelines and Techniques -Repository of analysis
Back-End PhantomJS
Framework JQuery
Result (techniques, classification, successes, failures , observation, recommendation)
Programmed techniques

Figure 1: Main server components of the evaluation tool.

The Front-End JEE module corresponds to the Web application of the user interaction, in which requests for analysis will be conducted and the visual results of the evaluation are presented. This component accesses to the entity relationship model of WCAG 2.0 principles, guidelines, success criteria and techniques and the database with the results of the evaluations performed. The user interface is accessible and responsive. Figure 2 shows the page with the results of an evaluation that are offered in a textual way, but also as a diagram with the percentage of compliance of each WCAG 2.0

principle. This is a novelty compared to other existing tools on the market.



Análisis WCAG 2.0 - Resumen

URI: http://gplsi.dlsi.ua.es/~slujan/
Título: Página personal - Sergio Luján Mora
Elementos: 112
Imagenes: 2
Enlaces: 37
Resolución de pantalla: 1366x768
Nivel de conformidad WCAG: AAA
Fecha/Hora: miércoles 17 de febrero de 2016 2:14:39 GMT

1. Perceptible

La información y los componentes de la interfaz de usuario deben ser presentados a los usuarios de modo que ellos puedan percibirlos.

1.1. Alternativas textuales:

Proporcionar alternativas textuales para todo contenido no textual de modo que se pueda

Figure 2: Screenshot of evaluation results interface.

The Back-End PhantomJS module is responsible for runtime analysing the success or failure of each of the implemented techniques (57 techniques implemented to date). For that it uses PhantomJS, a headless WebKit scriptable with a JavaScript API (Friesel, 2014). It has fast and native support for DOM handling, JSON generation, and includes the use of jQuery. The advantage of using this framework is that it renders the web page generating a DOM representation as the one handle by a web browser. Accessibility evaluation is performed on DOM and not on the HTML source code of the page, which is necessary when a page includes JavaScript code that is executed dynamically and affects the structure of the page. There are other tools on the market that carry out the evaluation on the HTML and do not perform the evaluation of dynamic web pages correctly.

Figure 3 shows code to check if the <input> elements of the web page under evaluation meet the technique *"H44: Using label elements to associate text labels with form controls"*, associated with the fulfillment of WCAG 2.0 success criteria 1.1.1, 1.3.1, 3.3.2 and 4.1.2 of WCAG 2.0, the disclosure of which is shown in Table 2.

```
//Selector of input elements which need to be tagged.
var selector = 'input:not([type=submit]):not([type=hidden]):not([type=image])'+
       ':not([type=button]):not([type=reset]), select, textarea';

//For each input element (field), it is checked if there is a label
//that through the attribute 'for' is able to refer the input element (field).
var inputWithLabel=$(selector).filter(function(){
   //The value of 'for' attribute must be the same as the value
   //of 'id' attribute of the form control.
   if  ($("label[for='"+$(this).attr('id')+"']").size()>0){
     return true;
   }else{
     console.log('Input without label: ' + this.outerHTML);
     return false; }
}).size();

//Number of elements with properly associated labels.
var successFactor  = inputWithLabel;
//Number of elements without properly associated labels.
var errorFactor  = $(selector).size() - inputWithLabel;
```

Figure 3: Server JavaScript code excerpt.

Table 2: Success Criteria associated with H44 technique.

Code	Description
1.1.1	Non-text Content: All non-text content that is presented to the user has a text alternative that serves the equivalent purpose. (Level A)
1.3.1	Info and Relationships: Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. (Level A)
3.3.2	Labels or Instructions: Labels or instructions are provided when content requires user input. (Level A)
4.1.2	Name, Role, Value: For all user interface components (including but not limited to: form elements, links and components generated by scripts), the name and role can be programmatically determined. (Level A)

As it can be seen in the code in figure 3, to assess the technical, all <input> elements of the page are processed, and it checks if they are associated with a <label>. All using the rendered version of the page as "this" object.

With a similar analysis per technique according to the specification and supported in other programming instructions along with PhantomJS, the following data is returned in JSON format for easy processing by the front-end module.

Code, identifier of the technique, e.g. H44.

Classification, to identify the type of result obtained after analysis, may be:

- *SUCCESS*, if the technique has been successfully verified by 100% and not required for subsequent manual checking. Example: All images have the ALT attribute defined.
- *WARNING*, or requires manual verification success, if the technique has been successfully verified but require a manual check. Example: Detected images with empty ALT attribute, verify that these images correspond to decorative images and do not contain information relevant to the user.
- *ERROR*, if the technique at the time of verification has detected one or more faults. Example: We have detected five picture that has not defined an ALT attribute.
- *DOES NOT APPLY*, if the technique cannot be verified on the content being assessed, it does not contain HTML objects of analysis. Example: No images have been detected in the analysed website.

Success Factor, corresponding to the number of elements that satisfy the check. Example: Five images with ALT attribute defined.

Error Factor, corresponding to the number of elements that do not satisfy the verification. Example: Three images does not contain the ALT attribute defined.

Observation, analysis resulting informational message about the evaluated content. Example: The site has not been structured using headings (titles), has not detected any title. In some cases the HTML objects having the error, such as images that do not present an ALT attribute will be included.

Recommendation Message, to reference or guide to compliance with the technique. Example: To facilitate navigation and understanding of overall document structure should be structured using headings properly nested (e.g., h1 followed by h2, h2 followed by h2 or h3, h3 followed by h3 or h4, etc.).

In order to develop the system's structure, we have used the Java Enterprise Edition architecture. Furthermore, the system was deployed through a physical topology of the four layers described in Figure 4.

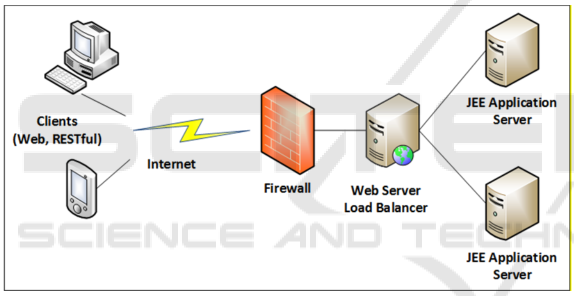


Figure 4: The physical topology of the business information system.

Firewall: is in charge of controlling the different connections coming from Internet to demilitarized zone that contains the public access server. The current implementation of the firewall is based on the network layer and has the support of a system to detect and prevent intruders. This detection system analyzes several patterns on the network layer or even in the application layer.

Web Server and Load Balancer: this component has been configured as a web server that uses data compression, and additionally, it is responsible to translate the client requests to the application (JEE) servers. The work distribution is done according to the availability of each JEE application's server, analyzing for that, the activities that must be carried out at a specific moment.

JEE Application Servers: this layer contains the required stack of technologies (in the web profile) to generate the output of the system's front end. Each of these servers contains a techniques analyzer that is based on PhantomJS and JQuery. In the same way, if the number of requests to use the application increases, it is possible to increase the number of applications server instances. On this basis, the system is able to manage hardware errors and guarantee the service continuity.

3 WEB SERVICES API

The scaling of applications, and in this case of accessibility evaluation tools, can be done through integration with other online evaluation tools. One of the most appropriate technology for this are web services. A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. There are mainly two ways of implementing Web services: SOAP and REST. In this case, we have used REST technology to develop a Web service API for the evaluation tool (Pautasso et al., 2008).

A Web service API that adhere to the REST architectural constraints is called RESTful API (Mehta, 2014), and is defined including these aspects:

- Base URL, such as http://observatorioweb.ups.edu.ec/oaw/srv/wcag
- An Internet media type for the data. This is often JSON but can be any other valid Internet media type (e.g., XML)
- Standard HTTP methods (e.g., GET, PUT, POST, or DELETE)

There is a low number of online web accessibility evaluation tools that expose their functionality through web services using RESTful Web API technology.

Some of them: AChecker (http://achecker.ca/documentation/web_service_api.php), Tenon (http://www.tenon.io/documentation/), WAVE (http://wave.webaim.org/api/).

The problem is that every tool is adding a web API makes creating their own services, and input and output parameters, being different in all cases. Another problem is the format of the result. AChecker, OWA y Tenon allow optionally specify the format as an input parameter (HTML, XML, JSON), while Tenon always returns the information in JSON format.

To potentiate the presented tool, and can be used from other potential applications, its functionality is exposed through web services that return the result of the evaluation in JSON and XML formats. It has created a RESTful API consists of four web services shown in the Table 3. To this it was added on the logical architecture of the application a component corresponding to the interface for external service provision, which was implemented with the support of the API JAX-RS (Java API for RESTful Web Services), making up the layer for providing external services and complementing the logical distribution of the system and the prepared stack of Java technologies (see Figure 5).

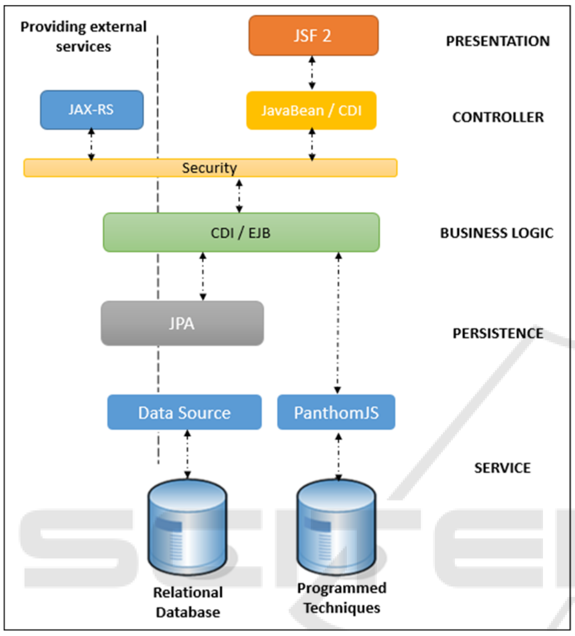


Figure 5: The logical topology of the business information system (n-layers).

Table 3: Implemented Web services in a RESTful API.

Web service	Description
json/techniques	Receives the URL of a website, and returns information, in JSON format, about if it meets or not the WCAG techniques implemented by the tool.
xml/techniques	Receives the URL of a website, and returns information, in XML format, about if it meets or not the WCAG techniques implemented by the tool.
json/compliance	Receives the URL of a website, and returns information, in JSON format, about if it meets or not each of the WCAG success criteria.
xml/compliance	Receives the URL of a website, and returns information, in XML format, about if it meets or not each of the WCAG success criteria.

Additionally, a data structure which supports a clear and simple presentation of the results of the evaluation was created (see Figure 6).

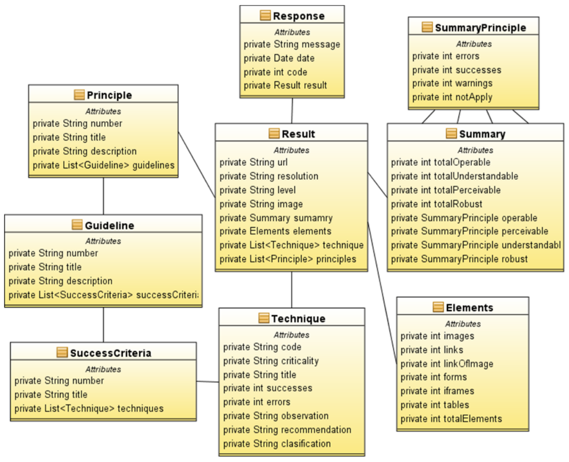


Figure 6: Data model for WCAG 2.0 accessibility evaluation results.

Each service can be called via a HTTP GET request, attaching the required input parameters. The following sections present the details for this. In order to simplify URL expression, in that sections we use a variable named URL_BASE, whose real value is http://observatorioweb.ups.edu.ec/oaw/srv/wcag/.

3.1 Web Services: "Techniques"

These services allows to an external application, consult only the techniques evaluated, and its verification (success or failure). Two versions have been created, depending on the format for the results: JSON or XML.

In the case of this service, the HTTP GET request URL are:

```
URL_BASE/json/techniques?<parameters>
URL_BASE/xml/techniques?<parameters>
```

The input parameters in both cases are the ones described in Table 4.

Table 4: Input parameters for "Techniques" Web services.

Parameter	Value space	Description
url	Valid URL	URL of the website to be evaluated.
key	User account key	Key obtain during user registration. As a basic security method. (Enriquez et al, 2014)
[resolution]	"WxH" (Default: 1366x668)	Display resolution with which the user want to evaluate the site, indicating width (W) and height (H) in pixels.

The result obtained when these services are executed, is a JSON or XML data structure with the main elements represented in Table 5.

Table 5: Output data of "Techniques" Web services.

Element	Description
Date	Date and time at which the request was made.
Message	Error message if unable to perform the evaluation. Possible problems: incorrect URL, invalid parameter format, incorrect API_KEY, etc.

Result	Dataset with information about the evaluation performed.
	`Result: {`
	`elements: {...},`
	`image: "...",`
	`techniques: [],`
	`resolution: "...",`
	`url: "..."`
	`}`

Every field of the result dataset has the meaning indicated in Table 6.

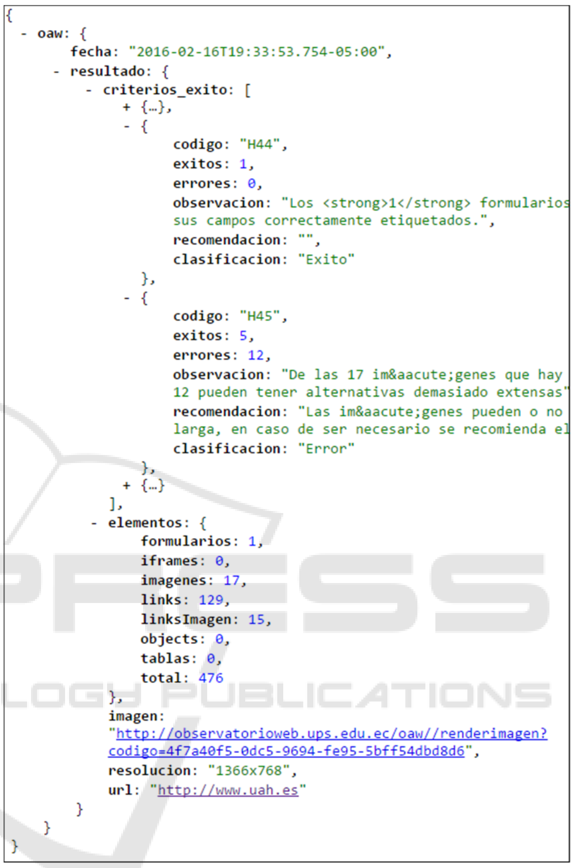
Table 6: Fields of "Result" dataset generated by "Techniques" Web services.

Field	Description
elements	Number of elements of each type included in the page: forms, iframes, images, links, links with image, tables, and total of elements in the page.
image	Link corresponding to a screenshot of the page image as specified resolution.
techniques	Verification results of each technique on the website. Each technique includes: Code, Title, Criticality (high, medium or low, according to user groups, which is reflected in a greater number of people who cannot access the content if not satisfies the technique), Classification (success, warning, error and does not apply as such resulted verification technique on page), Successes (number of hits or HTML elements that satisfy the rule check), Errors (number of failures or elements HTML which do not satisfy the rule check), Observation (informative message related to content analysis), Recommendation (message reference or guide the implementation of the technique).
resolution	Resolution that was assessed the requested web page.
summary	Statistical summary of the results of the evaluation.
url	URL of the website evaluated.

An example of use of "json/techniques" service to evaluate the techniques for a real web page could be as follows:

```
URL_BASE/json/techniques?
url=http://www.uah.es&key=a899ef4d-
714f-440b-a8c2-bc1cj6a42926
```

The result obtained is shown in Figure 7.



```
{
 - oaw: {
       fecha: "2016-02-16T19:33:53.754-05:00",
     - resultado: {
         - criterios_exito: [
             + {…},
             - {
                   codigo: "H44",
                   exitos: 1,
                   errores: 0,
                   observacion: "Los <strong>1</strong> formularios
                   sus campos correctamente etiquetados.",
                   recomendacion: "",
                   clasificacion: "Exito"
               },
             - {
                   codigo: "H45",
                   exitos: 5,
                   errores: 12,
                   observacion: "De las 17 im&aacute;genes que hay
                   12 pueden tener alternativas demasiado extensas"
                   recomendacion: "Las im&aacute;genes pueden o no
                   larga, en caso de ser necesario se recomienda el
                   clasificacion: "Error"
               },
             + {…}
           ],
         - elementos: {
               formularios: 1,
               iframes: 0,
               imagenes: 17,
               links: 129,
               linksImagen: 15,
               objects: 0,
               tablas: 0,
               total: 476
           },
           imagen:
           "http://observatorioweb.ups.edu.ec/oaw//renderimagen?
           codigo=4f7a40f5-0dc5-9694-fe95-5bff54dbd8d6",
           resolucion: "1366x768",
           url: "http://www.uah.es"
       }
   }
}
```

Figure 7: Output data of a real example.

3.2 Web Services: "Compliance"

These services allow to an external application, consults the detailed evaluation of accessibility of a website, depending on the success or failure of the verification techniques associated to WCAG 2.0 and grouped according to the set of principles, guidelines and Success Criteria defined in WCAG 2.0. Two versions have been created, depending on the format for the results: JSON or XML.

In the case of these services, the HTTP GET request URL are:

```
URL_BASE/json/compliance?<parameters>
URL_BASE/xml/compliance?<parameters>
```

The input parameters in both cases are the ones described in Table 7.

Table 7: Input parameters for "Compliance" Web services.

Parameter	Value space	Description
url	Valid URL	URL of the website to be evaluated.
API_KEY	User account key	Key obtain during user registration. As a basic security method.
[level]	"A" "AA" "AAA"	Compliance level on which the user want to evaluate the site. Default: "AAA".
[resolution]	"WxH"	Display resolution with which the user want to evaluate the site, indicating width (W) and height (H) in pixels. Default: "1366x668"

The result obtained when these services are executed, is a JSON or XML data structure with the main element represented in Table 8.

Table 8: Output data of "Compliance" Web services.

Element	Description
date	Date and time at which the request was made.
message	Error message if unable to perform the evaluation. Possible problems: incorrect URL, incorrect parameter format, invalid API_KEY, etc.

result	Dataset with information about the evaluation performed.
	
	`Result: {`
	`  elements: {...},`
	`  image: "...",`
	`  level: "...",`
	`  principles:[`
	`    guidelines:[`
	`      sc:[`
	`        techniques[ ]`
	`        ]`
	`    ]`
	`  ],`
	`  resolution: "...",`
	`  summary: {...},`
	`  url: "..."`
	`}`

Every field of the result dataset has the meaning indicated in Table 9.

Table 9: Fields of "Result" dataset generated by "Compliance" Web services.

Field	Description
elements	The number of elements of each type included in the page: forms, iframes, images, links, links with image, tables, and total of elements in the page.
image	A link corresponding to a screenshot of the page image as specified resolution.
level	The compliance level used to evaluate the webpage.
principles	A set of guidelines used to perform the evaluation. For each guideline is included the following information: number or code, title/name, description (according to WCAG normative), and rules (a set of rules used in the evaluation of the guideline).
guidelines	A set of rules used in the evaluation. For each rule is included the following information: identification number or code, name/title, description (according to WCAG normative), and criterions (a set of rules to determine the success/no-success).
success criteria	A set of criteria to determine the success/no-success during the evaluation. For each criterion is included the following information: number or code, title/name, description (according to WCAG normative), and techniques (a set of techniques related with the criterion that show the results of the verification process of each technique).
techniques	Same as in web services "Techniques" (table 6)
resolution	Resolution that was assessed the requested web page.
summary	Statistical summary of the results of the evaluation.
url	URL of the website evaluated.

An example of use of a "Complaint" service to evaluate the level AAA of compliance of a web page could be as follows:

```
URL_BASE/json/compliance?
url=http://www.uah.es&key=a899ef4d-
714f-440b-a8c2-bc1cj6a42926
```

4 CONCLUSIONS

Web Services enable integration between applications regardless of language or platform used in each. Thus, providing the possibility of developing new tools or more complete systems, adding more than one potential application or system.

If there were other Web services associated to other accessibility evaluation tools, that allowed to verify compliance of other techniques over four techniques and common failures defined by the W3C regarding compliance WCAG 2.0, it would be easy to add these techniques to the 59 checked by the developed tool, and this tool would have greater compliance rate with respect to the technical verification.

Subjectively this could translate into a greater degree of confidence the outcome of the evaluation; but always check and interpretation of the results by an expert is required.

As future work is to increase the number of techniques currently verified by the developed system. In addition, it is working on the expansion of the RESTful API, with new services that provide extra functionality useful in different aspects of web accessibility evaluation.

ACKNOWLEDGEMENTS

Thanks to “Web Accessibility Observatory Ecuador” project, to ESVI-AL project of the European Union ALFA Programme, and to the “Master on Software Engineering for the Web” of the University of Alcalá.

REFERENCES

Enríquez, R., & C, A. S., 2014. *RESTful Java Web Services Security*. Packt Publishing Ltd.

Friesel, R., 2014. *PhantomJS Cookbook*. Packt Publishing Ltd.

ISO, 2012. *ISO/IEC 40500:2012, information technology -- W3C web content accessibility guidelines (WCAG) 2.0*. International Organization for Standardization.

Mehta, B., 2014. *RESTful Java Patterns and Best Practices*. Packt Publishing Ltd.

Pautasso, C., Zimmermann, O., & Leymann, F. (2008). Restful Web Services vs. «Big»’ Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th International Conference on World Wide Web* (pp. 805–814).

Roig-Vila, R., Ferrández, S., and Ferri-Miralles, I., 2014. Assessment of Web Content Accessibility Levels in Spanish Official Online Education Environments. *International Education Studies*, 7(6), p31.

W3C, 2006. *Policies Relating to Web Accessibility*. World Wide Web Consortium.

W3C, 2008. *Web content accessibility guidelines (WCAG) 2.0*. World Wide Web Consortium.

W3C, 2014. *Web Accessibility Evaluation Tools List*. World Wide Web Consortium.

W3C, 2015. *Techniques for WCAG 2.0*. World Wide Web Consortium.