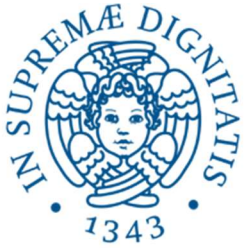


UNIVERSITÀ DI PISA

Corso di Laurea in Informatica Umanistica

Tesi di Laurea Magistrale

Monitoraggio dell'accessibilità di siti Web e presentazione dei risultati

Candidato: *Nicola Iannuzzi*

Relatore: *Prof. Fabio Paternò*

Anno accademico 2020/2021

INTRODUZIONE

L'importanza del Web come strumento di comunicazione e come portale di accesso ai più disparati aspetti del vivere quotidiano, è stata ancora più evidente negli ultimi anni, dove le limitazioni dei contatti "reali" dovuti alla pandemia di COVID-19 hanno dirottato molte delle nostre azioni verso uno schermo digitale. Queste "funzioni virtuali", necessarie alla vita sociale negli stati moderni, devono perciò essere rese fruibili a tutti gli esseri umani.

La crescente attenzione alle tematiche relative all'accessibilità delle risorse Web è riscontrabile sia nelle decisioni legislative europee a riguardo, che nel costante aumento del mercato commerciale legato allo sviluppo di software legati al tema.

L'accessibilità Web è una materia complessa perché l'attenzione deve essere rivolta verso molteplici soggetti, tra i quali: gli utenti che necessitano di soluzioni tecniche e tecnologiche di assistenza, gli enti che si occupano di redigere linee guida utili allo sviluppo in ottica di accessibilità, gli sviluppatori e i tecnici che si occupano di applicare queste linee guida ai loro prodotti digitali.

Una risorsa digitale ha bisogno, dunque, di essere valutata rispetto alla sua capacità di essere accessibile. Le analisi di accessibilità da condurre su tali risorse Web possono essere ricondotte a due tipologie principali: *manuale* e *automatizzata*. Entrambe le tipologie hanno le loro limitazioni in termini di prestazioni e costi, infatti, se l'analisi manuale è decisamente affidabile, è altrettanto dispendiosa in termini economici, così come l'analisi automatica, a fronte di un basso costo monetario, può fornire dei risultati che non rispecchiano la reale situazione. Ma è proprio dall'applicazione di entrambe le tecniche di analisi che è possibile trarre il miglior risultato: una prima analisi automatica può fornire al valutatore di accessibilità umano esperto un percorso da seguire riducendo così il tempo da impiegare.

I motivi che hanno spinto l'Human *Interfaces In Information Systems Laboratory* del ISTI-CNR (Consiglio Nazionale delle Ricerche) di Pisa a creare uno strumento

automatico di validazione dell'accessibilità Web, si crede possano essere riassunti dalla frase che Tim Berners-Lee pronunciò nel 1997[1]:

"Il potere del Web è nella sua universalità. L'accesso di tutti, indipendentemente dalla disabilità, è un aspetto essenziale".

La volontà di mantenere fede allo spirito con il quale si intraprese il percorso di creazione di questo strumento, che oggi è conosciuto con il nome di MAUVE++, ha fornito una solida base per intraprendere questo progetto di tesi. Da quando MAUVE++ ha visto la luce, molte cose nella tecnologia e nell'attenzione verso l'accessibilità Web sono cambiate, di conseguenza anche questo strumento ha dovuto ridefinirsi per continuare a garantire il suo supporto. Si è deciso di definire nuovi requisiti generali a cui uno strumento di valutazione dell'accessibilità Web dovesse mirare e, conseguentemente, sono state progettate e sviluppate funzionalità nuove per rendere MAUVE++ conforme a tali requisiti.

Il primo capitolo è dedicato alla descrizione degli agenti che operano nell'ambito dell'accessibilità Web, all'introduzione delle *Web Content Accessibility Guidelines* sviluppate dal *World Wide Web Consortium* e all'analisi delle diverse tipologie di valutazione dell'accessibilità Web, con particolare attenzione agli strumenti automatici.

Il secondo capitolo descrive il funzionamento e le possibilità che MAUVE++ offriva prima dello sviluppo della nuova versione rilasciata dopo il lavoro di questo progetto di tesi.

Il capitolo successivo ripercorre tutto il processo legato al progetto di tesi: dall'ideazione delle nuove funzionalità alla loro implementazione. Vengono descritti i nuovi requisiti ideati per la nuova generazione dei validatori automatici di accessibilità Web, in seguito vengono esposte e discusse le nuove funzionalità sviluppate per questo progetto di tesi: "Server Side Rendering", "Live preview" e il sistema di monitoraggio periodico dell'accessibilità.

[1] https://www.w3.org/Press/IPO-announce#:~:text=%22The%20power%20of%20the%20Web,of%20the%20World%20Wide%20Web.

Il quarto capitolo descrive un test di usabilità sull'interfaccia di MAUVE++ condotto nel gennaio 2022, dove vengono riportati i dettagli del test proposto ai partecipanti e i risultati ottenuti.

L'ultimo capitolo è dedicato alle conclusioni e ai possibili sviluppi.

1 L’accessibilità Web e gli strumenti automatici di validazione

1.1 Cos’è l’accessibilità di un sito Web?

Il *World Wide Web Consortium* (W3C) attraverso la sua *Web Accessibility Initiative* (WAI) fornisce indicazioni su cosa sia l’accessibilità di una risorsa Web e con quali metodi di sviluppo sia possibile raggiungerla.

Secondo il W3C “l'accessibilità del Web significa che i siti Web, gli strumenti e le tecnologie sono progettati e sviluppati in modo che le persone con disabilità possano usarli”[2]. Nello specifico questa definizione vuole intendere che tutte le persone dovrebbero essere in grado di *percepire*, *capire*, *navigare*, *interagire* con le risorse nel Web e *contribuire* ad esse.

L’accessibilità Web dipende da diversi componenti che lavorano insieme:

- Il **contenuto**, ossia le informazioni contenute in documento Web. Si intendono sia le “informazioni naturali”, quindi testi, immagini e suoni, sia il codice o il markup con il quale queste “informazioni naturali” vengono rese in rete;
- Gli **user agents**, quindi i Web browsers, i media players e gli altri strumenti in grado di interpretare e renderizzare i contenuti;
- Le **tecnologie assistive**, come gli screen reader, i dispositivi di input studiati per particolari disabilità (switches), etc;
- Le conoscenze, le esperienze e, in alcuni casi, le **strategie adattive degli utenti** che usano il Web;
- Gli **sviluppatori**. Designer, codificatori, autori, ecc., compresi gli sviluppatori con disabilità e gli utenti che contribuiscono al contenuto;
- Gli **authoring tools**, ossia quegli strumenti che permettono la creazione di siti Web;

[2] https://www.w3.org/WAI/fundamentals/accessibility-intro/#what

- Gli **strumenti di valutazione dell'accessibilità.**

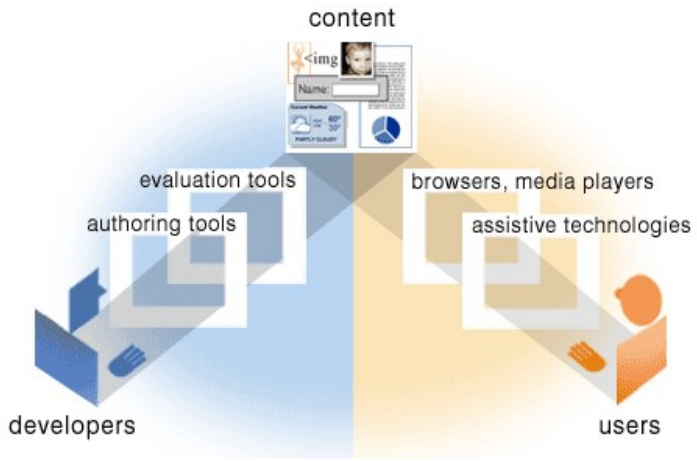


FIGURA 1 RELAZIONE TRA I VARI COMPONENTI: GLI *SVILUPPATORI* REALIZZANO CONTENUTI PER IL WEB ATTRAVERSO GLI *AUTHORING TOOLS* E GLI *EVALUATION TOOLS*. GLI *UTENTI* INTERAGISCONO CON QUESTI CONTENUTI ATTRAVERSO *BROWSER*, *MEDIA PLAYERS*, *TECNOLOGIE ASSISTIVE* OPPURE ALTRI *USER AGENT*. [3]

Per comprendere meglio l'interdipendenza tra i vari componenti nel garantire l'accessibilità di un contenuto Web, è possibile ipotizzare un caso specifico: una persona che non può vedere un'immagine può avere un testo alternativo letto ad alta voce usando un sintetizzatore vocale.

Le *specifiche tecniche* offrono la possibilità di fornire un testo alternativo alle immagini: HTML, ad esempio, permette di specificare il contenuto di un'immagine in forma di testo scritto grazie all'attributo *alt* di un tag *img*.

Le *linee guida per l'accessibilità*, ad esempio le *Web Content Accessibility Guidelines* (WCAG) che verranno illustrate brevemente in seguito, definiscono come implementare in HTML questo testo alternativo.

Gli *sviluppatori* forniscono praticamente un appropriato testo alternativo all'immagine e qualora vengano utilizzati gli *authoring tools* per la creazione dei contenuti, questi ultimi, facilitano e suggeriscono l'inserimento dello stesso.

Gli *strumenti di valutazione* vengono utilizzati per controllare che il testo alternativo sia effettivamente presente nel documento.

[3] https://www.w3.org/WAI/fundamentals/components/

Gli *user agents*, ad esempio i browser, interpretano il codice fornito e renderizzano il testo alternativo a favore degli umani e della macchina.

Le *tecnologie assistive* forniscono un'interfaccia umana al testo alternativo secondo varie modalità (ad esempio la lettura ad alta voce).

Gli *utenti* conoscono il modo con il quale ottenere l'informazione desiderata dalla tecnologia assistiva che stanno utilizzando.

Constatando la stretta relazione tra i vari agenti nel processo di creazione di contenuti Web accessibili, è facile comprendere come la mancanza dei requisiti richiesti in uno soltanto degli attori in gioco, possa inevitabilmente interrompere il flusso di informazioni dalla rete all'utente e viceversa.

1.2 Le linee guida: WCAG

Per il caso ipotizzato in precedenza, si è fatto riferimento alle *linee guida* come strumento grazie al quale si hanno degli indirizzi di sviluppo per l'implementazione di uno standard di accessibilità. Il W3C, attraverso la *Web Accessibility Initiative* (WAI), fornisce uno standard internazionale di riferimento per lo sviluppo di contenuto Web accessibile, con le sue *Web Content Accessibility Guidelines* (WCAG). Questo standard è universalmente riconosciuto, tant'è vero che la *Direttiva (UE) 2016/2102 del Parlamento europeo e del Consiglio, del 26 ottobre 2016, relativa all'accessibilità dei siti Web e delle applicazioni mobili degli enti pubblici*[4] ha imposto alle organizzazioni del settore pubblico l'applicazione ai loro siti Web e strumenti online la norma europea EN 301 549[5], *Accessibility requirements for ICT products and services*, che si basa sull'ultima versione delle WCAG 2.1 con livello di conformità AA[6].

Le WCAG, sviluppate dal W3C in cooperazione con singoli e organizzazioni di tutto il mondo, hanno l'obiettivo dichiarato di fornire uno standard condiviso per l'accessibilità dei contenuti Web che soddisfi le esigenze degli individui, delle

[4] https://eur-lex.europa.eu/legal-content/it/TXT/?uri=CELEX:32016L2102
[5] https://www.etsi.org/deliver/etsi_en/301500_301599/301549/02.01.02_60/en_301549v020102p.pdf
[6] https://www.w3.org/WAI/news/2018-09-13/WCAG-21-EN301549/

organizzazioni e dei governi a livello internazionale. I documenti WCAG spiegano come rendere i contenuti Web più accessibili alle persone con disabilità. Il "contenuto" Web si riferisce generalmente alle informazioni in una pagina Web o un'applicazione Web[7].

L'attuale versione di riferimento delle WCAG è la 2.1, pubblicata nel 2018.

Le WCAG sono strutturate su diversi livelli: le dodici *linee guida* si organizzano sotto quattro *princìpi*. Per ogni linea guida ci sono dei *criteri di successo* testabili organizzati in tre *livelli di conformità*: A, AA e AAA. Per ogni criterio, vengono fornite delle *tecniche*, ossia indicazioni pratiche (descrizione ed esempi di porzione di codice), utili a superare il controllo di conformità della linea guida a cui il criterio fa riferimento.

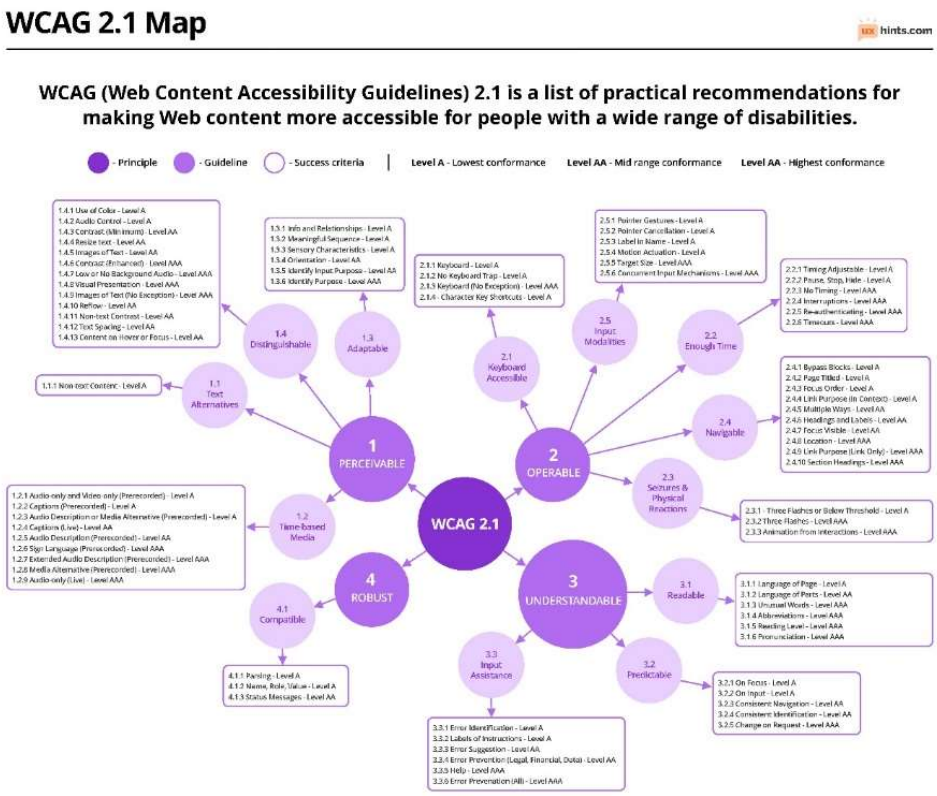


FIGURA 2 MAPPA WCAG 2.1[8]

[7] https://www.w3.org/WAI/standards-guidelines/wcag/
[8] https://uxhints.com/accessibility/wcag-2-1-map/

Sempre riferendosi al caso in precedenza, ossia alla necessità di fornire un testo alternativo alle immagini per le persone con disabilità visive, le WCAG inquadrano il caso in oggetto secondo questo schema:

- Principio: PERCEIVABLE
    - Linea guida: 1.1 Text Alternatives
        - Criterio di successo: 1.1.1 Non-text Content
            - Tecnica: H73 Using alt attributes on img elements[9]

1.3 Valutazione dell'accessibilità Web: i valutatori automatici

Per controllare se un sito Web o un'applicazione Web siano o meno conformi agli standard di accessibilità, è necessario condurre una valutazione dell'accessibilità. Questa valutazione può essere condotta attraverso diversi metodi:

1. Test automatizzato: tools, eseguiti in locale oppure online, che analizzano il codice della pagina per verificare la conformità ad un set di linee guida. Ma non tutte le linee guida possono essere tradotte in parti di codice da verificare, motivo per il quale, questi strumenti possono condurre la loro analisi soltanto in riferimento ad alcune di esse;
2. Ispezione manuale: analisi condotta da valutatori umani esperti. Questa modalità di analisi è altamente efficiente ma altrettanto dispendiosa in termini economici;
3. Test utente: questo metodo di valutazione si basa su test di usabilità condotti da utenti con particolari disabilità. Agli utenti è richiesto di compiere alcuni task e di fornire un riscontro sulla loro esperienza di utilizzo. Essendo un test empirico, i risultati ottenuti sono molto affidabili, ma come per

[9] https://www.w3.org/WAI/WCAG21/Techniques/html/H37

l'ispezione manuale questo metodo rischia di comportare un esoso esborso economico e di tempo[10].

Il progetto di tesi che questo scritto vuole descrivere, riguarda MAUVE++, il tool online per la validazione dell'accessibilità di un sito Web sviluppato all'interno dello *Human Interfaces In Information Systems Laboratory*[11] del ISTI-CNR (Consiglio Nazionale delle Ricerche) di Pisa[12].

Nonostante i tool automatici di valutazione non possano sostituire la valutazione umana esperta, questi ricoprono un ruolo decisamente importante a dispetto dei loro limiti.

I valutatori automatici dell'accessibilità presentano alcuni vantaggi, tra i quali[13]:

- Il processo di valutazione è decisamente meno oneroso in termini di spesa economica e di tempo;
- È possibile pianificare in modo più puntuale quale sforzo sarà necessario per una valutazione umana esperta, calcolandone dunque i costi;
- Una prima valutazione può essere condotta anche da valutatori umani poco esperti;
- I tool automatici di valutazione dell'accessibilità, se usati durante il processo di sviluppo, forniscono un quadro più organico dello stato di un prodotto Web, così da facilitare gli eventuali adempimenti da compiere nelle successive fasi di progettazione.

Come già accennato in precedenza, l'utilizzo di questi strumenti deve essere necessariamente accompagnato dalla conoscenza dei limiti degli stessi. La codifica di una linea guida e l'implementazione dell'algoritmo di ricerca e corrispondenza può produrre dei *falsi negativi* (elementi che non rispettano le linee guida ma che non vengono rilevati) oppure dei *falsi positivi* (lo strumento rileva dei problemi che

[10] Abascal J., Arrue M., Valencia X., (2019)
[11] https://hiis.isti.cnr.it/lab/home
[12] https://www.isti.cnr.it/it/
[13] Melody Y. Ivory and Marti A Hearst, (2001)

in realtà non sono presenti nel documento Web). L'efficacia di uno strumento automatico di valutazione dell'accessibilità può essere ricondotta a tre fattori: la *completezza*, la *correttezza* e la *specificità* dei risultati prodotti[14].

Lo sviluppo di uno strumento automatico di valutazione dell'accessibilità ha come nodo cruciale quello della traduzione in linguaggio computazionale delle linee guida che sono redatte in linguaggio naturale: quanto più questa rappresentazione informatica è aderente alle indicazioni delle linee guida, tanto più lo strumento automatico sarà capace di fornire un giudizio corretto. Oltre a questo, le linee guida possono avere un diverso livello di profondità e di specificità, dunque l'algoritmo di traduzione deve essere in grado di fornire la necessaria *flessibilità* nel processo di definizione della regola di valutazione.

Come accennato in precedenza, alcune linee guida non possono essere rappresentate in maniera accurata in linguaggio computazionale, è proprio per questo motivo che molti strumenti di valutazione forniscono due tipi di risultati:

1. Errori: quando nel codice vengono individuati delle devianze dalla linea guida presa in considerazione;
2. Warnings: quando una determinata porzione di codice individuata nella risorsa Web analizzata, potrebbe contenere una barriera di accessibilità, ma per la natura stessa della regola presa in esame, necessita di un controllo manuale perché non può essere verificata compiutamente dallo strumento automatico.

La WAI del W3C fornisce una lista degli strumenti automatici[15] disponibili in rete per la valutazione automatica dell'accessibilità di un contenuto Web. In questa lista sono elencati 159 strumenti, filtrabili secondo diversi criteri:

- Licenza: gratuita oppure commerciale;

[14] Giorgio Brajnik, (2004)
[15] https://www.w3.org/WAI/ER/tools/

- Linee guida: le linee guida che lo strumento può utilizzare per la valutazione;
- Lingua dello strumento;
- Tipologia dello strumento: API, authoring tool plugin, browser plugin, utilizzabile da linea di comando, applicazione desktop oppure mobile, strumento online;
- Formati supportati: lista di linguaggi di sviluppo Web per cui lo strumento è in grado di fornire una valutazione;
- Assistenza fornita: generazione di report della valutazione, possibilità di visualizzare gli errori direttamente sulla pagina Web, possibilità di modificare il codice della pagina durante l'utilizzo del tool;
- Campo di valutazione: singole pagine, un insieme di pagine oppure un intero sito Web.

Specificando alcuni parametri di filtraggio dei risultati in base ad alcune delle funzionalità offerte da MAUVE++, oggetto di questa tesi di laurea, il numero di strumenti elencati scende da 159 a 13.

I criteri di filtraggio dei risultati sono i seguenti:

- Possibilità di validare un documento Web secondo le linee guida *WCAG 2.0 — W3C Web Content Accessibility Guidelines 2.0* e *WCAG 2.1 — W3C Web Content Accessibility Guidelines 2.1*;
- Formati supportati: CSS, HTML;
- Possibilità di valutare un insieme di pagine;
- Gratuità del servizio.

Tra gli strumenti risultanti, oltre a MAUVE++, è possibile individuare *QualWeb*[16] della *Faculdade de Ciências da Universidade de Lisboa*[17] e *WAVE* di *WebAIM*[18]. Questi

[16] http://qualWeb.di.fc.ul.pt/evaluator/
[17] https://ciencias.ulisboa.pt/

tre strumenti sono stati oggetto di studio per la valutazione dell'affidabilità dei risultati tra gli strumenti di valutazione dell'accessibilità dei siti Web[19].

Lo studio in questione è utile a descrivere un altro limite di questi strumenti di valutazione automatica, ossia quello della consistenza dei risultati ottenuti, oltre a quello descritto in precedenza circa l'impossibilità di coprire tutti i casi previsti dalle linee guida.

L'obiettivo dello studio è quello di comparare i risultati ottenuti dai tre strumenti valutando alcune pagine Web con lo standard WCAG 2.1. L'analisi vuole mettere in evidenza l'affidabilità dei risultati *tra* gli strumenti, l'interrelazione, ossia quanto i risultati ottenuti dall'analisi di una stessa risorsa Web differiscano tra strumento e strumento.

Utilizzando la misura statistica *kappa* di Fleiss[20], utile a valutare l'accordo reale oltre il caso fra N osservatori quando è possibile assegnare valutazioni categoriche ai dati[21], lo studio evidenzia la scarsa inter-affidabilità dei risultati. Infatti, se l'indice k può assumere valori pari a -1 (massimo disaccordo) e 1 (massimo accordo), l'affidabilità dei risultati tra i tre strumenti risulta assumere un valore di 0,03.

È la WAI stessa a chiarire che "non è possibile controllare automaticamente tutti gli aspetti dell'accessibilità. È necessario il giudizio umano. A volte gli strumenti di valutazione possono produrre risultati falsi o fuorvianti. Gli strumenti di valutazione dell'accessibilità del Web non possono determinare l'accessibilità, possono solo aiutare a farlo"[22].

[18] https://Webaim.org/
[19] Maria Björkman, (2022)
[20] https://en.wikipedia.org/wiki/Fleiss%27_kappa
[21] https://it.knowledgr.com/02603384/KappaDiFleiss
[22] https://www.w3.org/WAI/test-evaluate/tools/selecting/

Pur tenendo in considerazione i limiti degli strumenti automatici di valutazione, essi giocano un ruolo chiave nello sviluppo di siti Web migliori, quindi più accessibili[23].

[23] Giorgio Brajnik. 2004

2 MAUVE++: lo strumento prima del progetto di tesi

Multiguideline Accessibility and Usability Validation Environment, MAUVE++ è un'ambiente per la valutazione e l'analisi di siti Web rispetto alla loro conformità ad un set di linee guida di accessibilità e usabilità[24]. Sviluppato all'interno dello *Human Interfaces In Information Systems Laboratory* del ISTI-CNR (Consiglio Nazionale delle Ricerche) di Pisa, MAUVE++ è inserito all'interno della lista fornita dal W3C nella quale vengono elencati gli strumenti di validazione automatica dell'accessibilità disponibili nel Web.

Prima di affrontare le nuove funzionalità che sono state progettate e implementate con questo progetto di tesi, si vuole descrivere brevemente il funzionamento di questo strumento[25].

I linguaggi informatici con cui è sviluppato MAUVE++ sono *Java*[26], *Javascript*[27], XML, *HTML*[28] e *CSS*[29] con l'integrazione di un database di tipo non relazionale, *MongoDB*[30]. Per la parte relativa all'interfaccia Web con la quale lo strumento è utilizzabile dall'utenza, le pagine Web sono generate dinamicamente attraverso la tecnologia *JavaServer Pages*[31] (JSP).

Il core dello strumento è composto: da un linguaggio astratto basato su XML con il quale è possibile tradurre in linguaggio computazionale le linee guida espresse in linguaggio naturale; dall'interprete di questo linguaggio, il quale mette in relazione le linee guida specificate e il codice della pagina Web oggetto di un'analisi di accessibilità. Il linguaggio basato su XML per la definizione delle linee guida si chiama *Language for Web Guideline Definition* (LWGD) ed è stato ideato con l'intento di non essere legato ad un particolare insieme di linee guida[32], ma al

[24] Schiavone, Paternò (2015)
[25] Per una trattazione più organica si rimanda a Schiavone (2012)
[26] https://www.java.com/it/
[27] https://it.wikipedia.org/wiki/JavaScript
[28] https://it.wikipedia.org/wiki/HTML
[29] https://it.wikipedia.org/wiki/CSS
[30] https://www.mongodb.com/it-it/company
[31] https://it.wikipedia.org/wiki/JavaServer_Pages
[32] Schiavone, Paternò (2015)

contrario, essere un linguaggio capace di tradurre dal linguaggio naturale a quello computazionale diverse tipologie di istruzioni.

Ad oggi MAUVE++ copre le tredici linee guida previste dalle WCAG 2.1, e settanta criteri di successo dei settantotto previsti dalle WCAG per il livello di conformità "AAA"[33].

L'utilizzo dello strumento può avvenire grazie ad un'interfaccia Web oppure attraverso un plugin per il Web browser Chrome. L'interfaccia Web permette di inserire il codice HTML da validare in molteplici modi: specificando l'indirizzo URL della risorsa Web, caricando direttamente dal proprio dispositivo un file contenente il codice oppure specificando direttamente il codice all'interno di un'area di testo dedicata.

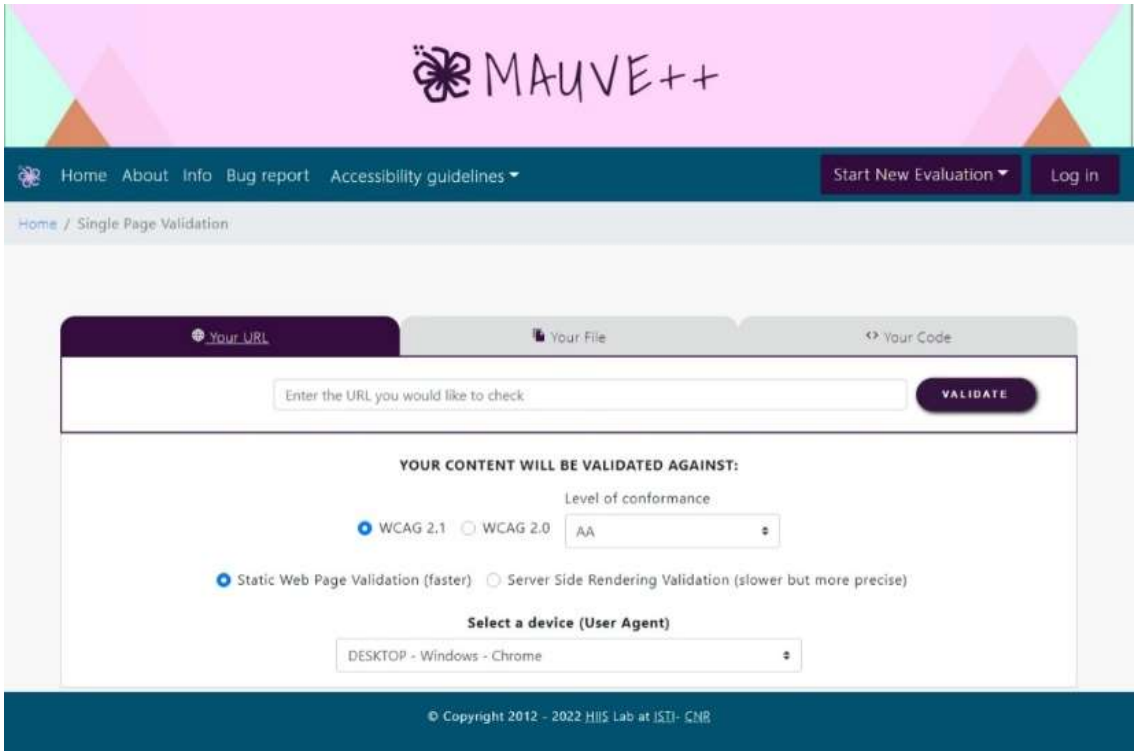


FIGURA 3 INTERFACCIA WEB DI MAUVE++ RELATIVA ALL'AVVIO DI UNA VALIDAZIONE DI UNA SINGOLA RISORSA

[33] https://www.essentialaccessibility.com/blog/Web-content-accessibility-guidelines

Lo strumento permette di selezionare il set di linee guida rispetto alle quali eseguire la validazione, è possibile scegliere tra le due versioni delle WCAG, la 2.0 e la 2.1, indicando anche il livello di conformità sulla base di tre livelli: A, AA, AAA.

Oltre alla linea guida, l'utente che si appresta ad avviare la validazione di una risorsa Web, ha la possibilità di specificare quale versione di questa risorsa debba essere oggetto di analisi attraverso l'indicazione del dispositivo, quindi desktop, tablet o mobile.

MAUVE++, prima dello sviluppo delle nuove funzionalità descritte in questa tesi di laurea, offriva due tipologie di validazione, ossia la *Single Web page evaluation* e la *Multipage evaluation*. La prima tipologia, mantenuta anche nella versione aggiornata da questo progetto, può essere avviata dall'utente senza effettuare la registrazione alla piattaforma e permette di passare in analisi una singola risorsa. La Multipage evaluation invece, ad oggi sostituita dalla funzionalità "Progetto" di tipo *simplified*, avvia l'analisi di accessibilità su un set di pagine Web, tutte appartenenti allo stesso dominio Web di riferimento. Questo set di pagine è creato dal *crawler*, un modulo di MAUVE++ che, utilizzando la URL di base specificata dall'utente, inizia a seguire i collegamenti scoperti all'interno della pagina e analizza iterativamente le nuove pagine scoperte. Il numero massimo di pagine e la profondità di "ricerca" sono due input che vengono forniti dall'utente nella fase di avvio di una Multipage evaluation e sono *Number of pages* e *Max depth*. La profondità di ricerca descrive la misura in cui questo modulo scopre le pagine di destinazione; per esempio, partendo dalla pagina di base "A", che si collega alla pagina "B", che si collega alla pagina "C", la struttura dei link sarebbe: A -> B -> C ->[...]; essendo "A" la pagina di partenza, avrà una profondità di 0, "B" avrà una profondità di 1 e così via.

Una volta terminata la validazione, sia di tipo Singlepage che Multipage, MAUVE++ genera il report in diversi formati: PDF, EARL, XML e tramite interfaccia Web. Focalizzando l'attenzione sull'interfaccia Web, perché più propriamente oggetto del progetto descritto in questo tesi di laurea, è possibile individuare tre diverse visualizzazioni del report di accessibilità: *Evaluation*

Summary, *End User*, *Web Developer*. Questa visualizzazione del report è sempre riferita ad una singola pagina Web, anche nel caso in cui la pagina fosse appartenuta ad una validazione multipagina: era possibile navigare tra i report delle varie pagine di una Multipage evaluation attraverso dei link (questa navigazione è stata mantenuta nella nuova tipologia di analisi sviluppata per questo progetto). Non era prevista una visualizzazione di insieme dei risultati ottenuti dall'analisi condotta sull'insieme delle pagine.

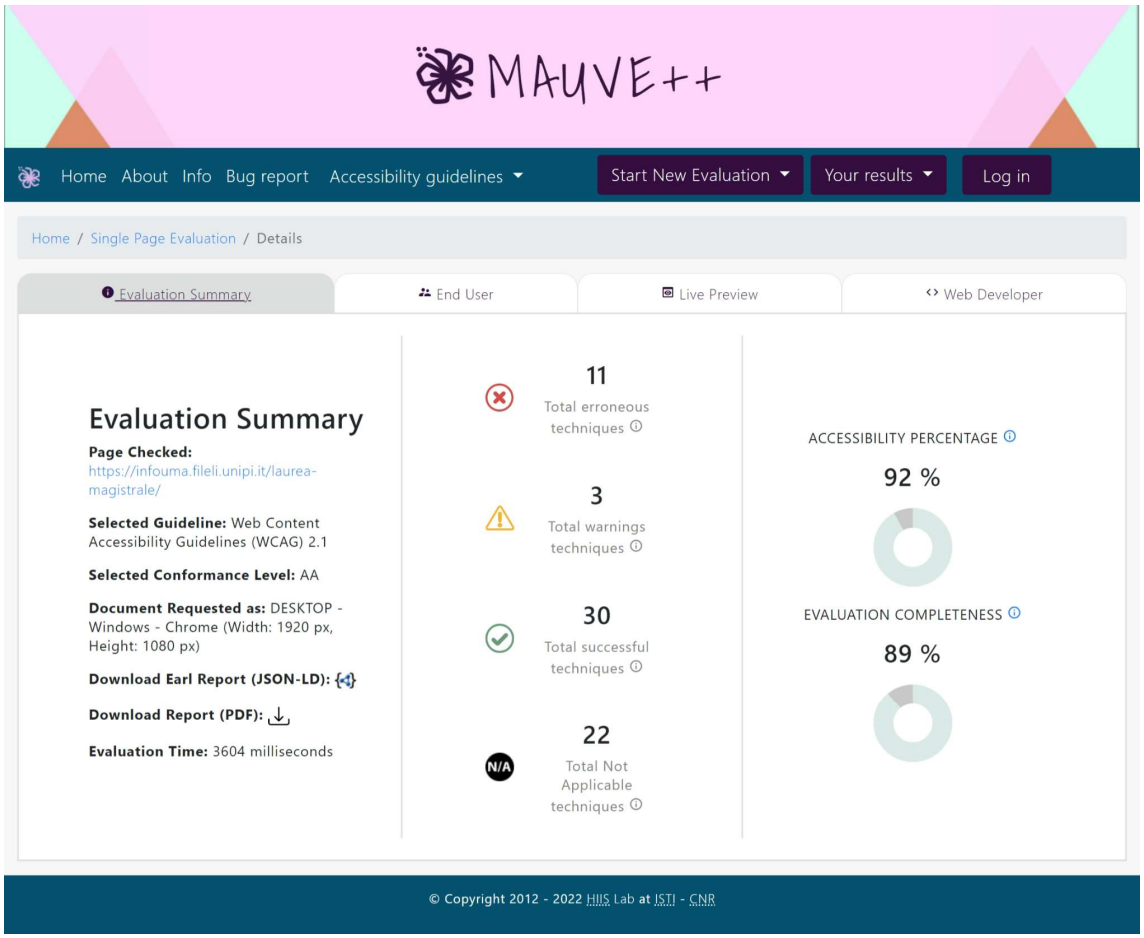


FIGURA 4 VISUALIZZAZIONE DEL REPORT IN MODALITÀ "EVALUATION SUMMARY"

La visualizzazione Evaluation Summary prevede tre colonne, la prima riassume i dati di configurazione di validazione di una singola pagina, riportando: la URL della pagina analizzata; le linee guida, il livello di conformità e l'user agent scelti dall'utente; i link di download del report in formato EARL e PDF; il tempo impiegato dallo strumento per eseguire la validazione. Nella seconda colonna

vengono mostrati i risultati della validazione attraverso l'indicazione del numero di tecniche analizzate distinte per la tipologia del risultato ottenuto: errors, warnings, success. Nella versione aggiornata di MAUVE++ è stata aggiunta un'ulteriore indicazione, ossia le tecniche WCAG che non sono applicabili nella validazione perché il documento HTML e/o CSS in analisi non presenta elementi a cui poter sottoporre i controlli di queste tecniche. Nella terza colonna della visualizzazione Evaluation Summary sono visualizzate le due metriche quantitative riassuntive di MAUVE++: *Accessibility Percentage* ed *Evaluation Completeness*.

Queste due metriche forniscono un'indicazione concisa sul corrente livello di accessibilità (*accessibility percentage*) e allo stesso tempo ricordano agli utenti che la valutazione potrebbe non essere completa (*evaluation completeness*).

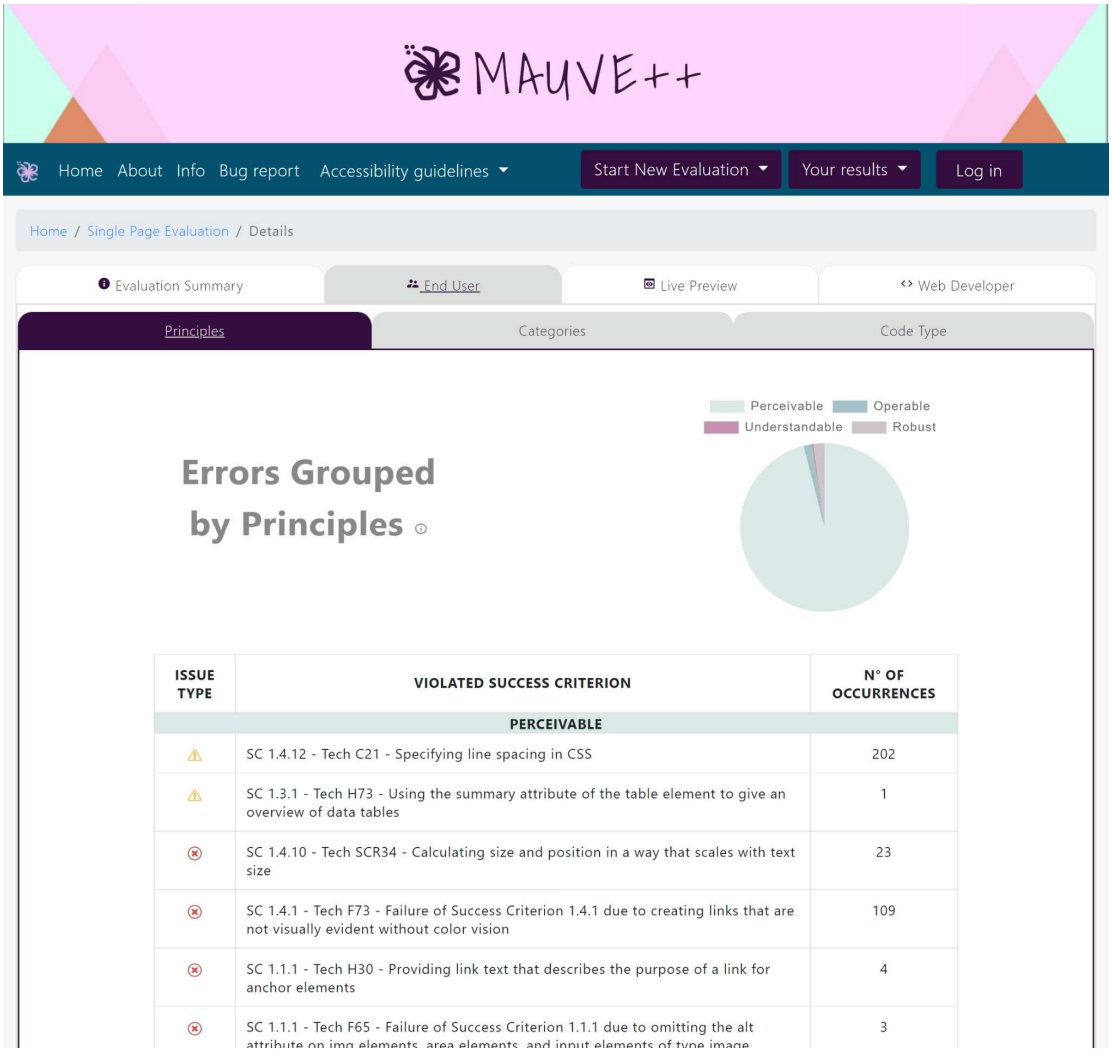
Il calcolo delle due metriche è basato sui valori:

- S = numero degli elementi valutati che hanno superato positivamente il controllo;
- E = numero degli elementi valutati che sono stati valutati come erronei;
- W = numero degli elementi che richiedono una valutazione umana.

Entrambe le metriche sono percentuali, e vengono calcolate come segue:

- *Accessibility percentage* = S/(S+E): numero degli elementi che hanno superato i controlli diviso il numero di tutti gli elementi per cui il validatore ha potuto dare un giudizio (positivo o negativo). Per una maggiore specificità di questo risultato, gli errori individuati dal validatore non hanno tutti lo stesso peso ai fini del calcolo di questa metrica, infatti, riferendosi ai tre livelli di conformità, "A", "AA" e "AAA", MAUVE++ applica un coefficiente di rilevanza che partendo da 1 per il grado "A", passa al valore di 0,6 per il grado "AA" e di 0,2 per il grado tripla A. Questo, in modo tale da dare un maggior peso agli errori compresi nei gradi di conformità più bassi richiesti;

- *Evaluation completeness* = (S+E)/(S+E+W): numero totale degli elementi per cui il validatore è riuscito ad arrivare ad un giudizio (positivo o negativo) diviso tutti gli elementi che sono stati presi in considerazione (compresi quelli per cui non è stato espresso un giudizio es. warnings).



- **FIGURA 5 VISUALIZZAZIONE DEL REPORT IN MODALITÀ "END USER"**

La visualizzazione del report "End User" è dedicata all'utenza dello strumento che non ha particolari competenze nello sviluppo Web, ma che ha comunque interesse nel visualizzare le problematiche di accessibilità attraverso diversi punti di vista. Infatti, questo tipo di visualizzazione offre tre diverse viste ognuna delle quali raggruppa le problematiche secondo un proprio criterio: *Principles*, problematiche raggruppate secondo i principi WCAG di appartenenza delle tecniche valutate

come “errors” o “warnings”; *Categories*, qui le problematiche sono raggruppate sulla base dell’appartenenza degli elementi HTML o CSS della pagina Web analizzata, rispetto a delle tipologie predefinite come “link”, “table”, “form” etc; *Code Type*, il raggruppamento delle problematiche ruota attorno alla tipologia di codice sulla quale la stesse sono state individuate, dunque HTML o CSS. Tutte e tre le viste basate su un diverso principio di raggruppamento, offrono un grafico riassuntivo e una tabella nella quale la problematica è descritta attraverso l’indicazione della tipologia, dunque “errors” o “warnings”, il titolo della tecnica violata e il numero di occorrenze della violazione che sono state riscontrate nell’analisi.

La terza visualizzazione del report offerta da MAUVE++ è dedicata agli sviluppatori Web. Infatti, la vista “Web Developer” mostra il codice HTML della pagina Web analizzata annotando sopra l’elemento che presenta una problematica la tecnica violata e il criterio WCAG di riferimento. Questa annotazione contiene anche un link diretto alla documentazione WCAG relativa alla tecnica violata.

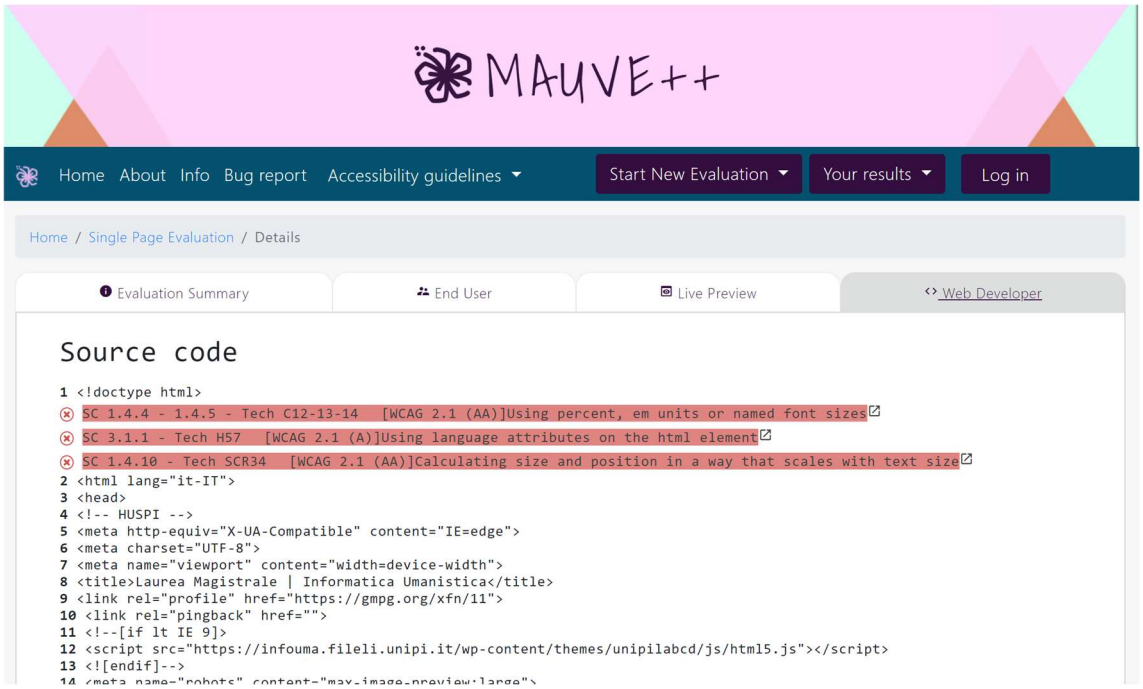


FIGURA 6 VISUALIZZAZIONE DEL REPORT IN MODALITÀ "WEB DEVELOPER"

Tutte e tre le visualizzazioni, “Evaluation Summary”, “End User” e “Web Developer” sono state mantenute nella nuova versione di MAUVE++ sviluppata per questo progetto di tesi.

3 MAUVE++ 2.2: i requisiti, “Live preview”, “Server Side Rendering” e il monitoraggio periodico dell’accessibilità

3.1 I requisiti per una nuova generazione di strumenti automatici di valutazione dell’accessibilità Web

Nella fase iniziale della progettazione del “nuovo” MAUVE++, sì è pensato ad alcuni requisiti generali che uno strumento di valutazione dell’accessibilità Web dovrebbe avere per essere efficace in un contesto ampio e sempre in mutazione come quello del Web. Questi requisiti possono essere elencati e descritti come segue:

1. *Lo strumento dovrebbe essere utile a diversi tipi di utenti.* Nella creazione di una risorsa Web solitamente sono coinvolte diversi tipi di persone con una differente formazione, ad esempio il committente, lo sviluppatore, alcune volte affiancato da un UI designer. Sono poche le volte in cui nel team di sviluppo è presente un esperto di accessibilità Web. Per questo motivo lo strumento dovrebbe essere in grado di fornire informazioni utili e comprensibili a tutti le personalità coinvolte e non soltanto tecnicismi comprensibili ai pochi;

2. *Lo strumento dovrebbe essere in grado di validare singole pagine e gruppi di pagine Web.* Potrebbe essere utile validare una singola pagina perché è particolarmente interessante fornire ad essa un alto grado di accessibilità, oppure, come nel caso in cui la risorsa Web appartenga ad apparati statali e quindi investiti dalla direttiva EU 2016/2012, è più opportuno validare un insieme di pagine o un intero sito Web. Proprio in ottemperanza alla direttiva sopra citata, l’Unione europea ha adottato una chiara metodologia (EU 2018/1524[34]) per la validazione di un sito Web di una pubblica istituzione, fornendo due modalità: *in-depth*, ossia la validazione di alcuni

[34] https://eur-lex.europa.eu/legal-content/IT/TXT/?toc=OJ:L:2018:256:FULL&uri=uriserv:OJ.L_.2018.256.01.0108.01.ITA

aspetti fondamentali di un sito Web come, ad esempio, l'interazione con i forms, i messaggi di dialogo e i vari feedbacks scaturiti dall'interazione dell'utente; *simplified*, l'altra modalità di analisi che deve includere dei test relativi a tutti gli aspetti di accessibilità;

3. *Lo strumento dovrebbe essere in grado di monitorare periodicamente il livello di accessibilità.* L'aggiornamento dei contenuti di un sito Web è una cosa molto frequente, dunque, il livello di accessibilità potrebbe subire variazioni nel corso del tempo. Inoltre, la già citata direttiva europea 2012 del 2016 all'articolo 8, prevede che gli stati membri monitorino il grado di accessibilità delle proprie applicazioni Web e che provvedano ad un report concreto di questa attività di monitoraggio;

4. Lo strumento dovrebbe essere in grado di identificare gli errori individuati sia sul codice della pagina che in un'interfaccia che riproduca la stessa pagina Web. L'identificazione della porzione di codice che presenta un errore è di fondamentale importanza per una più presta risoluzione del problema, d'altra parte visualizzare l'elemento che presenta l'errore direttamente sulla pagina Web oggetto di analisi può fornire l'effettivo impatto che la violazione della regola di accessibilità ha sull'utente che visita la pagina Web;

5. *Lo strumento dovrebbe fornire indicazioni riassuntive sullo stato di accessibilità*. Generalmente i risultati di una validazione di accessibilità eseguita da uno strumento automatico si riducono ad una lunga lista delle problematiche riscontrate. Questo tipo di visualizzazione non permette di stabilire una valutazione di massima sul grado di accessibilità della risorsa Web, motivo per il quale sarebbe utile elaborare una formula che riconduca le problematiche riscontrate e le loro occorrenze a valori quantitativi di più facile lettura;

6. *Lo strumento dovrebbe essere trasparente circa la sua capacità di analisi.* Il già citato studio sull'inter-affidabilità dei risultati (Björkman 2022), mostra

come i risultati generati da diversi tool su uno stesso set di pagine siano molto discordanti tra di loro. Questo dipende dalle tecniche diverse con cui gli strumenti analizzano il codice e dalla diversa quantità di linee guida per cui ogni strumento è in grado di fornire una valutazione. Esprimere in modo chiaro queste informazioni potrebbe diminuire il disorientamento che gli utilizzatori di questi strumenti provano di fronte a risultati così discordanti;

7. *Lo strumento dovrebbe fornire indicazioni pratiche sulle possibili soluzioni ai problemi individuati*. Fornire un accesso diretto ad informazioni tecniche relative allo specifico problema in analisi può favorire il processo di correzione;

8. *Lo strumento dovrebbe essere in grado di validare pagine Web dinamiche*. Sempre più spesso i siti Web vengono sviluppati attraverso frameworks Javascript che modificano profondamente il contenuto della pagina Web durante il processo di caricamento della stessa nel browser Web. I contenuti generati dovrebbero essere analizzati dal validatore per fornire dei risultati più completi e dunque più corretti.

Per tutte queste ragioni, il progetto di tesi descritto in questo elaborato ha riguardato lo sviluppo di diverse soluzioni che permettono a MAUVE++ di ottemperare ad alcuni dei requisiti sopra esposti.

3.2 Server side rendering

Con il crescente utilizzo di librerie capaci di creare dinamicamente le pagine Web sfruttando l'elaborazione client side attraverso i browsers Web, si è resa necessaria l'elaborazione di un sistema di *Server-side rendering* delle pagine Web.

Una pagina Web creata con framework come React[35], AngularJS[36] o Vue.js[37] genera il suo contenuto dopo l'elaborazione client side di uno script:

[35] https://it.reactjs.org/
[36] https://angularjs.org/

1. il *client* invia la richiesta di una risorsa Web ad un *server*;
2. il server risponde inviando una piccola porzione di codice Javascript contenente le istruzioni per l'elaborazione del contenuto della pagina Web richiesta;
3. il client, attraverso il browser installato sulla macchina, elabora il codice e renderizza la pagina Web sul dispositivo.

Prendendo ad esempio l'home page del sito Web dello *Human Interfaces In Information Systems Laboratory* del ISTI-CNR (Consiglio Nazionale delle Ricerche), realizzato con AngularJS, il codice che il server invia al client per la generazione della pagina è composto da 107 righe. Dopo l'elaborazione del browser lato client, il codice della pagina Web passa a 514 righe.

Con il servizio *Static Web page Validation*, MAUVE++ invia una richiesta ad un servizio REST che scarica il codice HTML e CSS della pagina, senza nessun tipo di elaborazione. Questo codice poi, viene sottoposto all'analisi e alla valutazione.

Facendo sempre riferimento alla homepage del sito dell'HIIS Laboratory, l'immagine di seguito mostra la pagina Web che Mauve++ effettivamente analizza dopo aver ricevuto il codice HTML e CSS attraverso il servizio Static Web page Validation:

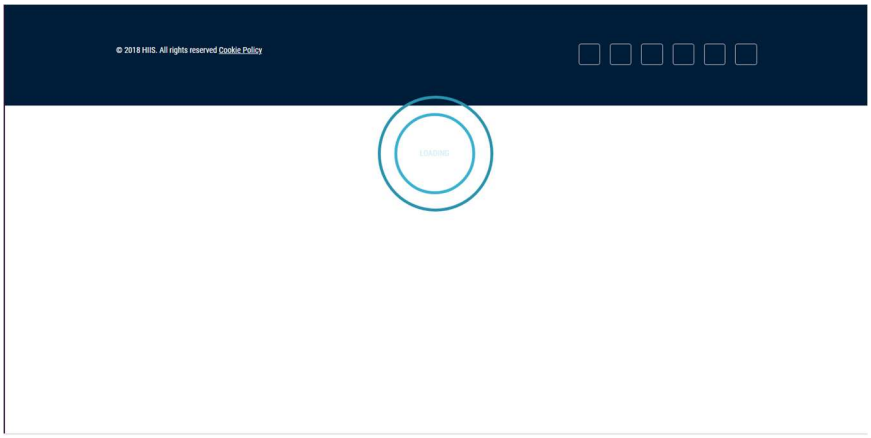


FIGURA 7 HOMEPAGE DEL SITO DEL HIIS LABORATORY SENZA L'ELABORAZIONE BROWSER CLIENT SIDE

[37] https://vuejs.org/

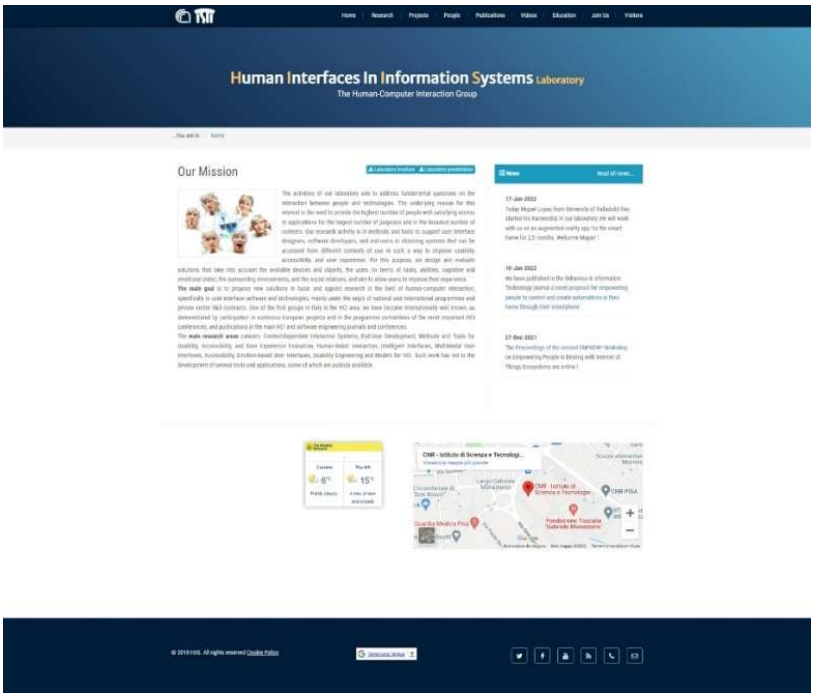


FIGURA 8 HOMEPAGE DEL SITO DEL HIIS LABORATORY DOPO L'ELABORAZIONE CLIENT SIDE

Questa discrepanza evidente tra le due immagini della stessa risorsa Web, si riflette sui risultati delle valutazioni: quello che MAUVE++, con il servizio di Static Web page Validation (figura 7), analizza non è ciò che realmente viene visto dall'utenza del sito Web (figura 8) e dunque i risultati sull'accessibilità sono sostanzialmente erronei.

Per superare questo inconveniente è stato sviluppato il servizio di *Server side rendering Validation.*

Utilizzando la libreria *Puppeteer*[38] per Node.js[39], quando viene avviata una richiesta di valutazione attraverso l'interfaccia Web di MAUVE++, sul server viene aperta un'istanza headless[40] di un browser basato su Chromium[41], che simula il comportamento del browser di un utente che visita la pagina Web specificata nel campo URL della richiesta. Quando il browser completa il caricamento di tutte le

[38] https://github.com/puppeteer/puppeteer
[39] https://nodejs.org/it/
[40] Modalità di esecuzione priva dell'interfaccia grafica.
[41] https://www.chromium.org/Home/

risorse della pagina, l'HTML risultante, anche quello eventualmente generato da script, viene restituito a MAUVE++ che inizia l'elaborazione.

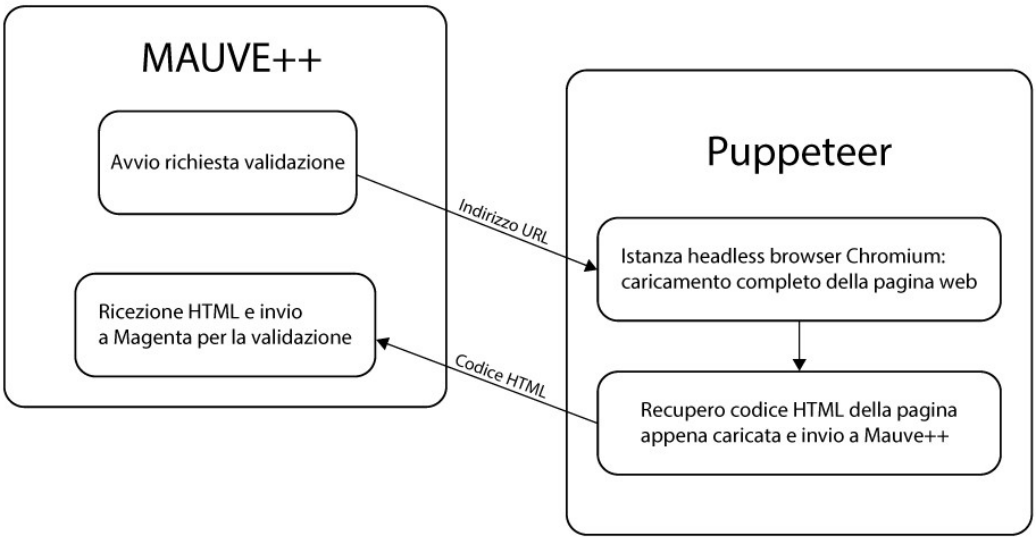


FIGURA 9 FLUSSO SERVER SIDE RENDERING VALIDATION: RECUPERO CODICE HTML PAGINA WEB DA VALIDARE 1

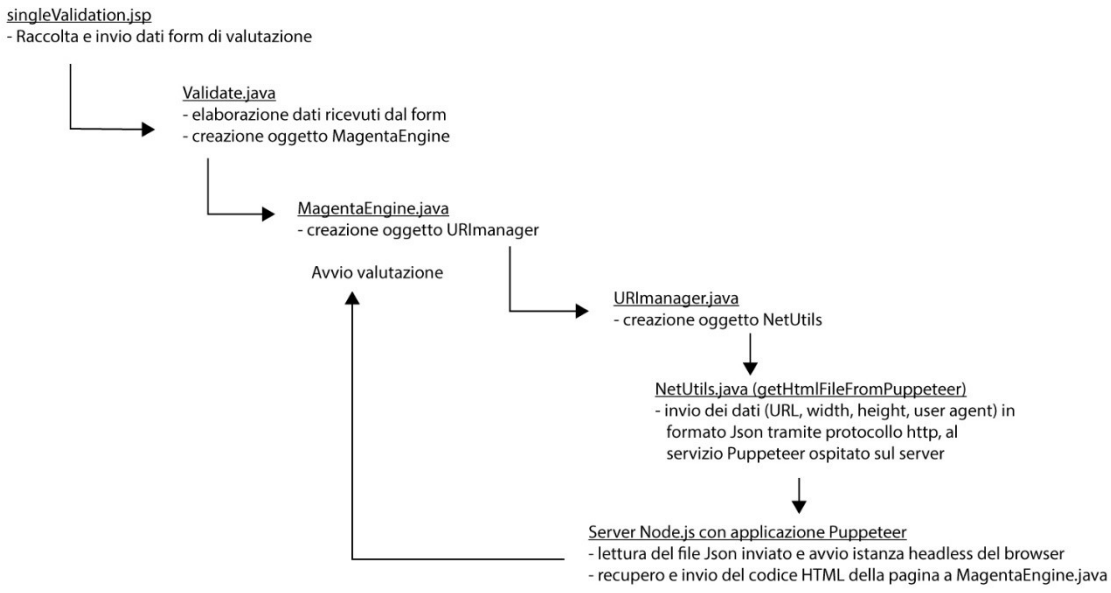


FIGURA 10 FLUSSO SERVER SIDE RENDERING VALIDATION: RECUPERO CODICE HTML PAGINA WEB DA VALIDARE 2

Il risultato pratico di questa applicazione del server side rendering è reso evidente dai dati risultanti di due valutazioni eseguite su una stessa risorsa Web, ossia l'homepage del sito Web dello HIIS Laboratory visto in precedenza. Una delle due valutazioni è stata eseguita con il servizio di server side rendering, l'altra con il precedente sistema di recupero del codice HTML e CSS da analizzare:

	ERRORS	WARNINGS	SUCCESS	NOT APPLICABLE
SERVER SIDE RENDERING	7	3	28	28
STATIC WEB PAGE VALIDATION	3	1	17	45

Per sottolineare l'incremento in positivo della capacità di analisi attraverso il nuovo sistema, si prenda come esempio il dato sulle tecniche *not applicable*, ossia quelle tecniche che il validatore segnala come non applicabili quando la pagina Web non presenta gli elementi sui quali andrebbero a ricadere. Le tecniche sulle quali MAUVE++ fornisce l'analisi sono in totale 66, nella valuatzione effettuata con l'HTML "statico" 45 di esse risultano essere non applicabili, con la renderizzazione della pagina lato server il loro numero scende a 28. Questo incremento delle capacità di valutazione fornisce dei dati certamente più corretti sull'accessibilità della pagina Web presa in esame.

3.3 "Live preview": visualizzazione grafica e interattiva del report

3.3.1 L'interfaccia grafica

Una nuova funzionalità sviluppata per questo progetto di tesi riguarda il live previewing della pagina Web analizzata dal validatore.

Altri strumenti di valutazione automatica dell'accessibilità disponibili nel Web e presenti nella lista pubblicata dal W3C[42], vista in precedenza, presentano la visualizzazione dinamica degli errori direttamente sulla pagina Web oggetto di analisi, ad esempio WAVE[43] oppure il plugin[44] per il browser Chrome sviluppato da Siteimprove[45].

Per lo sviluppo di questo nuovo componente, si è partiti dalla progettazione dell'interfaccia che sarebbe stata inserita all'interno di quella già presente di MAUVE++.

Il problema principale che si è affrontato in fase di progettazione riguarda l'organizzazione degli spazi per garantire una più facile consultazione del report unitamente ad una rappresentazione grafica della pagina Web presa in esame.

Si è deciso di dividere in due lo spazio a disposizione, fornendo a sinistra una colonna più stretta dove vengono mostrati i risultati del report e sulla destra uno spazio più ampio che ospita la riproduzione della pagina Web in analisi.

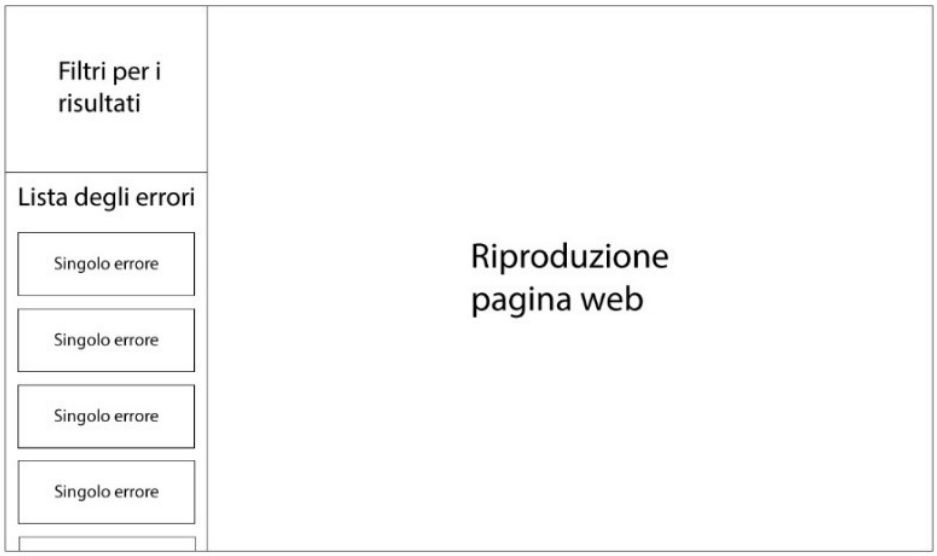


FIGURA 11 PRIMO MOCKUP DELL'INTERFACCIA PER LA "LIVE PREVIEW"

[42] https://www.w3.org/WAI/ER/tools/
[43] https://wave.Webaim.org/
[44] https://chrome.google.com/Webstore/detail/siteimprove-accessibility/djcglbmbegflehmbfleechkjhmedcopn
[45] https://siteimprove.com/

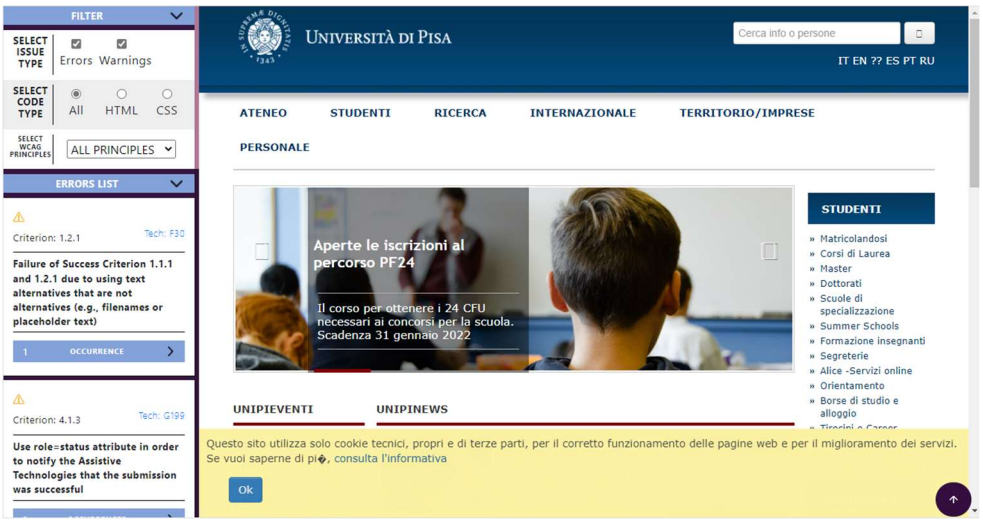


FIGURA 12 INTERFACCIA "LIVE PREVIEW"

L'interfaccia è stata sviluppata senza l'utilizzo di librerie dedicate al UI design ma attraverso codice custom, sfruttando le potenzialità delle proprietà *grid*[46] e *flexbox*[47] di CSS3[48].

Per la visualizzazione degli errori, vista la probabile presenza di diverse occorrenze per ogni errore, si è deciso di optare per un sistema ad accordion (fisarmonica), così da garantire una più ordinata visualizzazione qualora gli errori da presentare siano in gran numero.

[46] https://css-tricks.com/snippets/css/complete-guide-grid/
[47] https://css-tricks.com/snippets/css/a-guide-to-flexbox/
[48] https://www.w3schools.com/css/

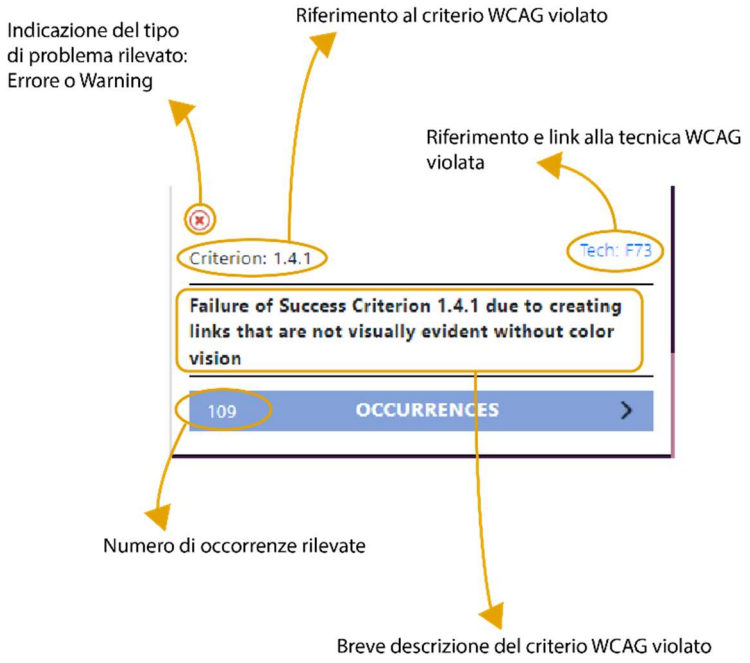


FIGURA 13 DESCRIZIONE ELEMENTI DELLA LISTA DEGLI ERRORI (1)

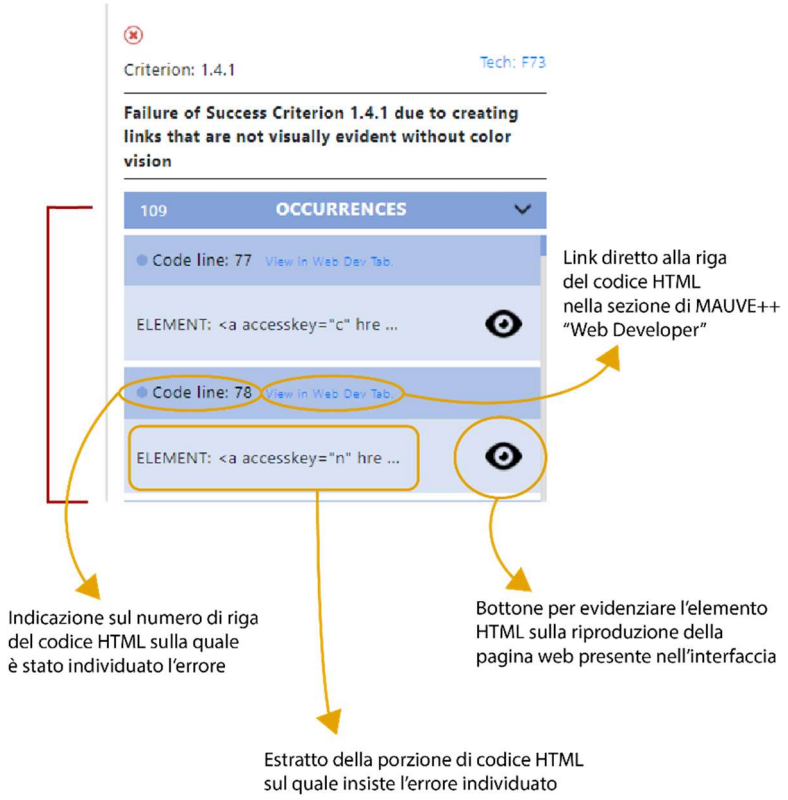


FIGURA 14 DESCRIZIONE ELEMENTI DELLA LISTA DEGLI ERRORI (2)

Le opzioni di filtraggio dei risultati ricalcano le categorizzazioni già presenti in MAUVE++:

- Tipo di problema: Errore o Warning;
- Tipo di codice sul quale si è riscontrato il problema: HTML o CSS;
- Principio WCAG di appartenenza dell'errore: Perceivable, Operable, Understandable, Robust.

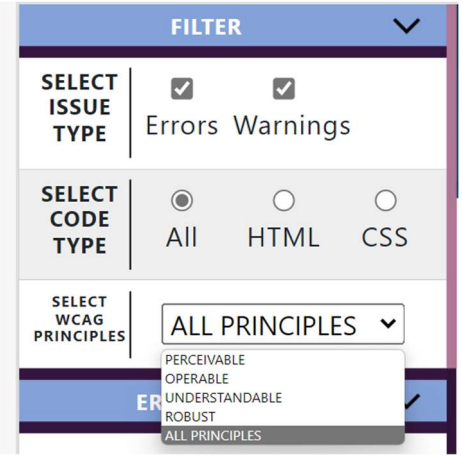


FIGURA 15 OPZIONI DI FILTRAGGIO DELLA LISTA DEGLI ERRORI

Alla modifica di uno dei parametri di filtraggio, la lista degli errori si aggiorna automaticamente mostrando soltanto gli elementi che rispecchiano il valore impostato dai vari filtri.

L'evidenziazione dell'errore avviene al momento del click dell'utente sull'icona a forma di occhio presente nella tab dedicata alla singola occorrenza dell'errore.

FIGURA 16 VISUALIZZAZIONE DELL'ERRORE DIRETTAMENTE NELLA PAGINA WEB RIPRODOTTA

3.3.2 La fase di programmazione: alcuni aspetti dell'implementazione

Dopo aver deciso quali fossero le informazioni che avrebbero trovato spazio nell'interfaccia della "Live preview", si è passati allo sviluppo di uno script in grado di recuperare queste informazioni e di organizzarle in uno schema determinato.

I dati necessari per la costruzione dell'interfaccia, relativi ai soli errori e warnings, sono:

- Principio a cui appartiene il criterio violato;
- Nome del criterio violato;
- Breve descrizione del criterio violato;
- Tecnica violata;
- Numero di occorrenze dell'errore:
    - Numero di linea del codice HTML sulla quale si è verificata la segnalazione del problema;

    - Porzione di codice HTML sul quale si è verificato l'errore riscontrato;
    - XPath relativo all'elemento della pagina HTML che presenta l'errore;
- Codice HTML della pagina Web da mostrare nell'iframe.

Per ottenere le informazioni necessarie, si è deciso di sviluppare uno script in Javascript capace di elaborare il report in formato JSON[49] che MAUVE++ genera al momento della validazione di un pagina Web. La funzione è *core(a)*, il parametro è il report in formato JSON generato dalla valutazione.

Il risultato di questa elaborazione è un nuovo documento JSON con questa struttura:


```
"err": [
        {
            "msg": null,
            "errInfo": {
                "errLine": [
                    {
                        "line": "107,1",
                        "xpathPointer":"//html/body/div[2]/header/div/div[2]/div/div[1]/div/form/fieldset/input"
                    }
                ]
            },
            "type": "e",
            "iderr": "h98-00",
            "occurrence": 1,
            "checkedElements": 8
        }
    ],
    "pass": [
        {
            "passInfo": {
                "passLine": [
                    {
                        "line": "1070,1",
                        "xpathPointer": "//html/body/div[7]/div/div/div[2]/div/button"
                    }
                ]
            },
            "occurrence": 1,
            "checkedElements": null
        }
    ],
    "id": "F78",
    "summary": "Failure due to styling element outlines and borders in a way that removes or renders non-visible the visual focus indicator",
    "category": "form",
    "checkpointsresult": "N",
    "idcriterion": "1.4.11 2.4.7"
  },
"err": [...
```


FIGURA 17 PORZIONE DI CODICE JSON DOPO L'ELABORAZIONE DEL REPORT DI VALIDAZIONE FORNITO DA MAUVE++. LA PORZIONE DI CODICE IN FIGURA MOSTRA UN PROBLEMA DI TIPO ERROR

Alcune delle proprietà dell'oggetto JSON in figura 17, sono utili a comprendere come queste informazioni vengano utilizzate per la creazione dell'interfaccia Live preview:

- "errLine";
    - "line": numero di linea del codice HTML;

[49] https://www.json.org/json-it.html

    - "xpathPointer": percorso xPath dell'elemento su cui si è verificato l'errore;

- "type": assume il valore di "e" in caso di errore, di "w" in caso di warning;
- "occurrence": numero di elementi della pagina che presentano la problematica;
- "id": identificativo della tecnica WCAG violata;
- "summary": titolo della tecnica WCAG violata;
- "checkpointresult": può assumere valori come "A" che indica il superamento del controllo, "W" che indica lo status di warning e "N" che definisce lo stato di errore della tecnica presa in considerazione;
- "idcriterion": identificativo del criterio (o dei criteri) WCAG, a cui appartiene la tecnica violata.

Questo file JSON viene utilizzato in un *for loop* il quale, per ogni oggetto "err" descritto in precedenza, richiama la funzione *createListItemError(a, b)* che crea gli elementi della lista degli errori dell'interfaccia Live preview.

Un altro aspetto dell'implementazione di questa funzionalità che si vuole approfondire è quello relativo alla visualizzazione all'interno di un tag *iframe*, della pagina Web oggetto di analisi.

Come visto in precedenza, MAUVE++ attraverso la Static Web page Validation o la Server side rendering Validation, recupera il codice HTML della pagina Web di cui si richiede la valutazione. Questo codice HTML viene salvato in un file sul server con un identificativo univoco, lo stesso identificativo è riportato all'interno del report JSON generato da MAUVE++ dopo la valutazione.

La visualizzazione della pagina Web oggetto di analisi, all'interno dell'interfaccia Live preview di MAUVE++, ha presentato fin da subito delle problematiche che non hanno permesso di utilizzare l'indirizzo URL della pagina stessa come valore

dell'attributo *href* di un tag *iframe*, all'interno del quale visualizzare il documento Web.

Queste problematiche sono dovute alla *Same Origin Policy* (SOP). La SOP è una politica di sicurezza del browser Web generale per le richieste cross-origin. Controlla l'accesso ai dati tra siti Web e applicazioni Web. Se non ci fosse SOP, qualsiasi pagina Web e qualsiasi codice JavaScript sarebbe in grado di accedere al DOM di altre pagine HTML: questo aspetto rappresenterebbe una falla nella sicurezza[50]. A volte lo scambio di risorse tra server diversi è utile, un esempio può essere quello relativo ai font Web ospitati su piattaforme come Google Fonts[51], che permettono di includere nella propria pagina Web un determinato set di caratteri senza che questi siano effettivamente ospitati sul proprio server, oppure ancora le mappe di Google Maps spesso incluse nelle pagine Web. Il meccanismo che permette di superare SOP è il *Cross-origin resource sharing* (CORS). Il CORS permette a un server di indicare qualsiasi origine (dominio, schema o porta) diversa dalla propria, da cui un browser dovrebbe permettere il caricamento di risorse[52]. I permessi di accesso a risorse Web da origini diverse da quelle nelle quali le stesse sono ospitate, sono specificati nell'intestazione HTTP che il server invia come risposta ad una richiesta. Nel momento in cui una pagina Web ospitata sul server A richiede contenuti ospitati su un server B, il CORS può o meno permettere di raggiungere queste risorse.

Il tag *iframe* di HTML, permettere di integrare pagine Web all'interno di altre pagine Web, inserendo nell'attributo *src* l'indirizzo URL della pagina da integrare. Nel caso della Live preview di MAUVE++, le pagine da integrare sono quelle sottoposte all'analisi di accessibilità. Integrando le pagine tramite l'attributo *src* si incorrerebbe inevitabilmente in errori CORS; quindi, le pagine Web integrate non avrebbero mostrato il loro aspetto reale, perché molti contenuti delle stesse non sarebbero stati caricati correttamente all'interno dell'iframe.

[50] https://www.coretech.it/it/service/articoli/articoli.php?ID=1425
[51] https://fonts.google.com/
[52] https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS

Si è deciso, dunque, di sfruttare un altro attributo del tag *iframe*, ossia *srcdoc*. Quest'attributo permette di specificare direttamente il codice HTML da mostrare all'interno dell'iframe. Recuperando il codice HTML che MAUVE++ ha utilizzato per la valutazione di accessibilità è possibile inserirlo come valore dell'attributo *srcdoc*, superando così le limitazioni SOP e CORS e mostrando correttamente la pagina Web integrata.

A tal proposito è stata creata la funzione *getHTML(a, b)* la quale ha il compito di recuperare il file contenente l'HTML che MAUVE++ ha precedentemente salvato, e modificarlo opportunamente affinché sia correttamente visualizzato.

```
function getHTML(baseUrl, REPORT) {
    console.log("Live Preview GETHTML "+baseUrl);
    let resource_id = REPORT.contextInfo.evalInfo.resourceId;
    let page_url = REPORT.results.pageReport[0].src;
    let base_tag = document.createElement("base");
    base_tag.setAttribute("href", page_url);
    let link_css = document.createElement("link");
    link_css.setAttribute("rel", "stylesheet");
    link_css.setAttribute("href", baseUrl + "/live_preview/styles/iframe_style.css");
    fetch("./api/v1/testSubject/" + resource_id + "?toolID=mauve&source", {
        method: "GET",
        headers: {
            'Content-Type': 'text/html'
        }
    })
        .then(response => response.text()
                    .then(data => str = data)
                    .then(() => {
                        let parser = new DOMParser();
                        let doc = parser.parseFromString(str, 'text/html');
                        let html = doc.getElementsByTagName("html")[0];
                        html.classList.add("cursor_not_allowed");
                        let body = doc.getElementsByTagName("body")[0];
                        let head = doc.getElementsByTagName("head")[0];
                        let scripts = doc.getElementsByTagName("script");
                        var arr_scripts = [].slice.call(scripts);
                        arr_scripts.forEach(element => {
                            element.setAttribute("src", "");
                            element.textContent = "";
                        });
                        let imgs = doc.getElementsByTagName("img");
                        let arr_imgs = [].slice.call(imgs);
                        let objects = doc.getElementsByTagName("object");
                        let arr_objects = [].slice.call(objects);
                        let links = doc.getElementsByTagName("link");
                        let arr_links = [].slice.call(links);
                        arr_imgs.forEach(element => {
                            if (element.hasAttribute("onerror")) {
                                element.setAttribute("onerror", "");
                                element.textContent = "";
                            }
                        });

                        arr_objects.forEach(element => {
                            if (element.hasAttribute("onerror")) {
                                element.setAttribute("onerror", "");
                                element.textContent = "";
                            }
                        });
                        arr_links.forEach(element => {
                            if (element.hasAttribute("onerror")) {

                                element.setAttribute("onerror", "");
                                element.textContent = "";
                            }
                        });
                        head.prepend(base_tag);
                        head.append(link_css);
                        let str_modified = doc.all[0].outerHTML;
                        iframe.setAttribute("srcdoc", str_modified);
                    }));

    let iframe = document.querySelector("#iframe");

    iframe.addEventListener("load", () => {
        core(REPORT);
    });

}
```

FIGURA 18 CODICE DELLA FUNZIONE GETHTML(A, B)

Tramite la *fetch* API[53], viene eseguita una richiesta di tipo GET richiedendo la pagina HTML da visualizzare, questa viene restituita sottoforma di stringa per cui è necessario creare un oggetto Javascript *DOMparser*[54] per ricreare il DOM della pagina.

A questo punto è possibile navigare all'interno del DOM della pagina da visualizzare nell'iframe per operare le modifiche necessarie, come:

- Aggiungere il tag *base*[55] nell'*head* della pagina affinché gli eventuali *path* delle risorse presenti, espressi in modo relativo[56] vengano correttamente reindirizzati. Il valore dell'attributo *href* del tag *base* assume il valore della *base URL*[57] della pagina da visualizzare. Ad esempio, se la pagina in oggetto fosse quella raggiungibile all'indirizzo *example.com*, potrebbe integrare un'immagine specificando soltanto il percorso relativo, quindi *src="/images/image.jpg"*. Il percorso completo, dunque il path assoluto, per raggiungere l'immagine, sarebbe invece *http://example.org/images/image.jpg*. Qualora la risorsa venisse caricata all'interno di un iframe in una pagina con un dominio diverso, la URL relativa non permetterebbe il corretto indirizzamento alla risorsa. Grazie al tag *base* con valore di *href="example.org"*, inserito nell'*head* della pagina, le risorse vengono correttamente ricercate nella loro effettiva posizione anche se caricate in un altro dominio Web;

- Disabilitare gli script affinché non modifichino l'aspetto della pagina durante la visualizzazione o ne permettano l'uscita.

Al termine dell'elaborazione, questo DOM viene convertito in stringa di testo e attribuito come valore all'attributo *srcdoc* dell'iframe della Live preview, così da permetterne la visualizzazione. Dopo il completo caricamento del contenuto

[53] https://developers.google.com/Web/updates/2015/03/introduction-to-fetch
[54] https://developer.mozilla.org/en-US/docs/Web/API/DOMParser
[55] https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base?retiredLocale=it
[56] https://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/
[57] https://Webtech.training.oregonstate.edu/faq/what-base-url

dell'iframe, viene chiamata la funzione *core(a)* per l'elaborazione dell'interfaccia come visto in precedenza.

Un ultimo aspetto dell'implementazione di questa funzionalità di live previewing che si vuole affrontare è quello relativo all'individuazione e all'evidenziazione dell'errore nella pagina Web riprodotta nell'iframe. Si affronterà in seguito il caso particolare in cui l'elemento che presenta una problematica, errore o warning, non sia visibile nella pagina Web e dunque non evidenziabile.

Come visto in precedenza, all'interno della lista degli errori presente sulla sinistra della schermata del Live preview, tutte le occorrenze dei problemi rilevati nella validazione hanno una propria tab, nella quale oltre alle indicazioni circa l'errore, si può trovare un bottone a forma di occhio che permette di evidenziare l'errore direttamente sulla pagina Web riprodotta a fianco.

Il *listener* dell'evento *click* è associato al bottone a forma di occhio nel momento della creazione della tab stessa, come spiegato nelle precedenti pagine, grazie alla funzione *createListItemError(a, b)*.

La funzione richiamata al click sul bottone è *addButtonAction(a, b)*, i cui parametri sono:

a. l'elemento HTML dello stesso bottone cliccato;

b. il link alla documentazione WCAG riguardante la tecnica violata.

```
function addButtonAction(error_occurrence_button, link_tech_wcag) {
    let path, element, error_container, previous_alert, previous_alert_iframe;
    if (document.querySelector("#invite")) {
        document.querySelector("#invite").remove();
    }

    path = error_occurrence_button.getAttribute("data-x");
    element = iframe.contentDocument.evaluate(path, iframe.contentDocument, null, XPathResult.FIRST_ORDERED_NODE_TYPE, null).singleNodeValue;

    error_container = error_occurrence_button.closest('.single_error_container');
    previous_alert = document.querySelectorAll(".mauve_alerts_border");
    previous_alert_iframe = iframe.contentDocument.querySelectorAll(".mauve_alerts_border_iframe");
    if (document.querySelector("#tooltip")) {
        document.querySelector("#tooltip").remove();
    }

    if (isInViewport(element)) {
        element.scrollIntoView({behavior: "smooth", block: "center", inline: "center"});
        if (previous_alert.length != 0) {
            previous_alert[0].classList.remove("mauve_alerts_border");
        }
        if (previous_alert_iframe.length != 0) {
            previous_alert_iframe[0].classList.remove("mauve_alerts_border_iframe");
        }
        setTimeout(() => {
            error_container.classList.add("mauve_alerts_border");
            element.classList.add("mauve_alerts_border_iframe");
            element.setAttribute("data-q", link_tech_wcag);
            addTooltip(element)
        }, 1);
    } else {
        if (previous_alert.length != 0) {
            previous_alert[0].classList.remove("mauve_alerts_border");
        }
        if (previous_alert_iframe.length != 0) {
            previous_alert_iframe[0].classList.remove("mauve_alerts_border_iframe");
        }
        setTimeout(() => {
            error_container.classList.add("mauve_alerts_border");
        }, 1);
        let invite_open_dev_tools, code_line_link;
        code_line_link = document.createElement("a");
        code_line_link.setAttribute("href", "#");
        code_line_link.setAttribute("title", "Go to Web Developer Tab");
        code_line_link.textContent = "view code line " + error_occurrence_button.getAttribute("data-line") + " in Web Developer tab.";
        code_line_link.addEventListener("click", () => {
            openWebDeveloperTab(error_occurrence_button.getAttribute("data-line"));
        });
        invite_open_dev_tools = document.createElement("span");
        invite_open_dev_tools.setAttribute("id", "invite");
        invite_open_dev_tools.textContent = "This element is not visible on page. Please, ";
        invite_open_dev_tools.append(code_line_link);
        error_container.append(invite_open_dev_tools);
    }
}
```

FIGURA 19 CODICE DELLA FUNZIONE ADDBUTTONACTION(A, B)

La funzione *addButtonAction(a, b)* si occupa di applicare (o di rimuovere) delle classi CSS, che attribuiscono all'elemento al quale vengono associate, le regole di stile capaci di evidenziare in rosso l'errore selezionato e contestualmente l'elemento della pagina Web riprodotta.

All'interno della funzione un *if else statement* permettere di variare il comportamento dell'interfaccia in adeguamento alla visibilità dell'elemento da evidenziare.

Il controllo sulla visibilità dell'elemento nella pagina Web riprodotta è affidato alla funzione *isInViewport(a)*, il parametro della funzione è l'elemento HTML della pagina Web riprodotta all'interno dell'iframe.

```
function isInViewport(element) {
    if (window.getComputedStyle(element).visibility == "hidden" || window.getComputedStyle(element).display == "none") {
        return false
    }
    const rect = element.getBoundingClientRect();
    let iframe_height = iframe.contentWindow.document.getElementsByTagName('body')[0].scrollHeight;
    let iframe_width = iframe.contentWindow.document.getElementsByTagName('body')[0].scrollWidth;
    if (rect.width < 2) {
        return false
    } else if (window.getComputedStyle(element).width < 2) {
        return false
    } else if (rect.height < 2) {
        return false
    } else if (window.getComputedStyle(element).height < 2) {
        return false
    } else if (rect.top < 0 && rect.left < 0 && (rect.bottom > (iframe_height)) && (rect.right > (iframe_width))) {
        return false
    } else {
        return true
    }
}
```

FIGURA 20 CODICE DELLA FUNZIONE ISINVIEWPORT(A)

Il primo controllo presente nella funzione *isInViewport(a)* è relativo a due proprietà che l'elemento potrebbe assumere rendendolo invisibile in pagina:

- visibility;
- display.

Questo controllo è effettuato sfruttando il metodo *getComputedStyle()* [58] che restituisce un oggetto che contiene tutte le proprietà CSS di un elemento HTML.

Se l'elemento ha la proprietà *visibility* con valore *hidden*, oppure la proprietà *display* con valore *none*, l'elemento non sarà visibile in pagina. Se questo è vero, la funzione *isInViewPort(a)* restituisce il valore *false* alla funzione *addButtonAction(a, b)* che provvederà a processare questa informazione agendo di conseguenza, come si vedrà nelle pagine seguenti.

Se il primo controllo è superato, e dunque l'elemento non possiede le proprietà che lo rendono invisibile nella pagina, vengono eseguiti ulteriori controlli, perché l'elemento potrebbe essere o troppo piccolo per essere visualizzato, oppure posizionato al di fuori della *viewport*[59] dell'iframe.

Innanzitutto, vengono definite tre variabili:

[58] https://developer.mozilla.org/en-US/docs/Web/API/Window/getComputedStyle
[59] https://developer.mozilla.org/en-US/docs/Web/CSS/Viewport_concepts

1. *rect*: contiene un oggetto *DOMRect*[60] che provvedere a fornire tutte le informazioni relative alla grandezza in pixel di un elemento, e alla sua posizione relativa alla *viewport*;
2. *iframe_height*: contiene un valore numerico pari all'altezza in pixel della pagina ospitata nell'iframe;
3. *iframe_width*: contiene un valore numerico pari all'ampiezza in pixel della pagina ospitata nell'iframe.

I controlli eseguiti sono, nell'ordine:

- l'elemento da evidenziare non sia più piccolo di due pixel sia in larghezza che in altezza;
- l'elemento non sia posizionato al di fuori dell'area visibile della pagina riprodotta nell'iframe.

Se uno di questi controlli non viene superato, la funzione restituisce il valore *false*, altrimenti viene restituito il valore *true*.

La funzione *addButtonAction(a, b)*, una volta ricevuto questo valore dalla funzione *isInViewport(a)*, prosegue il suo operato.

Se il valore ricevuto è *true*, e dunque l'elemento è visibile in pagina, viene applicata all'elemento all'interno della pagina nell'iframe, la classe CSS che aggiunge uno sfondo rosso e un bordo lampeggiante affinché sia ben visibile. Viene aggiunto inoltre il link alla documentazione tecnica WCAG relativa all'errore sollevato. Questo link viene utilizzato all'interno del popup *How to solve?* che appare all'evento *on hover* sull'elemento evidenziato. Viene evidenziata in egual modo la tab dell'errore selezionato all'interno della lista degli errori.

[60] https://developer.mozilla.org/en-US/docs/Web/API/DOMRect

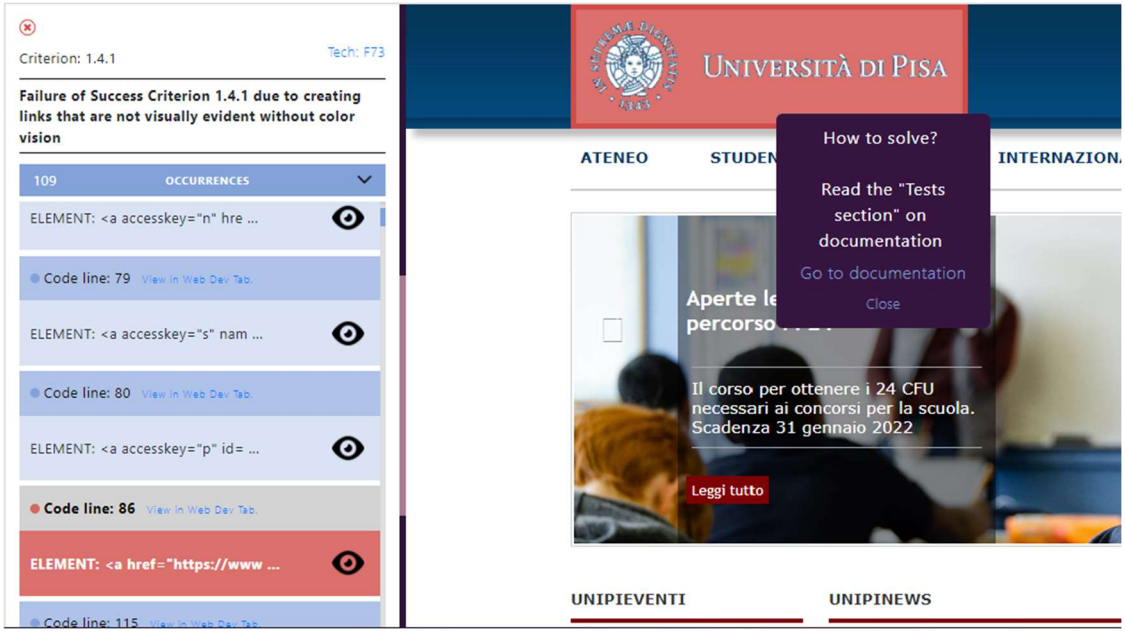


FIGURA 21 EVIDENZIAZIONE DELL'ELEMENTO CHE PRESENTA PROBLEMI DI ACCESSIBILITÀ

Se l'elemento non è visibile in pagina, e dunque il valore ricevuto dalla funzione *isInViewport(a)* è *false*, viene creato un avviso per invitare l'utente a visualizzare l'elemento che presenta l'errore nella tab *Web Developer*.

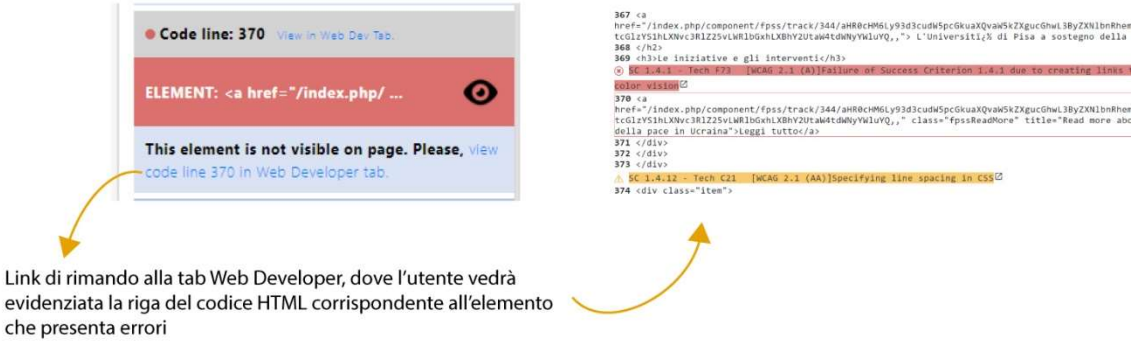


FIGURA 22 INTERFACCIA LIVE PREVIEW E WEB DEVELOPER: VISUALIZZAZIONE DI UN ERRORE NON VISIBILE DIRETTAMENTE IN PAGINA

3.4 Monitoraggio periodico: Progetto, Audit, Page

La direttiva (UE) 2016/2102 del Parlamento europeo e del Consiglio del 26 ottobre 2016 relativa all'accessibilità dei siti Web e delle applicazioni mobili degli enti pubblici[61], considerando la crescente tendenza alla digitalizzazione della società e la rapida crescita del mercato per il conseguimento di una maggiore accessibilità di prodotti e servizi digitali, grazie ad operatori economici che sviluppano siti Web o strumenti software per creare, gestire ed effettuare test di pagine Web o applicazioni mobili, afferma tra le altre cose, che la conformità alle prescrizioni in materia di accessibilità dovrebbe essere sottoposta a monitoraggio periodico[62].

Come visto nel capitolo 1, gli strumenti automatici di validazione dell'accessibilità non possono, da soli, fornire una valutazione completa sull'accessibilità di un sito Web, ma possono garantire una panoramica valida per riscontrare velocemente e a costo zero o quasi, lo stato della risorsa Web in relazione agli standard di accessibilità.

Per tali ragioni si è deciso di dotare MAUVE++ di una nuova funzionalità che permetta il monitoraggio periodico di un set di pagine.

Si è pensato ad un nuovo modello di task per l'applicazione che prevedesse la creazione di un "Progetto", impostando i vari parametri di valutazione, e la successiva presa visione dei risultati con la possibilità di comparare diverse valutazioni eseguite ad intervalli regolari di tempo l'una dall'altra.

3.4.1 L'architettura del tool

Si è pensato alla nuova articolazione delle entità che compongono il "progetto": un *progetto* raccoglie una serie di validazioni, chiamate *audit*, che sono eseguite a intervalli regolari su uno specifico gruppo di *pagine*.

[61] https://eur-lex.europa.eu/eli/dir/2016/2102/oj
[62] DIRETTIVA (UE) 2016/2102, L 327/7, (50)

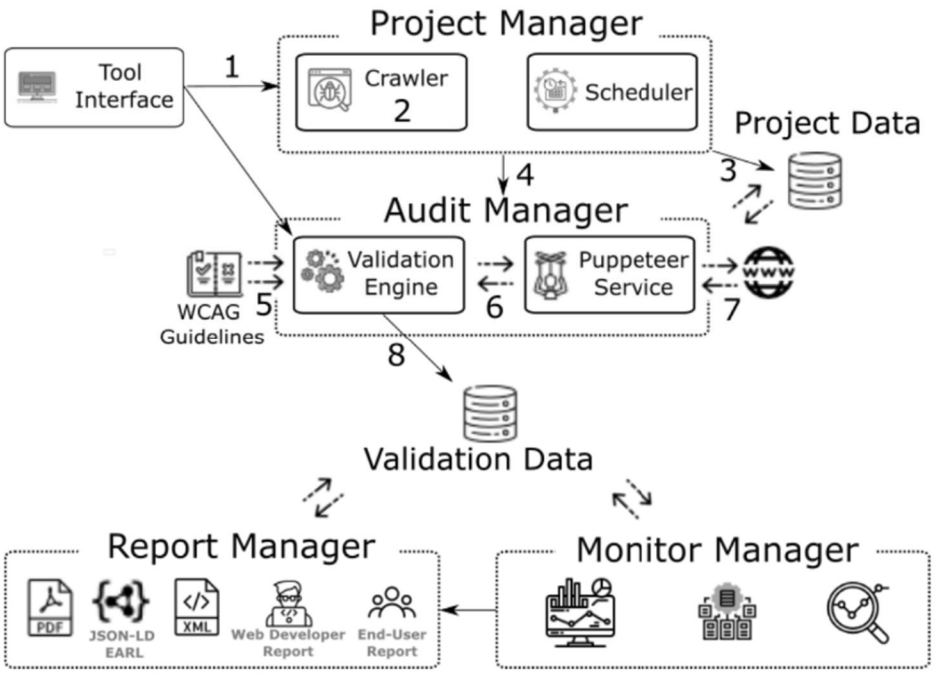


FIGURA 23 ARCHITETTURA DEL TOOL

La figura 23 mostra i vari moduli che giocano un ruolo nella creazione del progetto e nel monitoraggio dei risultati dello stesso:

- *Project manager*. Raccogliendo le opzioni di validazione fornite dall'utente, recupera le pagine da analizzare attraverso il *crawler* (qualora le pagine non siano specificate manualmente) e pianifica le validazioni automatiche periodiche;

- *Audit manager*. Si occupa della validazione delle pagine interagendo con specifici set di linee guida. Si occupa anche del recupero del codice HTML, sia nel caso in cui venga richiesto l'apporto del servizio di server-side rendering, sia nella modalità Static Web page Validation;

- *Monitor manager*. Recupera i dati delle singole validazioni, li aggrega e fornisce interfacce interattive nelle quali l'utente può analizzarli e monitorare l'evoluzione del grado di accessibilità delle pagine oggetto dell'analisi;

- *Report manager*. Questo modulo si occupa della creazione dei report di validazione in vari formati.

Facendo sempre riferimento alla figura 23, è possibile descrivere il flusso di lavoro durante il processo di creazione e validazione di un progetto: (1) l'utente può creare il progetto utilizzando l'interfaccia Web, scegliendo i parametri di validazione e il tipo di progetto. Un progetto può essere di tre tipi, *simplified* dove l'utente specifica una URL da cui il crawler partirà per la ricerca delle pagine in base ai valori di Max Depth e Number of Pages specificate (2); di tipo *single page* dove l'analisi è eseguita su una sola pagina specificata dall'utente; *in-depth* dove l'utente specifica un set di pagine escludendo così il lavoro di crawling. (3) Una volta raccolte le URLs e i parametri di validazione, questi dati vengono salvati in un documento del database.

I dati raccolti vengono poi forniti all'Audit manager che si occupa dell'effettiva validazione (4). Questa analisi può partire immediatamente dopo la creazione del progetto, oppure può essere eseguita in un determinato giorno della settimana diverso da quello della creazione. Inoltre, la validazione può essere eseguita una sola volta oppure ad intervalli regolari. Di questa programmazione periodica se ne occupa il servizio *Scheduler*.

Una volta iniziata la validazione, il *Validation Engine* mette in relazione (5) il set di linee guida selezionato dall'utente durante la creazione del progetto e i documenti HTML e CSS delle pagine Web recuperate attraverso le URLs specificate nel progetto.

Tra le possibili opzioni di validazione, l'utente può scegliere lo *user agent* con il quale richiedere il contenuto HTML e CSS della pagina Web, ad esempio è possibile scegliere tra la versione desktop o quella mobile della pagina stessa. Ad ogni user agent fa riferimento una viewport diversa, dunque le regole di stile specificate nel documento CSS associato possono o meno trovare applicazione. La corretta identificazione delle regole di stile che effettivamente agiscono sulla

renderizzazione della pagina Web, è compito di un HTML&CSS parser[63] che identifica le *media queries*[64] del codice CSS corrispondenti alla viewport del dispositivo selezionato con lo user agent.

Il recupero del codice HTML (e CSS) avviene attraverso il servizio Puppeteer (6 e 7) come descritto in precedenza (3.2).

Al termine della validazione, i dati vengono salvati sul database e utilizzati dal Report manager per fornire all'utente i risultati dell'analisi sia attraverso documenti scaricabili come PDF, EARL[65] o XML, sia attraverso l'interfaccia Web del tool che permette di interagire con i dati proposti.

3.4.2 La progettazione della nuova struttura del database

Il primo intervento riguarda la strutturazione del database in base alle nuove esigenze che l'applicazione richiede, a tal proposito si è progettato un nuovo modello entità-relazioni per ridefinire il database.

Il database è sviluppato con il *database management system*[66] (DBMS) *MongoDB*, un tipo di database definito *NoSQL*[67] in quanto, a differenza dei più diffusi *relational database management system*[68] (RDBMS) basati sul linguaggio SQL, i documenti che compongono il database sono definiti in linguaggio JSON e non presentano vincoli di relazione tra le varie entità che lo compongono.

Il database di MAUVE++ prima delle modifiche prevedeva tre documenti:

- Users: contenente i dati relativi agli utenti registrati;
- SinglePageEvaluation: raccoglie i dati relativi alle valutazioni di accessibilità nella modalità "Single page evaluation";
- CrawlMultiPage: raccoglie i dati relativi alle valutazioni di accessibilità eseguiti su un set di pagine non prestabilito.

[63] https://github.com/radkovo/jStyleParser
[64] https://www.w3schools.com/css/css3_mediaqueries.asp
[65] https://www.w3.org/TR/EARL10-Schema/
[66] https://it.wikipedia.org/wiki/Database_management_system
[67] https://it.wikipedia.org/wiki/NoSQL
[68] https://www.oracle.com/database/what-is-a-relational-database/

Le nuove entità individuate attorno alle quali plasmare il database sono:

- Projects: contiene tutte le informazioni circa il progetto, dal titolo dello stesso ai parametri di valutazione selezionati;
- Audits: rappresenta la singola validazione eseguita sul set di pagine specificate all'interno del progetto;
- Pages: rappresenta la singola pagina di un audit. Tra le informazioni contenute in questo documento, si ha il report in formato JSON generato da MAUVE++ al momento della valutazione.

Tabella delle entità:

ENTITÀ	DESCRIZIONE	ATTRIBUTI	IDENTIFICATORE
Users	Previa registrazione, può creare un progetto	_id; FirstName; Surname; Email; Password; Organization; Role; Country; MailList; SecurityQuestion; SecurityAnswer; isAdmin	_id
Projects	Unità base. Definisce i criteri della validazione e raccoglie i vari audit	_id; project_date; user_id; country; lang; sender; guideline; conformance_level; device_type; title; type; baseUrl; state; URLs; frequency; delay; Last audit; Next audit; scheduled; day; height; width	_id
Audits	Unità del progetto. Contiene i dati complessivi relativi ai risultati della validazione delle singole pagine	_id; project_id_object; user_id; state; accPercentage; avg_errors; avg_warnings; compPercentage; tot_cp_errors; tot_cp_success; tot_cp_warnings; tot_occ_errors; tot_occ_success;	_id

		tot_occ_warnings; audit_date	
Pages	Singola unità dell'audit	_id; project_id_object; audit_id_object; user_id page_title; url; cp_errors; occ_errors; cp_warnings; occ_warnings; cp_success; occ_success; data_type; report; vd_pageLocalCopy	_id

Tabella delle relazioni:

RELAZIONE	DESCRIZIONE	COMPONENTI	ATTRIBUTI
Creazione	Un utente crea un progetto	Users; Projects	
Appartenenza	Un audit appartiene ad un progetto	Audits; Projects	
Inclusione	Una pagina Web appartiene ad un progetto	Pages; Audits	

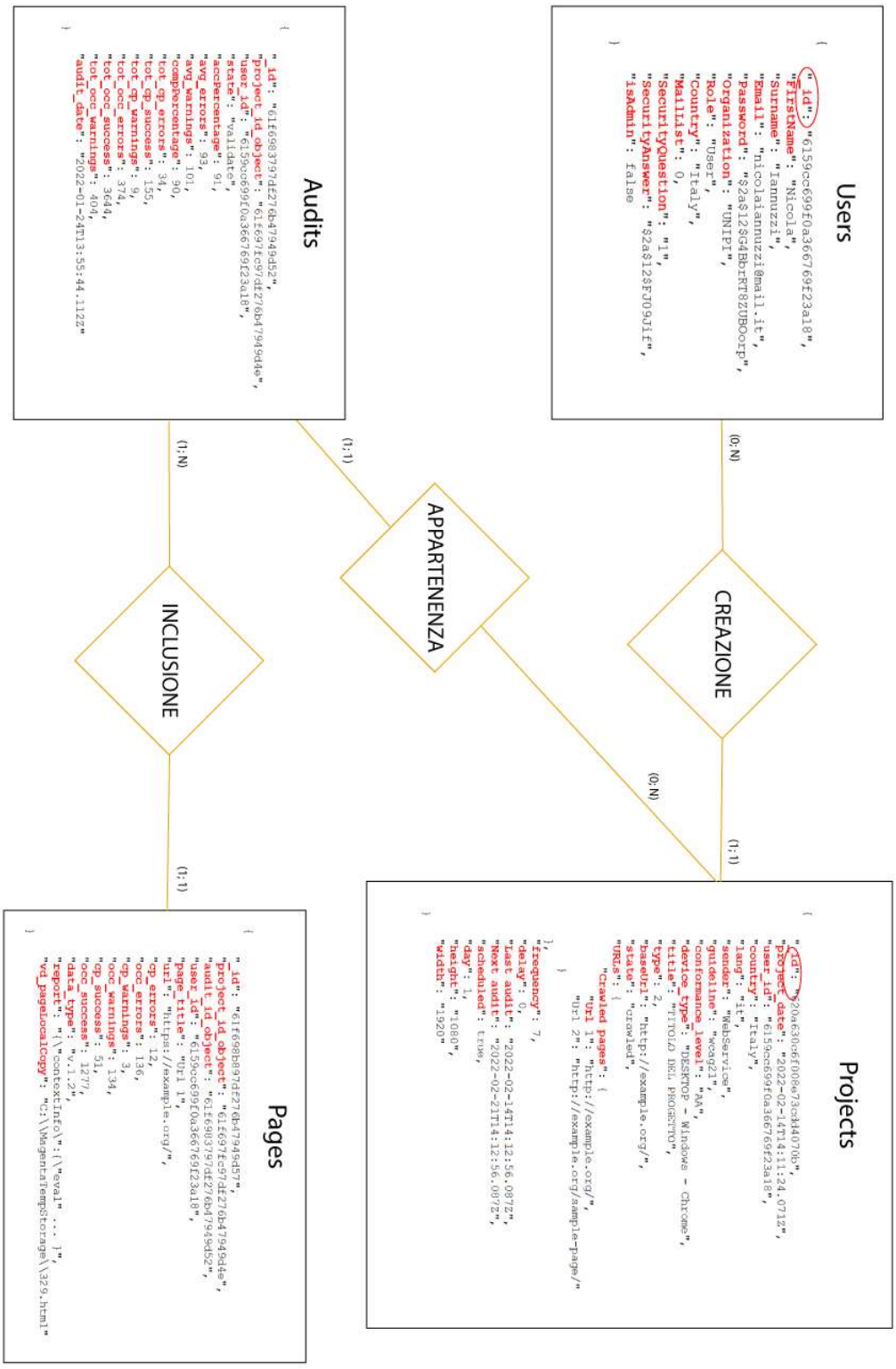


FIGURA 24 MODELLO E/R DATABASE MAUVE++. IL DOCUMENTO "SINGLE PAGE" È RIMASTO INALTERATO E NON PRESENTA NESSUNA RELAZIONE CON LE ALTRE ENTITÀ DEL DATABASE, DUNQUE NON È STATO RIPORTATO IN QUESTO MODELLO

Sulla base del modello e/r appena presentato, dopo aver definito le necessarie regole di vincolo e di derivazione per alcune proprietà dei documenti del database, si è proceduto all'implementazione.

Lo sviluppo della nuova funzionalità si è articolata su due fronti: lo sviluppo del backend, nei linguaggi Java e Javascript, per la gestione e l'elaborazione dei dati; lo sviluppo del frontend, con la tecnologia di programmazione Web *JavaServer Pages*[69] (JSP) per fornire a MAUVE++ l'interfaccia necessaria grazie a contenuti dinamici HTML, Javascript e CSS.

3.4.3 Implementazione della nuova funzionalità: il backend

Per implementare la funzionalità "Progetto" lato backend, è stato creato un nuovo pacchetto Java all'interno del progetto *MauveWeb*: *Project*.

Le classi che compongono questo pacchetto sono:

- Audit.java
- CrawlerAudit.java
- PageFetcherAudit.java
- EvalAudit.java
- Project.java

Le classi *CrawlerAudit.java* e *PageFetcherAudit.java* si occupano di recuperare le pagine da validare quando l'utente sceglie il tipo di progetto "Simplified": l'utente specifica una sola URL da cui il sistema di crawling deve cominciare la ricerca di altre pagine del sito Web. Il numero massimo di pagine e la profondità delle stesse sono sempre specificate dall'utente nella maschera di creazione del progetto.

La classe *Project.java* è una *servlet*[70] che si occupa di gestire gli input in ingresso del form di creazione del progetto. Tutti i dati inseriti nella maschera di creazione del

[69] https://it.wikipedia.org/wiki/JavaServer_Pages
[70] https://www.html.it/articoli/primi-passi-con-le-servlet/

progetto vengono inviati alla servlet in formato JSON tramite una chiamata AJAX[71] di tipo POST. Una volta ricevuti questi dati, la classe *Project.java* compie fondamentalmente due operazioni:

1. Inserimento dei dati nel database attraverso la creazione di un nuovo documento all'interno della collezione *Projects*;
2. Avvio della validazione.

L'avvio della validazione può non essere eseguito nello stesso momento di creazione del progetto; infatti, l'utente ha la possibilità di scegliere il giorno della settimana nel quale eseguire la validazione. Qualora il giorno scelto non sia uguale a quello di creazione del progetto, la validazione verrà posticipata al giorno prescelto.

Oltre alla posticipazione della validazione, l'utente può scegliere se eseguire la validazione ad intervalli regolari di 7, 14, 28 giorni oppure eseguirla soltanto una volta.

A tal proposito è stato realizzato lo *Scheduler* presentato nella figura 23.

Lo Scheduler è composto da diversi oggetti Java:

- *schedulation*, un oggetto Java dell'interfaccia *ScheduledFuture*[72], che è generato dalla variabile *executor*, un oggetto di tipo *ScheduledThreadPollExecutor*[73] che permette di avviare delle istruzioni dopo un iniziale periodo di ritardo e/o di eseguirle periodicamente. L'oggetto *schedulation* ha come proprietà l'operazione da eseguire (il thread da avviare relativo all'audit), il periodo di delay e la frequenza con cui l'operazione debba essere eseguita. Il delay e la frequenza sono espressi in giorni;

[71] https://it.wikipedia.org/wiki/AJAX
[72] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledFuture.html
[73] https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledThreadPoolExecutor.html

- *schedulatorManager*, è una *HashMap*[74] che permette di raccogliere tutti gli oggetti di tipo *schedulation*, identificando ogni schedulazione con l'identificativo univoco del progetto a cui appartiene l'audit da avviare.

Dunque, i dati necessari per realizzare la schedulazione di una validazione sono:

- l'eventuale *delay*, ossia la distanza di tempo, espressa in giorni, che intercorre dal giorno della creazione del progetto e il giorno scelto dall'utente per far eseguire la validazione;
- l'eventuale *frequenza*, espressa in giorni, che segna la cadenza tra una validazione e l'altra.

La *frequenza* è espressa direttamente dall'utente nella maschera di creazione del progetto. Qui, l'utente può scegliere se eseguire la validazione ogni 7,14, 28 giorni oppure soltanto una volta. Questo dato viene raccolto insieme agli altri relativi al progetto e viene inviato alla servlet *Project.java*.

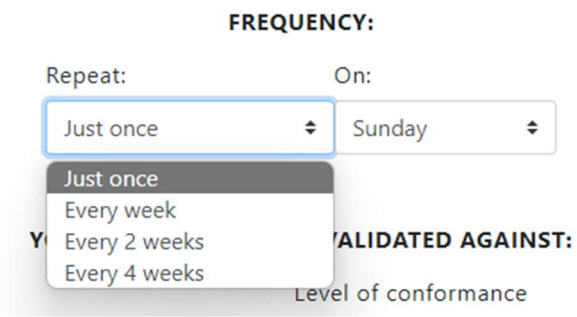


FIGURA 25 SCELTA DELLA FREQUENZA CON CUI ESEGUIRE LA VALIDAZIONE DEL PROGETTO

Il valore di *delay* invece, ha bisogno di alcuni calcoli affinché sia elaborato. L'utente, infatti, nella maschera di creazione del progetto esprime la sua preferenza circa il giorno in cui eseguire la validazione. La lista delle opzioni è composta dai sette giorni della settimana, a ciascuna opzione è associato un valore numerico crescente, che partendo da 1 equivalente al lunedì, può assumere il valore massimo di 7 quando la selezione ricade sul giorno domenica.

[74] https://docs.oracle.com/javase/8/docs/api/java/util/HashMap.html

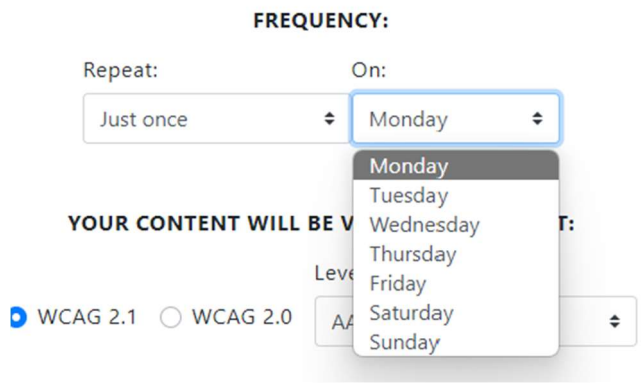


FIGURA 26 SCELTA DEL GIORNO IN CUI ESEGUIRE LA VALIDAZIONE

Per calcolare la differenza tra il giorno scelto dall'utente e il giorno di creazione del progetto, è stato sviluppato un semplice algoritmo:

```
if (today > day) {
        delay = (7 – today) + day;
} else {
        delay = day – today;
}
```

La variabile *today* rappresenta il giorno di creazione del progetto, può assumere valori compresi tra 1 e 7, corrispondenti ai giorni della settimana, con inizio al lunedì.

La variabile *day* rappresenta il giorno scelto dall'utente per l'avvio della validazione, anch'essa può assumere i valori compresi tra 1 e 7 corrispondenti ai giorni della settimana con inizio al lunedì.

Il *delay* rappresenta il numero di giorni che dovranno intercorrere tra la creazione del progetto e la prima validazione, può assumere valori compresi tra 0 e 6.

Questi due valori, *delay* e *frequenza*, vengono utilizzati dall'oggetto *schedulation* ed inseriti nello *schedulatorManager* come visto in precedenza.

La classe *EvalAudit.java* è designata a gestire l'avvio della validazione vera e propria attraverso il recupero delle informazioni del progetto dal database e l'invio

degli stessi al motore di validazione di MAUVE++. Inoltre, al termine della validazione, la classe *EvalAudit.java* ha il compito di salvare i risultati della validazione all'interno del database.

La classe *Audit.java* è stata realizzata per garantire un'ulteriore possibilità all'utente: qualora l'utente voglia avviare manualmente una validazione, perché non è stata impostata una frequenza di validazione automatica oppure per eseguire una validazione prima del termine che è stato calendarizzato, la classe *Audit.java* gestisce l'input dell'utente e crea un nuovo audit che viene immediatamente validato.

3.4.4 Implementazione della nuova funzionalità: il frontend

Dopo aver sviluppato il backend delle nuove funzionalità, si è rivolta l'attenzione alla progettazione e all'implementazione dell'interfaccia di MAUVE++ che potesse fornire all'utenza la possibilità di utilizzare le nuove funzioni.

Utilizzando i linguaggi HTML, CSS (anche con il framework Bootstrap[75]), Javascript all'interno di pagine Web realizzate attraverso la tecnologia JSP, è stato creato un nuovo set di pagine:

- user_profile.jsp
- new_project.jsp
- projects.jsp
- single_project.jsp
- audit.jsp

[75] https://getbootstrap.com/

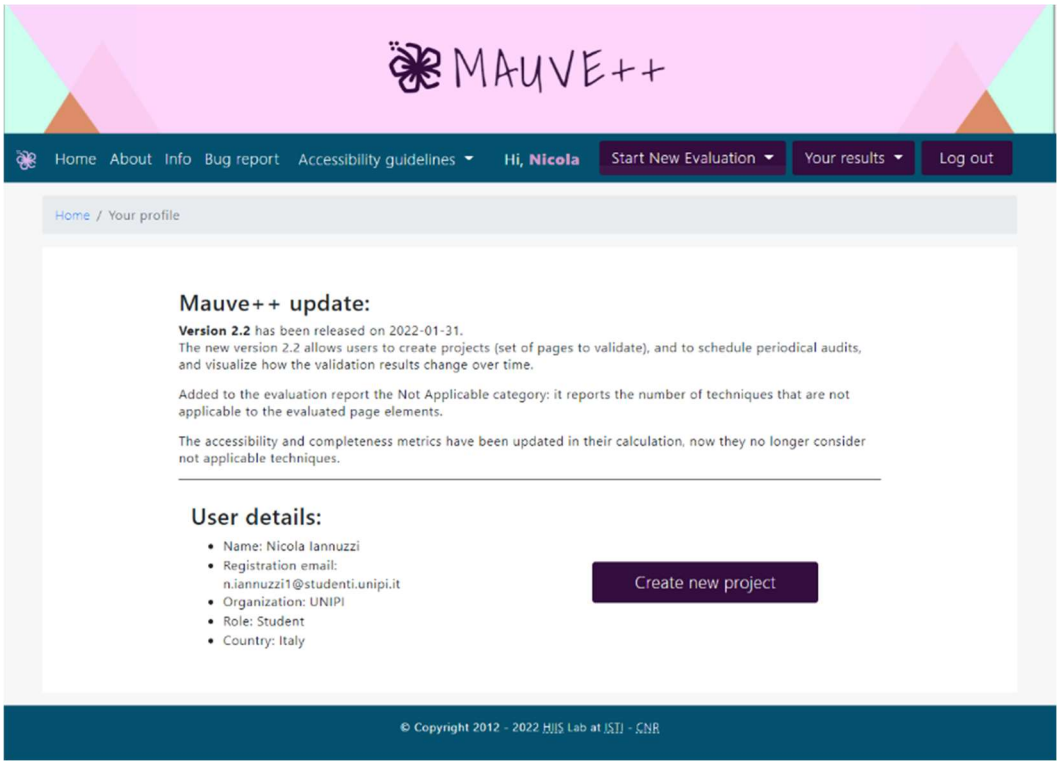


FIGURA 27 PAGINA DEL PROFILO UTENTE

La pagina *user_profile.jsp* è visualizzata subito dopo che l'utente effettua il login all'interno del sistema. Vengono fornite indicazioni sul proprio account, informazioni sugli ultimi aggiornamenti di MAUVE++ e una *call to action* per l'avvio di un nuovo progetto.

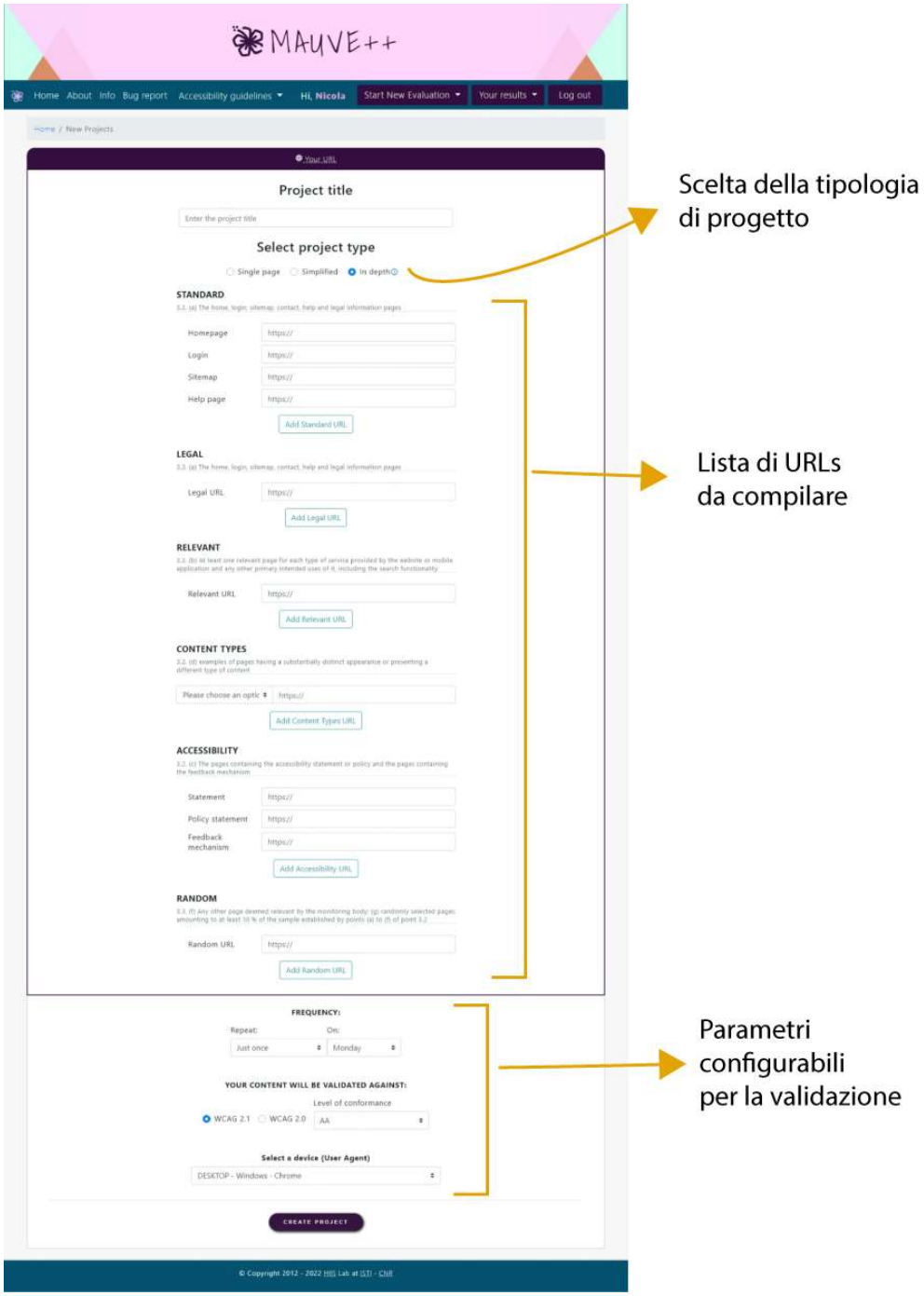


FIGURA 28 MASCHERA DI CREAZIONE DI UN PROGETTO DI TIPO "IN DEPTH"

La pagina *new_project.jsp* ospita il form di creazione del progetto, qui l'utente può configurare i parametri di validazione e scegliere la tipologia del progetto. I parametri configurabili sono:

- il numero massimo di pagine e la profondità delle stesse;
- la frequenza di validazione;
- le linee guida e il livello di conformità sulla base delle quali la validazione debba essere eseguita;
- lo user agent con il quale richiedere il codice HTML e CSS delle pagine da analizzare.

Le tipologie di progetto sono tre: *Single page*, *Simplified*, *In depth.*

Il "Single page project" è il più semplice dei progetti, prevede infatti che vi sia un'unica pagina Web sulla quale avviare la valutazione dell'accessibilità. Anche se MAUVE++ prevedeva già la possibilità di analizzare una singola risorsa Web senza la creazione di un progetto, il "Single page project" permette di monitorare l'andamento dei risultati di accessibilità nel tempo sia automaticamente, specificando una frequenza di validazione, sia manualmente avviando l'analisi ogni qual volta l'utente ne abbia necessità.

Il progetto di tipo "Simplified" prevede l'analisi di un set di pagine scelte casualmente dal *crawler* partendo da una URL specificata dall'utente.

L' "In depth project" è il livello più articolato di un progetto; infatti, viene proposta all'utente una lista di URL significative da compilare in accordo con le specifiche tecniche dell'articolo 6 della direttiva EU 2016/2012.

L'interattività del form di creazione del progetto è sviluppata attraverso Javascript, così come l'invio dei dati al server.

Dopo la compilazione dei campi e il click sul bottone "Create project", se non vengono riscontrati errori di compilazione (ad esempio URL mal formate, oppure campi obbligatori non compilati), un popup avviserà l'utente della riuscita creazione del progetto e lo inviterà, attraverso un link, a visitare la pagina che contiene le informazioni su tutti i progetti appartenenti all'utente.

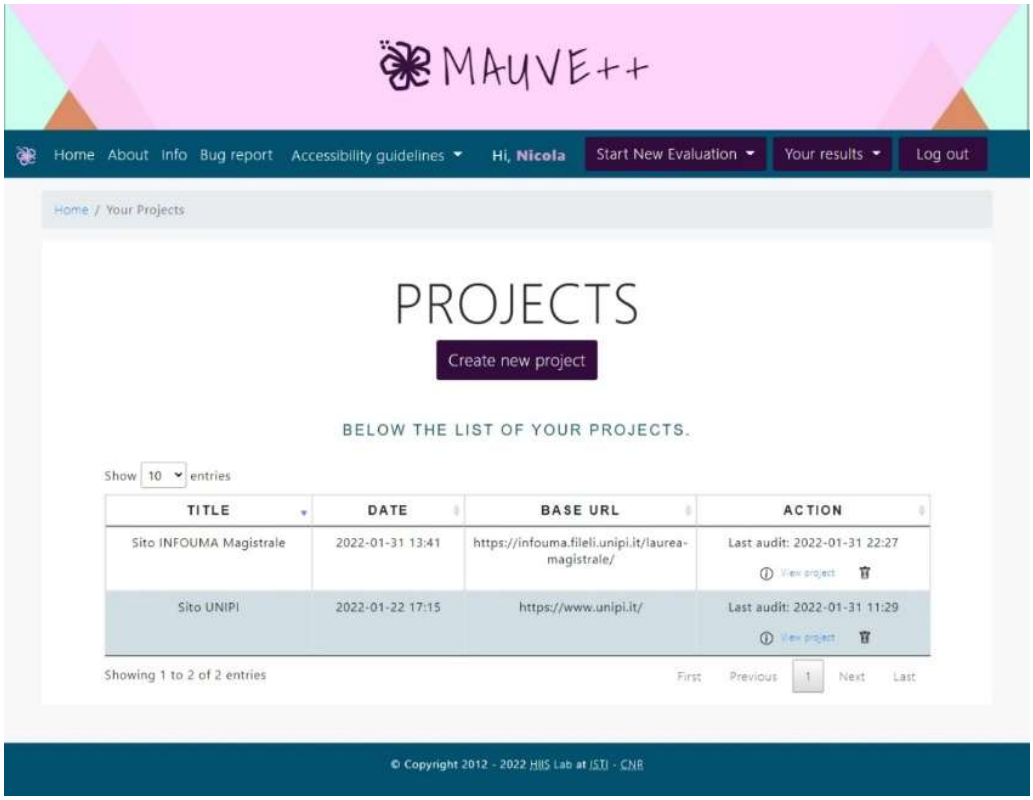


FIGURA 29 PAGINA DEI PROGETTI DELL'UTENTE

3.4.4.1 Lista dei progetti

Nella pagina *projects.jsp*, oltre alle informazioni circa il titolo, la data di creazione e la base URL, l'utente potrà visualizzare lo stato del progetto. Se il progetto prevede una frequenza di validazione automatica, nella colonna "Action" della tabella presentata in figura 29, l'utente vedrà la data relativa all'ultimo audit eseguito e la data futura di quello calendarizzato.

Sempre grazie agli elementi presenti all'interno della colonna "Action", per ogni progetto l'utente potrà effettuarne l'eliminazione oppure visualizzarne i dettagli.

Cliccando sul link "View project", l'utente verrà reindirizzato nella pagina relativa al singolo progetto per visualizzarne i dettagli e i dati associati agli audit eseguiti.

3.4.4.2 Pagina del singolo progetto

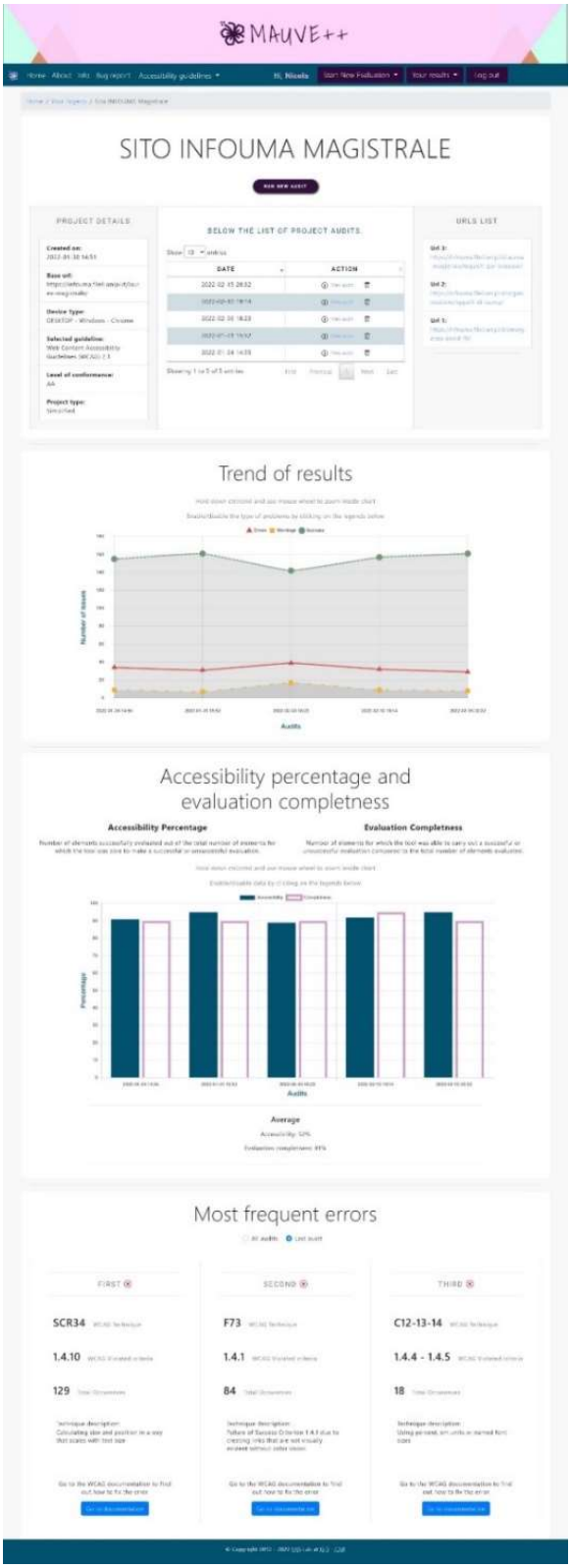


FIGURA 30 PAGINA RELATIVA AD UN SINGOLO PROGETTO

La pagina *single_project.jsp* è composta da quattro sezioni:

- Dettagli del progetto, lista degli audit eseguiti, lista delle URLs;
- “Trend of results”;
- “Accessibility percentage and evaluation completness”;
- “Most frequent errors”.

FIGURA 31 SEZIONE DEDICATA ALLE INFORMAZIONI DEL PROGETTO

Nella sezione dedicata ai dettagli del progetto vengono presentate tre colonne:

1. “Project details”, dove vengono presentate le informazioni del progetto come, ad esempio, le linee guida prescelte per la validazione, lo user agent e la base url;
2. Una tabella popolata dai vari audit eseguiti, dove per ogni audit, è possibile trovare la data di esecuzione e le azioni possibili su di esso (cancellazione e visione nel dettaglio). I dati della tabella possono essere ordinati sulla base

della cronologia di esecuzione degli audit, in ordine ascendente o discendente;

3. “URLs list”, dove vengono fornite tutte le URLs che appartengono al progetto.

La sezione “Trend of results” presenta un grafico a linee, dove tre diverse linee mostrano, per ciascun audit presentato sull’asse delle ascisse, il numero degli elementi, i cui valori sono riportati sull’asse delle ordinate, che sono stati valutati come errori, warnings o superati. Tutti i grafici presenti nelle pagine Web di MAUVE++ sono stati sviluppati utilizzando *Chart.js*[76], una libreria open-source che sfrutta l’elemento *canvas* di HTML5.

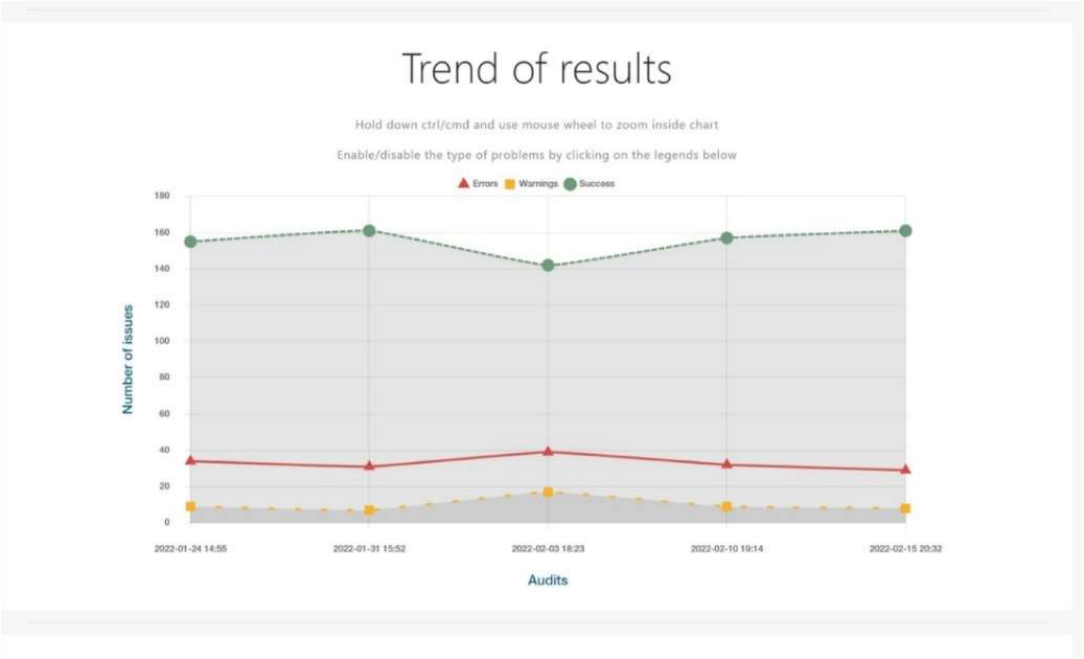


FIGURA 32 GRAFICO "TREND OF RESULTS" PRESENTE NELLA PAGINA DEL SINGOLO PROGETTO

Il grafico permette di interagire con esso attraverso l’esclusione dalla visualizzazione di alcune linee, così come ammette lo zooming in una determinata sezione in modo tale da visualizzare soltanto un certo numero di audit.

[76] https://www.chartjs.org/

La forma dei punti e delle linee del grafico sono state pensate affinché siano distinguibili anche agli utenti che soffrono di una difficoltà visiva, ossia non abbiano la possibilità di distinguere i colori: sono stati usati forme diverse per i punti (cerchio, triangolo, quadrato) e un diverso tratteggio (più o meno distanziato).

Questo grafico è utile a comprendere l'andamento generale dei risultati di accessibilità nel tempo, con riferimento alle tre categorie di problematiche previste da MAUVE++ (errori, warnings, success).

Per popolare il grafico è stato creato un array di oggetti, *jsonDataAudit*, dove vengono raggruppati i dati di ogni singolo audit. Ogni oggetto è formato dalle seguenti proprietà:

- *accPercentage*, dove viene registrata la percentuale di accessibilità;
- *compPercentage*, che rappresenta la completezza dell'analisi (questa metrica e quella precedente, accPercentage, verranno spiegate più avanti);
- *date*, ossia la data dell'audit;
- *errors*, numero di errori;
- *warnings*, numero di warnings;
- *success*, numero di controlli superati.


```
[
    {
        "date": "2022-01-24 14:55",
        "errors": 34,
        "warnings": 9,
        "success": 155,
        "accPercentage": 91,
        "compPercentage": 90
    },
    {
        "date": "2022-01-31 15:52",
        "errors": 31,
        "warnings": 7,
        "success": 161,
        "accPercentage": 95,
        "compPercentage": 90
    },
    {
        "date": "2022-02-03 18:23",
        "errors": 39,
        "warnings": 17,
        "success": 142,
        "accPercentage": 89,
        "compPercentage": 90
    }
]
```


FIGURA 33 STRUTTURA ARRAY CON DATI DI ESEMPIO PER LA CREAZIONE DEI GRAFICI

Utilizzando il metodo *map*[77] sull'array, vengono creati quattro nuovi array, *totErrorsAudits*, *totWarningsAudits*, *totSuccessAudits* e *labels* che raccolgono i dati rispettivamente degli errori, dei warnings, dei success e delle date di tutti gli audit.

```
let labels = jsonDataAudit.map(b => b.date);
let totErrorsAudits = jsonDataAudit.map(b => b.errors);
let totWarningsAudits = jsonDataAudit.map(b => b.warnings);
let totSuccessAudits = jsonDataAudit.map(b => b.success);
```

FIGURA 34 CREAZIONE DEGLI ARRAY PER L'UTILIZZO NEL GRAFICO "TREND OF RESULT"

Le due metriche riassuntive di MAUVE++ viste nel capitolo 2, "Accessibily percentage" e "Evaluation completeness", vengono visualizzate nella sezione "Accessibility percentage and evaluation completeness" in un grafico a barre, dove sull'asse delle ascisse vengono visualizzati i singoli audit, mentre sull'asse delle ordinate il valore in percentuale delle due metriche.

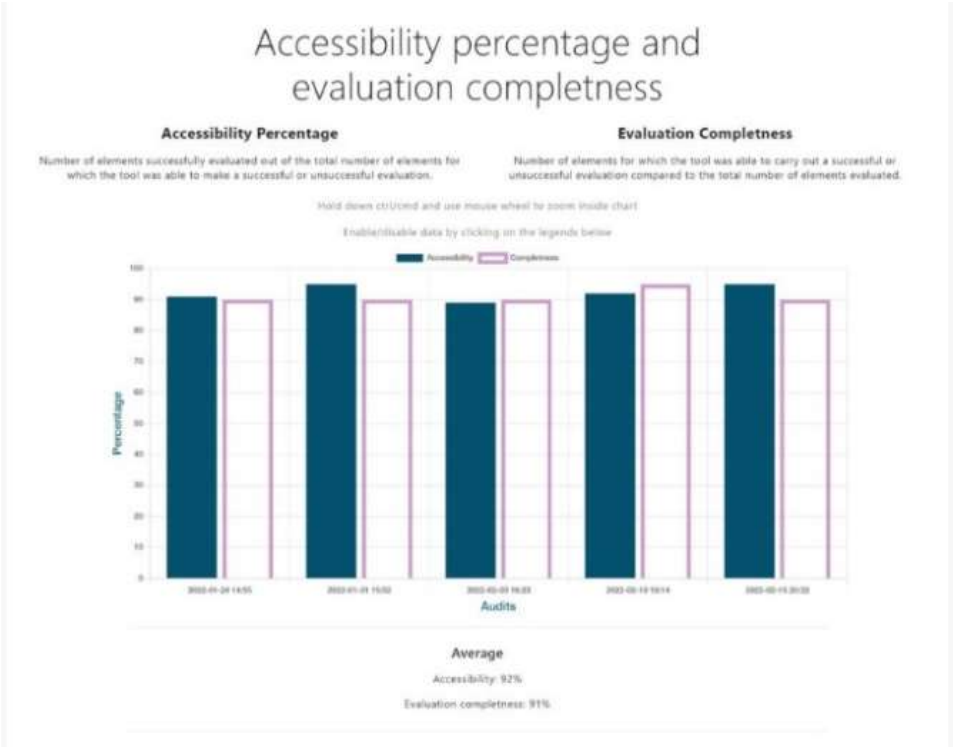


FIGURA 35 GRAFICO SULLE METRICHE DI ACCESSIBILITÀ E COMPLETEZZA DELL'ANALISI

[77] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map?retiredLocale=it

Anche questo grafico permette di interagire con i dati mostrati, escludendo una delle due metriche dalla visualizzazione, oppure zoomando in una porzione specifica del grafico.

L'ultima sezione presente nella pagina del progetto mostra i tre errori più frequenti riscontrati in tutti gli audit o soltanto nell'ultimo audit eseguito in ordine temporale. Nelle tre colonne della sezione, vengono descritti gli errori indicando la tecnica violata, il criterio WCAG a cui la tecnica appartiene, il numero di occorrenze, una breve descrizione dell'errore e un link che rimanda alla documentazione WCAG per la soluzione al problema di accessibilità sollevato.

L'utente ha la possibilità di visualizzare i tre errori più frequenti relativi all'ultimo audit oppure in relazione a tutti gli audit eseguiti.

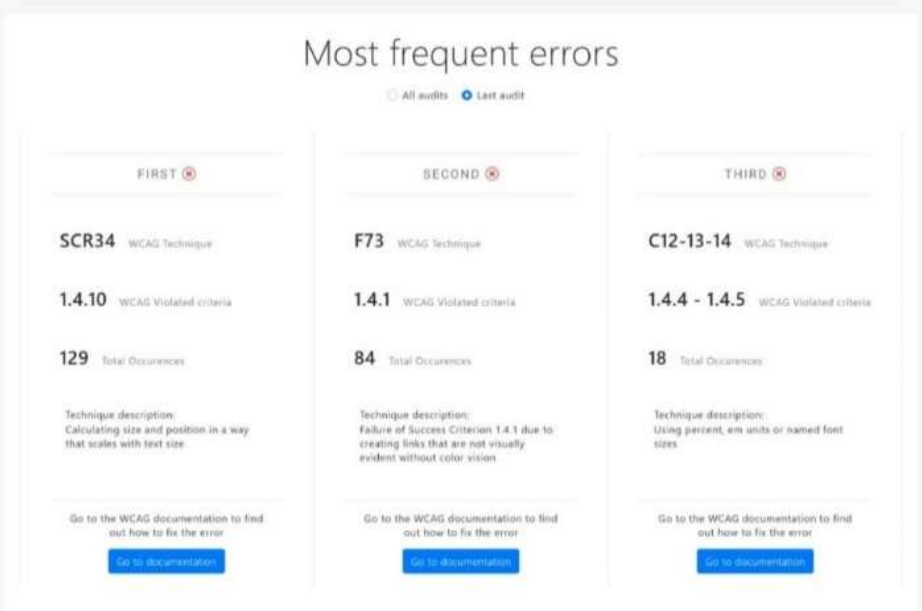


FIGURA 36 SEZIONE DEGLI ERRORI PIÙ FREQUENTI RISCONTRATI NEL PROGETTO

3.4.4.3 Pagina del singolo audit

Selezionando un audit dalla lista presente nella pagina del progetto, è possibile analizzare nel dettaglio i risultati della validazione.

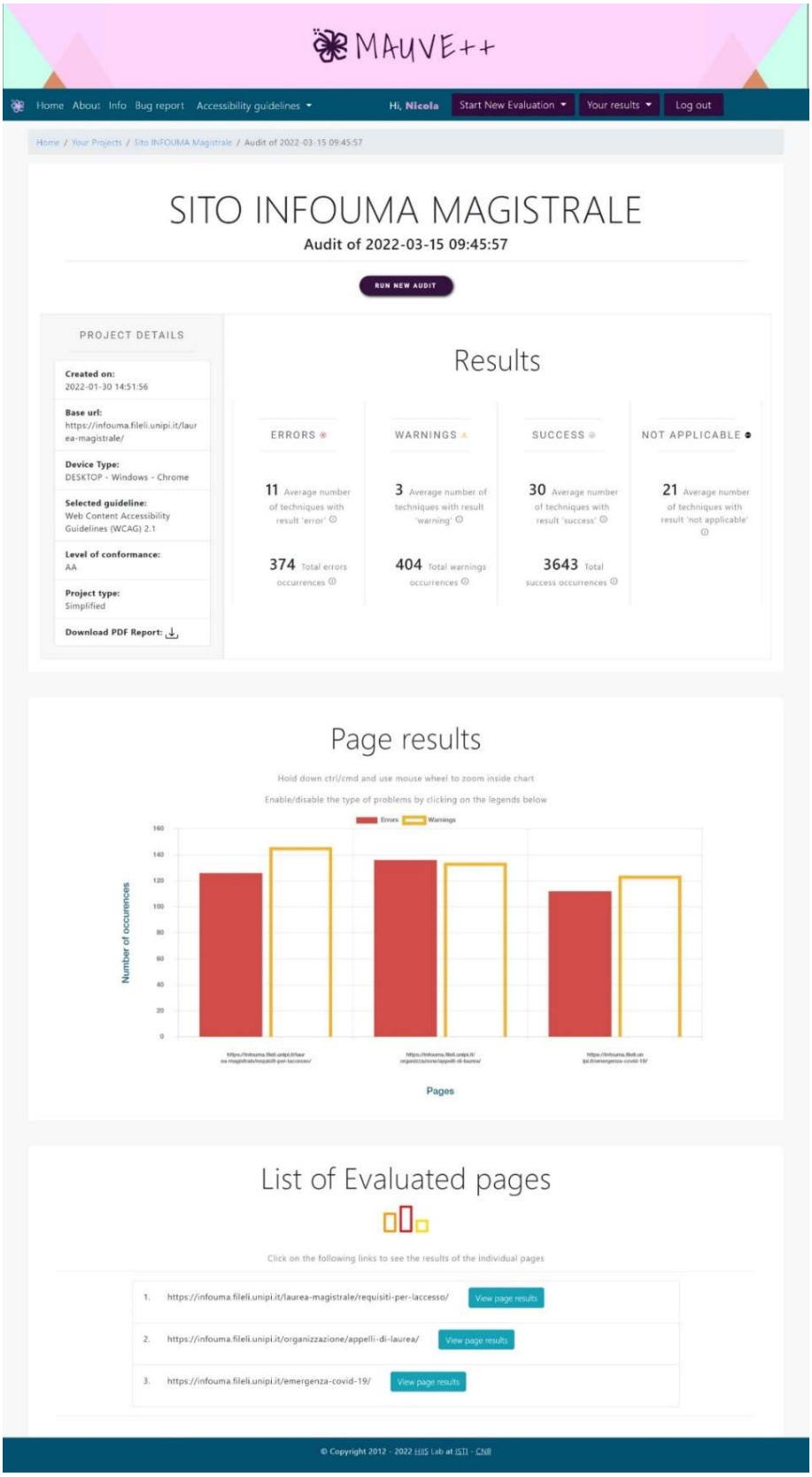


FIGURA 37 PAGINA RELATIVA AD UNA SINGOLA VALUTAZIONE

La pagina dedicata ad un singolo audit è sviluppata in tre sezioni (figura 37):

- La prima sezione è composta da due colonne, la prima delle quali riporta un riepilogo dei dati del progetto offrendo la possibilità di scaricare il report in formato PDF dell'analisi relativa; la seconda colonna è occupata dai dati riguardanti il numero di errori, warnings, success e not applicable. Questi ultimi dati riportano i valori medi delle problematiche rilevate nella totalità delle pagine analizzate che compongono l'audit;
- La seconda sezione, *Page results*, prevede un grafico a barre dove, sull'asse delle ascisse sono organizzate le singole pagine che compongono l'audit, e sull'asse delle ordinate i valori rilevati di errors e warnings.

FIGURA 38 GRAFICO PROBLEMATICHE/PAGINE RELATIVE AD UN SINGOLO AUDIT

- La terza sezione presente nella pagina del singolo audit, si compone della lista delle URLs analizzate, per ognuna delle quali viene fornito un link per visualizzare in dettaglio i risultati dell'analisi sulla singola pagina.

4 Test di usabilità: i partecipanti, il test e i risultati

Le nuove funzionalità descritte nel capitolo precedente sono già disponibili online sul sito di MAUVE++[78]. Il servizio "Server side rendering" e la visualizzazione del report "Live preview" sono stati attivati nel corso del mese di giugno 2021, mentre la funzionalità relativa al monitoraggio è disponibile da gennaio 2022.

Per ricevere un feedback diretto sull'usabilità dello strumento, viste le nuove funzionalità introdotte, è stato condotto un test di usabilità, nel quale gli utenti hanno avuto modo di compiere alcune operazioni sull'interfaccia e fornire le loro impressioni in un questionario.

4.1 I partecipanti

Il numero di partecipanti al test è stato di quindici persone, con un'età media di circa 41 anni. L'invito al test è stato diretto a persone con competenze nel mondo del Web, dagli sviluppatori agli addetti alla creazione di contenuti.

Le risposte circa il profilo professionale ricoperto sono state (i partecipanti potevano esprimere più di una qualifica professionale):

PROFILO	NUMERO
Web commissioner	5
Web developer	11
Accessibility expert	2
Ricercatore	1
Dottoranda in Human Centered Artificial Intelligence	1
UI designer	1

Alla domanda relativa ad un precedente utilizzo di un qualunque strumento di validazione automatica dell'accessibilità Web, gli utenti hanno risposto

[78] https://mauve.isti.cnr.it/

affermativamente in 8 casi e negativamente nei restanti 7 casi. A coloro i quali hanno dichiarato di aver utilizzato strumenti di validazione dell'accessibilità precedentemente al test in questione, è stato chiesto di citare gli strumenti utilizzati: *Site improve* e *Wave* sono stati segnalati da 3 utenti, *MAUVE++* e *altri strumenti del W3C* da 2 utenti, *Google Lighthouse* da 1 utente.

Il test è stato condotto da remoto attraverso la piattaforma *Skype*, le sessioni sono state registrate con il consenso scritto fornito dei partecipanti prima dell'inizio della sessione.

Agli utenti del test è stato fornito un breve documento informativo per introdurre gli aspetti generali delle linee guida WCAG e alcune informazioni sullo strumento che avrebbero utilizzato.

Al termine della lettura della documentazione fornita, l'utente ha ricevuto il link allo strumento, le credenziali di accesso opportunamente create per il test e un ulteriore link al questionario da compilare contemporaneamente allo svolgimento dei task proposti.

È stato chiesto agli utenti di condividere lo schermo dove fosse presente l'interfaccia dello strumento per seguire meglio l'andamento del test e, per prevenire qualsiasi tipo di influenza sulle risposte date nel questionario, è stato chiesto all'utente di non condividere la finestra dove fossero visualizzate le domande.

La sessione di ciascun test ha avuto una durata compresa tra 30 minuti e 60 minuti.

4.2 I tasks e il questionario

4.2.1 Task 1: "Single Page Evaluation – Analisi delle viste"

Il task 1 ha riguardato la "Single Page Evaluation" e l'analisi delle varie viualizzazioni proposte dallo strumento. Si è chiesto agli utenti di avviare la validazione di una singola pagina impostando i seguenti parametri:

- URL da analizzare: https://it.wikipedia.org/wiki/Pagina_principale;

- Linee guida: WCAG 2.1;
- Livello di conformità: AAA;
- Richiesta HTML attraverso il servizio di server side rendering;
- User agent: Windows – Desktop – Edge.

Una volta terminata l'esecuzione della validazione e aver preso visione delle varie viste proposte dallo strumento per analizzare i risultati, è stato chiesto agli utenti di rispondere alle seguenti domande:

- Analizzando i risultati, quanti tipi di problemi, diversi tra loro, di categoria "ERROR", ha identificato il tool?
- All'interno dei problemi di categoria ERROR, quale tipo di problema ricorre più frequentemente? Quante volte ricorre?
- Ha incontrato qualche difficoltà a comprendere una o più informazioni incluse nelle varie viste ("Evaluation Summary", "End User", "Live preview" e "Web developer")? Se sì, cosa in particolare ha trovato poco comprensibile o chiaro, e perché?

4.2.2 Task 2: "Analisi della Live Preview"

Sfruttando l'analisi condotta nel precedente task, è stato chiesto agli utenti di prendere visione della sezione "Live Preview" e di impostare i seguenti filtri sui risultati:

- Tipo di problematica: "Errors"
- Tipo di codice: "CSS"
- Principio WCAG: "PERCEIVABLE"

Dopo aver filtrato i risultati, gli utenti hanno avuto il compito di individuare all'interno della lista degli errori, l'errore relativo alla violazione del criterio 1.4.10, tecnica SCR34 e di analizzarne le prime 5 occorrenze. Proporre questo task ha

permesso di far valutare il comportamento dello strumento quando un'occorrenza dell'errore fosse visibile o meno nella pagina riprodotta.

Nelle domande del questionario relative al task 2, utilizzando una scala Likert 1-5 (1=fortemente in disaccordo, 5=fortemente d'accordo), è stato chiesto agli utenti di esprimere il loro giudizio rispetto a queste affermazioni:

- Nei casi in cui è stato possibile visualizzare l'errore all'interno della pagina Web, la visualizzazione offerta all'utente è utile a localizzare/comprendere il problema identificato.
- Nei casi in cui il tool è stato in grado di evidenziare l'errore all'interno della pagina Web, MAUVE++ fornisce anche alcune informazioni per risolvere l'errore (passando il cursore sopra l'elemento evidenziato). La visualizzazione offerta all'utente è utile per la risoluzione del problema identificato.
- Nei casi in cui l'occorrenza dell'errore non è visibile all'interno della pagina, il tool rimanda alla vista "Web developer", per l'analisi del codice sorgente. È utile poter analizzare l'errore direttamente nel codice sorgente.

È stata proposta un'ulteriore domanda a risposta aperta per raccogliere le eventuali indicazioni personali degli utenti:

- Avrebbe suggerimenti su come migliorare la Live Preview, o cosa modificare in essa?

4.2.3 Task 3: "Creazione di un progetto"

Nel terzo task proposto agli utenti è stato chiesto di creare un nuovo progetto di tipo "In-depth" con i seguenti parametri:

- URLs:
    - STANDARD:
        - Homepage: https://www.unipi.it/

        - Login: https://unipi.idp.cineca.it/idp/profile/SAML2/Redirect/SSO?execution=e2s1
    - CONTENT TYPES:
        - Forms: https://unimap.unipi.it/cercapersone/cercapersone.php
    - ACCESSIBILITY:
        - Statement: https://www.unipi.it/index.php/documenti-ateneo/item/14764

- PARAMETRI:
    - Linee guida: WCAG 2.1
    - Livello di conformità: AA
    - Repeat: Just Once
    - On: selezionare il giorno in cui si sta eseguendo il test
    - User agent: Desktop – Windows – Chrome

Dopo aver creato il progetto e aver avviato contestualmente un nuovo audit, l'utente ha risposto alla domanda con risposta aperta:

- Ha incontrato particolari difficoltà nella creazione di un progetto o nella comprensione dei vari parametri da specificare? Se sì, indicare dove.

4.2.4 Task 4: "Analisi di un audit"

In relazione al progetto creato nel task precedente, è stato chiesto all'utente di entrare nel dettaglio dell'audit avviato in precedenza e di prendere visione delle sezioni "Results" e "Page results" presentate nella pagina dell'audit.

Attraverso le informazioni presenti nelle sezioni prese in considerazione, è stato chiesto all'utente di individuare la pagina che presenta il maggior numero di errori.

Al termine del task, l'utente ha valutato il proprio accordo con la seguente affermazione (utilizzando una scala Likert 1-5, dove 1=fortemente in disaccordo, 5=fortemente d'accordo):

- Le visualizzazioni presentate dal tool e associate ad un audit sono chiare e comprensibili.

A seguire, è stata proposta un'ulteriore domanda a risposta aperta:

- Avrebbe suggerimenti su cosa modificare o come migliorare le informazioni associate ad ogni audit?

4.2.5 Task 5: "Analisi di un progetto (già creato su dati simulati)"

Per valutare la nuova funzionalità di monitoraggio dell'accessibilità di un sito Web, è stato creato un progetto su MAUVE++ con alcuni audit in cui sono stati simulati i dati così da mostrare una certa variazione nei risultati. Il progetto è stato associato ad un account utente generico, del quale sono state fornite le credenziali di accesso alla piattaforma a tutti gli utenti del test.

In questo task, è stato chiesto agli utenti di aprire la pagina del progetto appena descritto ed esplorare le varie sezioni della pagina ("Trend of Results", "Accessibility percentage and evaluation completeness", "Most frequent errors"). Inoltre, dopo aver interagito con i grafici presenti nella pagina stessa, l'utente ha risposto alle seguenti domande:

- In quale audit l'Evaluation Completeness è risultata migliore?
- In quale audit si è avuto il minor numero di successi?
- Qual è il tipo di errore più frequente riscontrato nell'ultimo audit?
- Le visualizzazioni presentate nel tool ed associate ad un progetto sono chiare e comprensibili?
- Se ha incontrato qualche difficoltà a comprendere una o più informazioni incluse nelle varie visualizzazioni associate al progetto, oppure ad interagire con uno dei grafici visualizzati, potrebbe specificare in che situazione?

- Ha suggerimenti su cosa si potrebbe modificare per migliorare le informazioni associate ad ogni progetto?

Al termine dei task, il questionario ha previsto un'ulteriore sezione con domande a risposta aperta dove gli utenti hanno potuto esprimere la loro esperienza complessiva di utilizzo di MAUVE++. L'utente ha potuto fornire il proprio personale parere come risposta alle seguenti domande:

- Quali sono le tre cose che Le sono piaciute di più del tool?
- Quali sono le tre cose che Le sono piaciute di meno in MAUVE++?
- In generale, Le sembra che MAUVE++ dia in modo chiaro tutte le informazioni necessarie per avere un quadro dei problemi di accessibilità di un sito, e poterne monitorare l'accessibilità nel tempo?
- C'è qualche informazione che Lei pensa sia utile e che invece non ha trovato in MAUVE++?
- Ha ulteriori suggerimenti per migliorare il tool?

4.3 I risultati

In relazione alla prima domanda del task 1 nella quale è stato chiesto di indicare il numero di problemi di categoria "Error" che il tool ha rilevato durante l'analisi, tredici utenti su quindici hanno risposto correttamente. Successivamente, alla domanda in cui veniva chiesto all'utente di identificare, all'interno della lista degli errori rilevati, l'errore che presentava il maggior numero di occorrenze, undici utenti sul totale di quindici hanno risposto correttamente. Rispetto alle risposte incorrette fornite a questa domanda, durante i test e la revisione delle video registrazioni effettuate, è stato notato che il termine "categoria" usato nella domanda ha creato confusione negli utenti, perché lo stesso termine è utilizzato nell'interfaccia di MAUVE++ per indicare la categorizzazione degli errori sulla base degli elementi HTML ai quali gli errori risultanti appartengono.

La domanda successiva ha chiesto all'utente di specificare il numero di occorrenze per l'errore individuato in precedenza: nove utenti hanno risposto correttamente; dei restanti sei, tre non hanno fornito la risposta richiesta, mentre gli altri tre hanno risposto non correttamente in seguito all'ambiguità della domanda come spiegato poco sopra.

Rispondendo alla domanda "Ha incontrato qualche difficoltà a comprendere una o più informazioni incluse nelle varie viste ("Evaluation Summary", "End User", "Live preview" e "Web developer")? Se sì, cosa in particolare ha trovato poco comprensibile o chiaro, e perché?", dieci utenti hanno dichiarato che le informazioni riportate erano chiare e che non hanno riscontrato alcun tipo di difficoltà. Un utente di questo gruppo ha fornito come consiglio quello di dare maggiore visibilità alla lista degli errori presente sulla sinistra della schermata della "Live preview", aggiungendo l'apprezzamento per la vista "Web developer" dove i diversi tipi di problemi rilevati, errori e warnings, erano facilmente identificabili grazie all'uso dei colori. I cinque utenti che hanno incontrato alcune difficoltà hanno fornito dei commenti, alcuni dei quali sono riportati qui di seguito. Due utenti hanno riportato delle difficoltà nel capire la connessione tra le informazioni presentate nelle varie viste dello strumento; Due utenti hanno suggerito di offrire la possibilità di ordinare la tabella, nella vista "End User" dove sono presentati gli errori, sulla base della colonna dove viene riportato il numero delle occorrenze degli errori. Questo avrebbe garantito una più rapida individuazione dell'errore che presenta più occorrenze, così come richiesto nel task.

Le successive domande prevedano una risposta sulla base del livello d'accordo che l'utente aveva con alcune affermazioni fornite. L'accordo era valutabile su una scala Likert a 5 punti, dove il valore 1 equivaleva al massimo disaccordo e 5 al massimo accordo. Qui di seguito vengono riportati dei grafici riepilogativi relative alle domande proposte:

Nei casi in cui è stato possibile visualizzare l'errore all'interno della pagina web, la visualizzazione offerta all'utente è utile a localizzare/comprendere il problema identificato

15 risposte

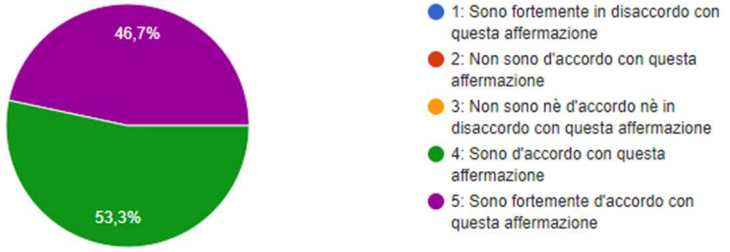


FIGURA 39 MIN=4; MAX=5; MEDIANA=4; MEDIA=4.5

Nei casi in cui il tool è stato in grado di evidenziare l'errore all'interno della pagina web, MAUVE++ fornisce anche alcune informazioni per risolvere l'errore (passando il cursore sopra l'elemento evidenziato). La visualizzazione offerta all'utente è utile per la risoluzione del problema identificato.

15 risposte

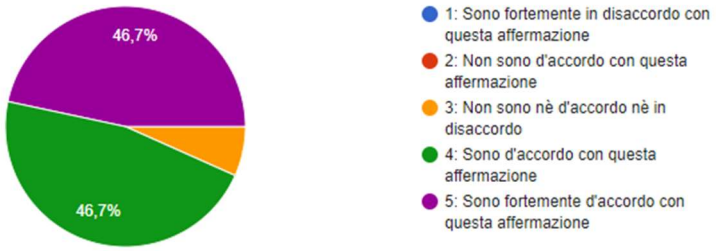


FIGURA 40 MIN=3; MAX=5; MEDIANA=4; MEDIA=4.4

Nei casi in cui l'occorrenza dell'errore NON è VISIBILE all'interno della pagina, il tool rimanda alla vista "Web developer", per l'analisi del codice sorgente. E' utile poter analizzare l'errore direttamente nel codice sorgente.

15 risposte

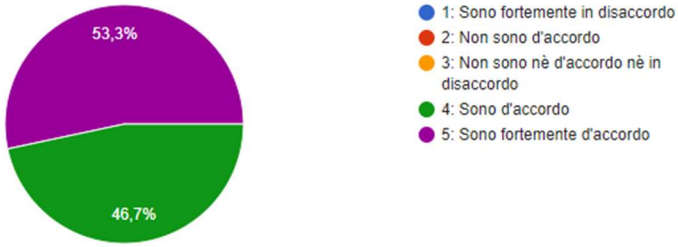


FIGURA 41 MIN=4; MAX=5; MEDIANA=5; MEDIA=4.5

La domanda successiva invitava gli utenti a fornire alcuni suggerimenti su come potesse essere migliorata la Live preview, attraverso modifiche oppure aggiunte all'interfaccia. Sei utenti hanno dichiarato di aver trovato tutto comprensibile e utile, e dunque di non aver nessun consiglio sui possibili miglioramenti da apportare. I restanti nove invece hanno fornito le seguenti indicazioni: tre utenti hanno suggerito alcuni accorgimenti da applicare alla sezione dedicata al filtraggio dei risultati, uno di questi avrebbe preferito che la sezione fosse maggiormente distinta da quella che presenta la lista degli errori, mentre gli altri due hanno dichiarato che sarebbe opportuno fornire un apposito bottone per applicare i filtri (ad oggi i filtri si applicano in automatico ogni qual volta viene cambiato il valore di uno di essi); sei utenti si sono focalizzati sugli aspetti di visualizzazione offerti dall'interfaccia della Live preview:

- un utente ha suggerito di cambiare l'icona a forma di occhio qualora l'errore non fosse visibile nella pagina riprodotta, magari sostituendola con quella di un occhio barrato;
- due utenti hanno dichiarato che l'avviso recante il link alla sezione "Web Developer" quando l'errore non è direttamente visibile nella pagina, fosse poco visibile;
- un utente ha riscontrato che il popup contenente le indicazioni "How to solve?", che appare al passaggio del puntatore sull'elemento evidenziato, una volta chiuso non riappare se ci si sposta nuovamente con il puntatore sopra l'elemento;
- un utente ha dichiarato che l'apertura del popup "How to solve?" legata all'evento on hover è poco intuitiva;
- un utente ha trovato poco evidente l'interattività del tasto a forma di occhio.

Relativamente al task "Creazione di un progetto", la successiva domanda chiedeva agli utenti se avessero avuto particolari difficoltà nella creazione di un progetto o nella comprensione dei vari parametri di configurazione offerti dalla maschera

dell'interfaccia. Dieci utenti hanno dichiarato di non aver incontrato nessun tipo di difficoltà. Cinque utenti avrebbero preferito che le varie sezioni del form di creazione del progetto fossero maggiormente distinte tra loro, indicando anche che sarebbe opportuno fornire un feedback circa i campi da compilare obbligatoriamente e quelli invece opzionali.

Il quesito successivo, relativo alla visualizzazione dei risultati di un singolo audit, prevedeva che gli utenti fornissero il loro grado di accordo (scala Lickert 1-5) sull'affermazione riportata nel grafico che segue:

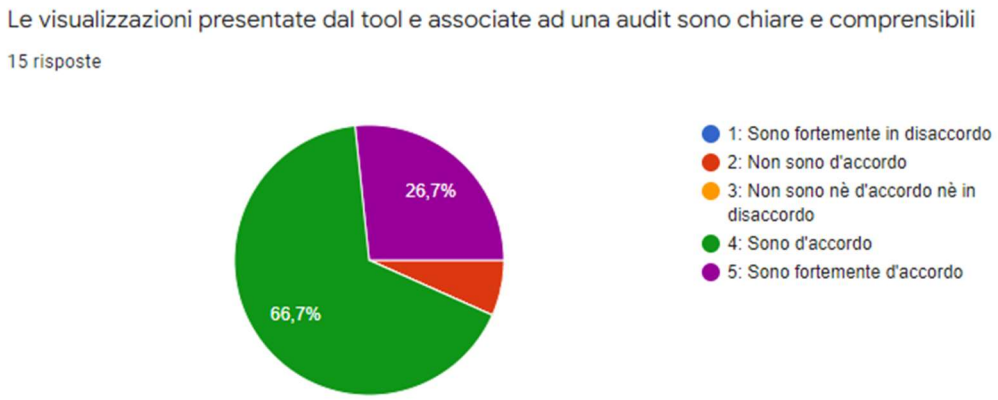


FIGURA 42 MIN=2; MAX=5; MEDIANA=4; MEDIA=4.1

Sempre relativamente alla visualizzazione dei risultati di un singolo audit, gli utenti erano invitati a lasciare alcuni suggerimenti su cosa cambiare o come migliorare le informazioni associate all'audit. Otto utenti hanno dichiarato di non avere particolari suggerimenti a riguardo. Un utente ha suggerito di aggiungere un ulteriore grafico che mostrasse il numero di tecniche violate per ogni pagina. Quattro utenti hanno segnalato che i punti del grafico "Page results" erano troppo piccoli per essere identificati immediatamente, così come i punti più vicini all'asse Y del grafico fossero poco evidenti. Un altro utente ha suggerito di modificare il tipo di grafico "Page results", passando da un grafico a linee ad un istogramma.

Alcuni di questi suggerimenti sono stati accolti e sono già stati modificati alcuni aspetti dello strumento: i punti dei grafici sono stati resi maggiormente visibili,

differenziandoli anche nelle forme assunte, così come il grafico “Page results” è stato modificato in istogramma.

Le prime tre domande del questionario relative al task 5, “Analisi di un progetto”, sono state poste per comprendere se le informazioni che l’interfaccia propone fossero comprensibili agli utenti. Alla domanda “In quale audit l'Evaluation Completeness è risultata migliore?”, quattordici utenti su quindici hanno risposto correttamente. Alla seconda domanda, “In quale audit si è avuto il minor numero di successi?”, tutti e quindici gli utenti hanno risposto correttamente. Alla terza domanda, “Qual è il tipo di errore più frequente riscontrato nell'ultima audit?”, un utente ha risposto non correttamente, i restanti quattordici hanno risposto correttamente.

Il grado di comprensibilità delle informazioni proposte su un progetto, sono state oggetto della successiva domanda in cui veniva chiesto agli utenti di valutare il loro grado di accordo (scala Likert 1-5) all’affermazione presentata nel grafico qui di seguito:

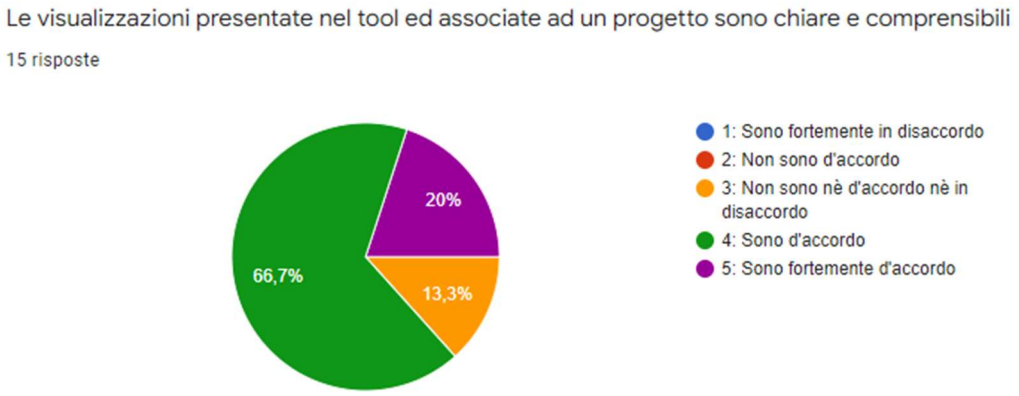


FIGURA 43 MIN=3; MAX=5; MEDIANA=4; MEDIA=4.1

Le successive due domande a risposta aperta concludono i quesiti relativi alle informazioni associate ad un progetto. Alla prima domanda, “Se ha incontrato qualche difficoltà a comprendere una o più informazioni incluse nelle varie visualizzazioni associate al progetto, oppure ad interagire con uno dei grafici visualizzati, potrebbe specificare in che situazione?”, cinque utenti hanno risposto

di non aver incontrato nessuna difficoltà. Tre utenti hanno rilevato alcune difficoltà di comprensione relativamente alla sezione "Most frequent errors", più specificatamente, in un caso, le indicazioni di "First", "Second" e "Third" sono stati interpretati erroneamente come indicazioni sull'audit di riferimento invece che come indicazioni circa il "il primo errore più frequente", "il secondo errore più frequente" e il "terzo errore più frequente". Un utente ha suggerito di specificare che la misura "Evaluation completeness" sia anch'essa una misura percentuale come "Accessibility percentage". Un altro utente ha dichiarato che avrebbe preferito una maggiore interattività dei grafici. Quest'ultima indicazione potrebbe essere oggetto dei futuri sviluppi dello strumento, prevedendo il passaggio dalla libreria Javascript *Chart.js*, attualmente utilizzata per la realizzazione dei grafici, alla libreria *D3.js*[79], che permette una maggiore possibilità nello sviluppo di funzioni interattive sui grafici.

La seconda domanda a risposta aperta relativa al progetto è stata "Ha suggerimenti su cosa si potrebbe modificare per migliorare le informazioni associate ad ogni progetto?". A questa domanda nove utenti hanno risposto affermando di non avere particolari suggerimenti a riguardo. Un utente ha evidenziato come le descrizioni delle tecniche WCAG non siano uniformi (alcune volte viene descritto l'errore, altre volte vengono proposte soluzioni operative per la risoluzione dell'errore). Un altro utente ha suggerito di mostrare due differenti grafici per le metriche "Accessibility percentage" e "Evaluation completeness", invece di un unico grafico con la rappresentazione delle due metriche. In riferimento a questa ultima segnalazione, si crede sia opportuno puntualizzare che è comunque possibile nascondere una delle due metriche visualizzate dal grafico.

Terminati i quesiti relativi ai task proposti, il questionario ha previsto una serie di domande a risposta aperta per valutare l'esperienza complessiva dell'utilizzo dello strumento.

Alla domanda "Quali sono le tre cose che Le sono piaciute di più del tool?", quattro utenti hanno menzionato la chiarezza delle informazioni presentate e il loro layout.

[79] https://d3js.org/

Sempre in quattro occasioni, è stata citata la possibilità di procedere a multiple audit così da poterne visualizzare la progressione dei risultati nel corso del tempo. A tre utenti è piaciuta la possibilità di visualizzare gli errori sia direttamente sulla pagina che sul codice sorgente, così come tre utenti hanno espresso il loro apprezzamento per lo stile grafico dell'interfaccia. In tre occasioni gli utenti hanno menzionato la chiarezza dell'interfaccia dello strumento, altri due utenti hanno apprezzato la sua intuitività, così come altri due utenti hanno manifestato il loro apprezzamento per l'oggettività dei risultati e la loro puntuale esposizione. A due utenti è piaciuta la velocità dell'esecuzione delle validazioni, così come altri due utenti hanno manifestato il loro compiacimento per la connessione puntuale alla documentazione che permette di ricevere indicazioni circa la risoluzione del problema. Altri apprezzamenti sono stati rivolti ai seguenti aspetti: la completezza delle informazioni, la possibilità di valutare una singola pagina o un intero sito Web, la Live preview così come la vista Web Developer, la versatilità dello strumento, la sua interattività, la capacità responsive dell'interfaccia, la possibilità di portare a compimento una valutazione di accessibilità anche da parte di utenti "meno esperti", la chiarezza del flusso di creazione di un progetto che l'interfaccia garantisce.

La domanda successiva chiedeva agli utenti di esprimere le tre cose dello strumento che sono piaciute meno. Quattro utenti hanno dichiarato di non riuscire a trovare un aspetto che non hanno apprezzato. Tre utenti hanno focalizzato l'attenzione sui grafici presenti nella pagina del progetto e del singolo audit: un utente ha dichiarato che avrebbe preferito un altro metodo per nascondere alcune informazioni dal grafico invece di quello previsto, ossia il click sulle legende dello stesso; un altro utente ha trovato le legende dei grafici troppo piccole rispetto al grafico stesso; un utente ha suggerito di aumentare il contrasto delle etichette delle legende affinché siano più visibili. Due utenti hanno riferito che alcune funzionalità dello strumento non sono immediatamente visibili: i link di connessione alla vista Web developer presenti nella vista Live preview non sono abbastanza evidenziati, così come i link presenti nelle tabelle riepilogative dei progetti e degli audit. Due utenti avrebbero voluto trovare più informazioni, nelle pagine dello strumento, che

avrebbero chiarito maggiormente alcuni aspetti degli elementi visualizzati. Altre indicazioni hanno riguardato: l'eccessiva spaziatura dell'interfaccia nel presentare le informazioni; la difficoltà nel leggere il codice sorgente in relazione con gli errori evidenziati; l'assenza della possibilità di approfondire ancora più nel dettaglio le informazioni presentate. Un'indicazione di carattere concettuale: nella visualizzazione "End user", quando viene selezionata la categorizzazione degli errori secondo i princìpi WCAG, un utente ha segnalato che sarebbe opportuno presentare una gerarchia tra gli errori basata sulla gravità degli stessi (l'utente in questione ha esemplificato dicendo che un solo errore appartenente al principio "Operable" potrebbe essere più grave rispetto a molti errori "Perceivable").

Tutti i partecipanti al test hanno risposto affermativamente alla successiva domanda "In generale, Le sembra che MAUVE++ dia in modo chiaro tutte le informazioni necessarie per avere un quadro dei problemi di accessibilità di un sito, e poterne monitorare l'accessibilità nel tempo?". Sei utenti non hanno aggiunto altre indicazioni oltre a fornire il proprio assenso. Cinque utenti, invece, hanno aggiunto alcuni degli aspetti dello strumento per cui hanno risposto affermativamente alla domanda: la possibilità di creare dei progetti, il fatto che lo strumento sia utilizzabile da sviluppatori Web e non, le indicazioni aggiuntive che lo strumento può offrire quando richieste. Un utente ha comunque specificato che nonostante l'indubbia utilità dello strumento, avrebbe preferito che in alcune occasioni venisse utilizzato un linguaggio meno specialistico.

La successiva domanda chiedeva agli utenti se avessero qualche indicazione circa informazioni che potrebbero aggiungersi a quelle già presenti del tool. La totalità dei partecipanti ha dichiarato di non essere in grado di fornire indicazioni circa informazioni da aggiungere, anche se un utente ha menzionato la velocità di caricamento delle pagine come informazione che potrebbe essere aggiunta a quelle già previste. Inoltre, un utente ha suggerito di inserire un'indicazione circa i link non funzionanti che potrebbero essere presenti nelle pagine analizzate. Queste ultime due informazioni, comunque, non sono strettamente connesse con le indicazioni fornite dalle linee guida di accessibilità WCAG.

Analizzando le risposte fornite dai partecipanti al test utente, è possibile affermare che tutti siano rimasti abbastanza soddisfatti dallo strumento, considerando le numerose novità introdotte, sia a livello di interfaccia che di funzionalità. Inoltre, molti utenti non avevano una previa conoscenza dello strumento né particolari competenze nella tematica relativa all'accessibilità Web. Gli aspetti che hanno riscosso maggiore apprezzamento sono stati la funzionalità del progetto, dunque la possibilità di poter monitorare nel tempo l'accessibilità delle risorse Web, e la chiarezza nell'esposizione delle informazioni. Suggerimenti per migliorare lo strumento hanno riguardato aspetti minori relativi a una maggiore evidenziazione delle funzionalità offerte, così come una revisione dei testi informativi in un'ottica di semplificazione, in modo tale da rendere lo strumento ancora più utile agli utenti che non possiedono particolari competenze nell'ambito dell'accessibilità Web.

4.3.1 I suggerimenti dei partecipanti e i possibili accorgimenti da apportare all'interfaccia

Dal test utente non sono emerse particolari criticità nei termini della facilità di utilizzo e della chiarezza delle informazioni presentate, tuttavia, alcuni elementi dell'interfaccia utente sembrano richiedere degli accorgimenti. Per quanto riguarda la sezione "Live preview", la funzionalità di collegamento con la vista "Web developer" è risultata poco evidente, è dunque possibile intervenire sugli elementi di collegamento (i link presenti nelle singole occorrenze dell'errore), in modo tale che questi ultimi siano più evidenti. Sempre nella stessa vista, il popup che appare al passaggio del mouse sopra l'elemento evidenziato non è di facile intuizione, si potrebbe indicare all'utente tale possibilità inserendo un breve testo esplicativo all'avvio della "Live preview". Gli elementi che hanno coinvolto maggiormente i partecipanti al test sono stati i grafici riepilogativi delle analisi effettuate, proprio per questo motivo sono state forniti alcuni suggerimenti su come migliorare l'esperienza d'utilizzo degli stessi. A fronte di ciò, sarebbe utile riprogettare i grafici presenti attraverso una libreria Javascript più performante, ad esempio D3.js, così da garantire una maggiore profondità di analisi all'interno di un unico grafico, invece di dividere le informazioni in più sezioni della pagina. Un ultimo

punto su cui si crede bisognerebbe riflettere e apportare alcune modifiche, è relativo al linguaggio utilizzato nell'interfaccia nella descrizione degli errori individuati. Alcuni utenti hanno espresso una certa difficoltà nel comprendere quale fosse realmente l'errore e, di conseguenza, avrebbero avuto bisogno di uno sforzo ancora maggiore qualora avessero dovuto provvedere alla correzione dello stesso. A tal proposito si potrebbero riscrivere le definizioni delle linee guida, dei criteri e delle tecniche WCAG che il W3C offre su sul portale dedicato, in un linguaggio più discorsivo, fornendo contestualmente degli esempi pratici di codice HTML o CSS corretto da cui poter prendere ispirazione per le correzioni da apportare al proprio sito Web.

5 Conclusioni

Durante lo svolgimento di questo progetto sono state sviluppate nuove funzionalità per lo strumento automatico di valutazione di accessibilità MAUVE++.

Prima di procedere allo sviluppo e all'implementazione di queste nuove funzioni, sono stati elaborati una serie di requisiti che potrebbero essere alla base dello sviluppo di questi strumenti in un'ottica rinnovata e al passo con le novità tecnologiche e legislative.

Dopo aver introdotto il concetto di accessibilità Web e aver descritto le linee guida WCAG, sono stati analizzati i limiti e le potenzialità degli strumenti di valutazione automatica di accessibilità.

Avendo, dunque, inquadrato il contesto di riferimento nel quale MAUVE++ trova la sua collocazione funzionale, si è passati alla descrizione dei requisiti intesi come necessari per lo sviluppo di una "nuova" generazione di strumenti automatici di valutazione dell'accessibilità Web. Dopo aver esposto la nuova architettura dello strumento, analizzandone sommariamente le varie parti coinvolte, dall'interfaccia al motore di validazione, è stata posta l'attenzione alle nuove tecnologie di sviluppo Web, come i framework Javascript che creano pagine Web altamente dinamiche, e le connesse problematiche che uno strumento come MAUVE++ può incontrare nell'espletamento delle sue funzioni. Così come si è evidenziata la volontà di fornire uno strumento in grado di soddisfare le richieste di un'utenza con un diverso grado di preparazione circa gli aspetti concettuali e implementativi delle "regole" di accessibilità, fornendo molteplici soluzioni grafiche ai risultati cosicché potessero essere utili sia agli "End user" che ai "Web developer". Sono state descritte le funzionalità di "Server side rendering" e della "Live preview", sia da un punto di vista implementativo, dunque con qualche delucidazione sul codice di sviluppo, sia dal punto di vista dell'usabilità dell'interfaccia grafica, ricorrendo a riproduzioni della stessa interfaccia con annotazioni a riguardo.

Citando le nuove regole europee vigenti circa la necessità per gli enti pubblici di fornire contenuti Web accessibili e soprattutto l'obbligatorietà per gli stessi, di effettuare una valutazione di accessibilità delle proprie risorse disponibili in rete con una frequenza periodica, nonché di fornire la reportistica legata ai risultati ottenuti da queste valutazioni, è stato descritto il processo di progettazione e sviluppo della funzionalità "Progetto". Dopo aver motivato la decisione di ristrutturazione del database sulla base delle nuove necessità dello strumento e averne descritto i dettagli, sono state riportate le due fasi di sviluppo relative rispettivamente al backend e al frontend della nuova funzionalità. Con l'ausilio di figure tratte dall'interfaccia dello strumento, sono state descritte le nuove possibilità offerte all'utenza di MAUVE++ relative al monitoraggio dell'accessibilità Web.

Il successivo capitolo ha riguardato il test utente condotto per valutare il grado di usabilità dello strumento a fronte delle novità introdotte. I paragrafi sono stati dedicati rispettivamente: agli utenti partecipanti, alla descrizione dei task proposti e alle domande del questionario associate a ciascuno di essi, ai risultati ottenuti riportando i dati oggettivi e le impressioni che i partecipanti hanno espresso.

Le nuove funzionalità progettate e implementate, descritte in questa tesi di laurea, sono già disponibili all'interno dell'interfaccia Web dello strumento e dunque utilizzabili.

Volgendo lo sguardo oltre, è possibile scorgere numerose possibilità di sviluppo di nuove funzionalità per MAUVE++: migliorare la parte relativa alla risoluzione dei problemi, ampliando e migliorando i suggerimenti tecnici adottabili dagli sviluppatori ricorrendo ad esempi implementativi di facile comprensione; fornire agli esperti di accessibilità la possibilità di integrare le proprie analisi a quelle dello strumento, così da permettere la generazione di una reportistica più dettagliata e affidabile; migliorare l'esperienza di analisi dei dati ottenuti attraverso dei grafici più interattivi; vista l'imminente pubblicazione delle WCAG 2.2, prevista per il

settembre 2022[80], sarà necessario implementare le eventuali novità, in termini di linee guida, criteri e tecniche, che saranno introdotte da questo nuovo documento.

[80] https://www.w3.org/WAI/standards-guidelines/wcag/new-in-22/

Bibliografia e sitografia

Bibliografia

1. Abascal J., Arrue M., Valencia X. Tools for Web Accessibility Evaluation. In: Yesilada Y., Harper S. (eds) Web Accessibility. Human–Computer Interaction Series. Springer, London, 2019. https://doi.org/10.1007/978-1-4471-7440-0_26

2. Antonio Giovanni Schiavone, M.A.G.EN.T.A. 2.0 : Un sistema per la valutazione di linee guida per l'accessibilità dei siti Web, tesi di Laurea Magistrale Corso Di Laurea in Informatica, Facoltà di Scienze Matematiche Fisiche e Naturali, Università degli Studi di Pisa, anno accademico 2011-2012

3. Antonio Giovanni Schiavone, Fabio Paternò. "An extensible environment for guidelinebased accessibility evaluation of dynamic Web-applications", Universal Access in the Information Society v. 14, no. 1, marzo 2015, pp. 111-132

4. European Telecommunications Standards Institute, Comité Européen de Normalisation, Comité Européen de Normalisation Electrotechnique. Accessibility requirements for ICT products and services EN 301 549 V2.1.2, 2018

5. Giorgio Brajnik. Comparing accessibility evaluation tools: a method for tool effectiveness. Universal access in the information society 3, 3-4 2004, pp. 252–263

6. Marco Manca, Vanessa Palumbo, Fabio Paternò, Carmen Santoro. The Transparency of Automatic Web Accessibility Evaluation Tools: Design Criteria, State of the Art and User Perception, submitted paper, EICS, 2018.

7. Maria Björkman. Inter-tool Reliability of Three Automated Web Accessibility Evaluators, in Jiang, L., Jonsson, A., & Vanhée, L..

Proceedings of Umeå's 25th Student Conference in Computing Science, USCCS 2022. pp. 15-25 http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-191144

8. Melody Y. Ivory and Marti A Hearst. The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. 33, 4, dicembre 2001, pp. 470–516. DOI: https://doi.org/10.1145/503112.503114
9. Nicola Iannuzzi, Marco Manca, Fabio Paternò, Carmen Santoro. Engineering Automatic Support for Web Accessibility Validation, submitted paper, EICS, 2022
10. Santos Tania, Carlos Duarte. Comparing accessibility evaluation plug-ins, In W4A '20: Proceedings of the 17th International Web for All Conference, 2020, pp. 1-11

Sitografia

11. Absolute vs. Relative Paths/Links | CoffeeCup Software https://www.coffeecup.com/help/articles/absolute-vs-relative-pathslinks/
12. AngularJS — Superheroic JavaScript MVW Framework https://angularjs.org/
13. Bootstrap · The most popular HTML, CSS, and JS library in the world. (getbootstrap.com) https://getbootstrap.com/
14. Chart.js | Open source HTML5 Charts for your Website (chartjs.org) https://www.chartjs.org/
15. Chromium https://www.chromium.org/Home/
16. Cross-Origin Resource Sharing (CORS) e intestazione Access-Control-Allow-Origin - CoreTech https://www.coretech.it/it/service/articoli/articoli.php?ID=1425

17. CSS-Tricks - Tips, Tricks, and Techniques on using Cascading Style Sheets. https://css-tricks.com/

18. EUR-Lex — Access to European Union law https://eur-lex.europa.eu/

19. GitHub - puppeteer/puppeteer: Headless Chrome Node.js API https://github.com/puppeteer/puppeteer

20. GitHub - radkovo/jStyleParser https://github.com/radkovo/jStyleParser

21. Google Developers https://developers.google.com/

22. Home: Java Platform, Standard Edition (Java SE) 8 Release 8 (oracle.com) https://docs.oracle.com/javase/8/

23. it.knowledgr.com https://it.knowledgr.com/

24. JSON https://www.json.org/json-it.html

25. MDN Web Docs (mozilla.org) https://developer.mozilla.org/en-US/

26. Node.js (nodejs.org) https://nodejs.org/it/

27. React – Una libreria JavaScript per creare interfacce utente (reactjs.org) https://it.reactjs.org/

28. Servlet: tutorial e primi passi con esempi Java | HTML.it https://www.html.it/articoli/primi-passi-con-le-servlet/

29. Siteimprove - SEO, Accessibility, Analytics, GDPR, & More https://siteimprove.com/

30. Siteimprove Accessibility Checker - Chrome Web Store (google.com) https://chrome.google.com/Webstore/detail/siteimprove-accessibility/djcglbmbegflehmbfleechkjhmedcopn

31. UX Hints – Product Design Knowledge https://uxhints.com/

32. Vue.js - The Progressive JavaScript Framework | Vue.js (vuejs.org) https://vuejs.org/

33. W3Schools Online Web Tutorials https://www.w3schools.com/

34. WAVE Web Accessibility Evaluation Tool (Webaim.org) https://wave.Webaim.org/

35. Web Content Accessibility Guidelines: WCAG by the Numbers | eSSENTIAL Accessibility https://www.essentialaccessibility.com/blog/Web-content-accessibility-guidelines

36. What is a base URL? | Web Technology Training | Oregon State University https://Webtech.training.oregonstate.edu/faq/what-base-url

37. What Is a Relational Database | Oracle https://www.oracle.com/database/what-is-a-relational-database/

38. Wikipedia, the free encyclopedia https://en.wikipedia.org/wiki/Main_Page

39. World Wide Web Consortium (W3C) https://www.w3.org/