ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI SCIENZE
Corso di Laurea in Informatica

SahARIAn: Uno strumento visuale per la progettazione di pagine web accessibili

Relatore:
Chiar.mo Prof.
Fabio Vitali

Presentata da:
Luca Morosini

III Sessione
2017/2018

Abstract

L’accessibilità è un tema importante che fa parte anche di Internet e delle pagine web e consente a tutte le persone, indipendente dalle capacità fisiche e mentali, di accedere ai medesimi contenuti e servizi. Il problema sta nelle difficoltà che incontrano gli sviluppatori di siti web durante la scrittura degli stessi, spesso con risultati non sufficienti e che non soddisfano l’utenza con disabilità, nonostante le diverse linee guida rilasciate nel corso degli anni e il progresso tecnologico nella creazione di strumenti sempre più sofisticati, fatti per agevolare lo sviluppo degli sviluppatori.
In questa tesi viene proposto uno strumento che opera in maniera diversa da quelli attuali, con lo scopo di dare un contributo ed un miglioramento a questo settore.

Indice

Capitolo 1

Introduzione

Internet è senza dubbio uno dei migliori avvenimenti in cui le persone con disabilità potessero sperare. Prima di Internet, per esempio, le persone non vedenti non potevano leggere giornali o riviste dato che i prodotti scritti in Braille erano molto costosi rispetto ai normali giornali e quindi, per poterne conoscere il contenuto, avevano sempre la necessità di qualcun altro che li leggesse per loro.
Ora invece, la maggior parte dei quotidiani pubblica online il contenuto delle versioni cartacee e lo rende disponibile anche tramite le applicazioni per dispositivi mobili, in questo modo anche le persone con problemi di vista possono in autonomia consultare a loro piacimento qualsiasi contenuto semplicemente aprendo un browser e utilizzando il sintetizzatore vocale per ascoltarne il testo.
Lo stesso vale anche per altre tipologie di disabilità che riguardano le parti uditive, motorie e cognitive.
Nonostante questo potenziale, però, c'è ancora molto lavoro da fare perché molti siti sono ancora navigabili solo con il mouse, rendendoli inaccessibili a chi ha disabilità visive e motorie, altri invece non descrivono con accuratezza le immagini contenute o non lo fanno completamente e altri ancora presentano ulteriori problematiche distinte di vario genere.
Ogni disabilità ha bisogno di adattamenti specifici che però, nella maggior parte dei casi, garantiscono benefici per tutti: per esempio avere immagini con illustrazioni più ricche agevola un non vedente nel capire il significato di un articolo, ma questo ulteriore dettaglio può essere un aiuto sia per persone con altro tipo di disabilità sia per utenti privi di problematiche.
Inoltre, pensando al web come un'opportunità, è chiaro che gli sviluppatori hanno una grande responsabilità nel far sì che questa occasione non venga sprecata, altrimenti si rischia di ottenere l'opposto, cioè creare di nuovo una

barriera nella vita delle persone con disabilità, come era presente prima di Internet.
È allora molto importante cercare di dare agli sviluppatori di siti Internet il materiale necessario affinché possano creare pagine web nella maniera più accessibile possibile e allo stesso tempo farlo in modo facile per non gravare troppo sui tempi di sviluppo e quindi sui costi di produzione.

Scopo di questa dissertazione è sostenere che dotando gli sviluppatori web di strumenti che permettono di rappresentare visivamente le notazioni per l'accessibilità, diventa più semplice ed immediato progettare adeguatamente pagine web accessibili.
Lo strumento che andrò a descrivere, può infatti essere impiegato dagli sviluppatori durante la progettazione di pagine web, similmente a come avviene con altre tipologie di strumenti già in uso da tempo. Per un approfondimento al riguardo, si rimanda ai capitoli 4 e 5.
La necessità nasce dal fatto che, in questo momento, gli strumenti disponibili agiscono da segnalatori delle possibili problematiche di accessibilità, senza una rappresentazione visiva.
In questo elaborato mi focalizzerò in particolare sulla disabilità visiva, quindi sulle persone non vedenti, per le quali è possibile ridurre il problema utilizzando un buon web design e per esso si intendono tutte le buone pratiche e strumenti utili a far sì che una persona non vedente possa capire il contenuto di una pagina web.

Come accennato in precedenza, le persone con disabilità visiva hanno come ausilio alla lettura da computer o dispositivi elettronici lo strumento della sintesi vocale, che permette loro di capire il contenuto ascoltando una riproduzione vocale di qualsiasi testo, compreso il contenuto di una pagina web.
Le caratteristiche di un lettore vocale sono le seguenti [DPS04]:

- Legge ogni cosa, inclusi gli elementi di HTML utili solamente alla visualizzazione e che quindi non aggiungono significato al testo.

- La lettura delle pagine è lineare e sequenziale da sinistra verso destra e dall'alto verso il basso.
  Questo rende noiosa la lettura dovendo aspettare del tempo per arrivare alle informazioni utili e per ovviare a questo il lettore vocale legge molto velocemente il testo, fino a quando l'utente non sente

qualcosa di suo interesse, simulando quello che un utente dotato di vista fa con gli occhi scorrendo velocemente il testo fino ad individuare le parole chiave.

- Non sono in grado di capire automaticamente l'organizzazione della pagina e la rilevanza che ha una parte rispetto ad un'altra.
- Alterna la lettura del contenuto con la lettura dei link, creando molta confusione al lettore
- La selezione dei link è macchinosa, perché occorre confermare di volerlo aprire nel momento in cui il sintetizzatore lo legge, ma spesso ci sono problemi di sincronizzazione che rendono impossibile eseguire l'azione voluta.
- Il layout della pagina e tutta la parte grafica (font, colori, ecc...) viene persa perché il lettore vocale non pone enfasi maggiore o minore a seconda del contenuto che sta leggendo.

I lettori vocali dispongono anche di due modalità di lettura della pagina: *document mode* e *application mode*.
La prima, chiamata anche browser mode, consente di muoversi liberamente all'interno della pagina web anche tra contenuti non selezionabili da tastiera, tipicamente aree di testo, ma non permette di interagire con il contenuto della pagina come per esempio riempire un form.
La seconda è necessaria laddove la prima non permette l'interazione e quindi è una modalità utile a permettere all'utente di eseguire operazioni sulla pagina già visitata attraverso la document mode.

Queste funzionalità però hanno in particolare due punti deboli:

- *Immagini*: non possono descriverle automaticamente, di conseguenza lo sviluppatore deve inserire un testo per spiegarne il contenuto in maniera esaustiva
- *Layout*: non possono descrivere globalmente una pagina web come si può fare a colpo d'occhio invece, hanno la capacità di leggere linearmente e una parola per volta il contenuto della pagina

Avendo questi limiti si capisce che le pagine web devono essere ben strutturate, ricche e complete di tutti gli attributi necessari per fare in modo

che vengano ben interpretate dal sintetizzatore vocale, il quale altrimenti potrebbe leggere in maniera errata e disordinata il contenuto rendendolo incomprensibile.
Il punto fondamentale è che un cattivo design della pagina web può non ripercuotersi sull'aspetto estetico di essa, infatti una pagina potrebbe essere graficamente accattivante e utilizzabile senza problemi da un utente dotato di vista, nonostante nasconda al suo interno una struttura mal progettata, la quale pregiudica la fruizione dei contenuti da parte di persone non vedenti per i motivi discussi prima.
Per questa ragione è necessario adottare uno strumento che permetta di ridurre al minimo il fattore grafico di una pagina, in modo da focalizzare l'attenzione dello sviluppatore sul design dello scheletro della pagina.

Creare un sito web accessibile non è cosa facile, è necessario impegno per conoscerne i problemi al fine di non incorrere in errori con troppa facilità rendendo i siti inaccessibili, è essenziale che le grandi compagnie rendano sempre più centrale questo tema e in ultimo è indispensabile informarsi sugli standard attuali e sulle procedure da seguire per consentire uno sviluppo coerente e uniformato.
L'accessibilità dei siti web quindi è un tema molto complesso e vario che ha bisogno di linee guida per i motivi appena spiegati con lo scopo di definire metodi e soluzioni chiare in modo da poter progettare al meglio pagine web accessibili: queste sono le Web Content Accessibility Guidelines (WCAG), pubblicate dal Web Accessibility Initiative (WAI), che a sua volta fa parte del World Wide Web Consortium (W3C[1]).
Le WCAG, attualmente alla versione 2.0 che sostituiscono la prima versione 1.0, offrono quindi delle linee guida utili ai progettisti di pagine web, ma anche agli sviluppatori di strumenti per valutarne l'accessibilità.
Queste prime linee guida saranno poi trattate con maggior profondità nel capitolo 2.

Negli ultimi tempi però, c'è una tendenza generale ad implementare pagine web molto più ricche di contenuti dinamici rispetto al passato attraverso tecnologie di aggiornamento asincrono che consentono di aggiornare il contenuto della pagina senza effettuare un refresh[2]; questa tecnica si chiama

[1] Una ONG che ha lo scopo di migliorare il World Wide Web.

[2] Ricaricamento completo della pagina attraverso il tasto F5 oppure il pulsante apposito nella barra degli strumenti del browser.

AJAX (Asynchronous Javascript and XML) e consente appunto di realizzare applicazioni web dinamiche e interattive denominate Rich Internet Application.
Le Rich Internet Application aumentano di molto la qualità di fruizione dei contenuti di un sito, rendendolo più veloce, immediato e piacevole da navigare, ma hanno il difetto di rendere poco accessibili i contenuti che vengono aggiornati in questa maniera e, dato che questa tecnica di sviluppo web si è ormai diffusa in larga scala, è evidente come si siano creati allo stesso tempo numerosi problemi da risolvere per quanto riguarda l'accessibilità.

Per risolvere questa situazione il W3C, ha pubblicato un insieme di documenti per migliorare l'accessibilità dei contenuti dinamici e dei componenti dell'interfaccia utente, che vanno sotto il nome di WAI-ARIA (Web Accessibility Initiative – Accessible Rich Internet Applications), specificando come aggiungere una semantica o altri metadati al codice HTML allo scopo di rendere i controlli lato utente e i contenuti dinamici più accessibili.
A supporto di questo tema vengono fatte numerose conferenze in tutto il mondo, nelle quali si discute delle problematiche che attualmente rendono difficile il raggiungimento dell'accessibilità ai siti web, si questiona anche di come meglio progettare il design e i comportamenti di questi in modo da renderli più comprensibili ai non vedenti.
Io mi focalizzerò di più sul portare articoli volti a migliorare la progettazione delle pagine web per tutti, in modo che al termine delle analisi si riescano ad avere gli strumenti necessari per comprendere con abbastanza profondità quali siano le problematiche più importanti.

Molti strumenti già disponibili per il controllo dell'accessibilità ai siti web si basano sulle linee guida WCAG e sui documenti WAI-ARIA, combinandone le caratteristiche per ottenere dei software in grado di evidenziare automaticamente le mancanze in termini di metadati, caratteristiche e comportamenti non conformi alle indicazioni fornite dal W3C.
Risultano utili perché in breve tempo fanno capire allo sviluppatore quali sono o possono essere le debolezze della pagina, ma non danno una rappresentazione visiva di esse e, come si è dimostrato da alcune ricerche della comunità scientifica dettagliate al capitolo 3, questi metodi hanno

ancora delle lacune abbastanza evidenti in termini di copertura dei problemi degli utenti.

È allora necessario provare a dare agli sviluppatori strumenti diversi per sviluppare i siti web, quello da me proposto in questa tesi si chiama SahARIAn ed è, in breve, un plugin per i browser web quali Chrome e Firefox che, una volta attivato, rappresenta visivamente le notazioni di accessibilità ARIA, alterando la pagina visualizzata e rendendola trasparente a quella che è la sua struttura, evidenziando in modo molto chiaro le problematiche presenti in essa.
Meno la pagina è accessibile, maggiore sarà l'evidenza di ciò per una persona dotata di vista in quanto il plugin renderà il suo contenuto molto disordinato e non graficamente gradevole.
Di questo strumento ne esaminerò le funzionalità all'interno del capitolo 4, nel quale porterò anche qualche esempio di caso d'uso per poi analizzare tecnicamente la struttura di esso nel 5° capitolo.

Capitolo 2

WCAG e il supporto per l'accessibilità delle pagine web

Precedentemente ho parlato di linee guida utili allo sviluppo di siti web accessibili chiamate Web Content Accessibility Guidelines (WCAG) che fanno parte di una serie di linee guida pubblicate dal Web Accessibility Initiative (WAI), appartenente al World Wide Web Consortium (W3C).
Partendo allora proprio da WCAG, analizzo le sue caratteristiche per poi arrivare alle motivazioni che hanno spinto il W3C a pensare a nuove soluzioni per migliorare l'accessibilità con l'introduzione di ARIA.

2.1: WCAG 1.0 e 2.0

La prima versione di queste linee guida, la 1.0, venne rilasciata nel 1999 e specifica 3 livelli di priorità di requisiti che devono (A), dovrebbero (AA) o possono (AAA) essere rispettati a seconda del livello al quale appartengono, in questo modo tutto il mondo ha potuto uniformarsi sotto alcuni standard da seguire e che, se implementati, permettevano già da allora di rendere più semplice l'accesso ai siti per gli utenti non vedenti.
L'orientamento di queste linee guida era più di carattere tecnico ed erano caratterizzate da 14 principi, il che le rendeva un po' dispersive.

Col passare degli anni, ovviamente, quelle linee guida cominciarono a risultare obsolete visto il progredire delle tecnologie web e dei metodi di sviluppo dei siti.
Inoltre, uno dei limiti della prima versione stava nella rigidità dell'orientamento tecnico di esse, che non poneva l'accento sul perché si dovesse fare una certa cosa, ma sul farla e basta e questo poteva causare comunque degli effetti collaterali: se per esempio si considera la regola di

dover aggiungere un testo alternativo come attributo delle immagini, avendo capito il perché lo si utilizza, si può scrivere un qualcosa di utile per un non vedente, nel caso opposto invece si potrebbe rischiare di creare un testo seguendo la linea guida, ma che non è abbastanza dettagliato o che specifica punti meno importanti per un disabile di quel genere, risultando infine poco utile alla causa per cui si è pensato di adottare quella tecnica.
Nel 2008 allora venne rilasciata la versione 2.0 che ridefinisce i tre livelli della prima versione correggendoli in modo che siano più attuali e orienta le linee guida verso dei principi cardine, 4 per l'esattezza[3]:

- *Percettibile (perceivable)*: il modo in cui le informazioni vengono codificate deve essere percepibile dalle persone, di conseguenza se una persona ha problemi di vista, bisogna trovare un modo alternativo per farle arrivare le informazioni che non riesce ad ottenere attraverso gli occhi in altre modalità.
  Una buona pratica è anche di separare il contenuto dallo stile, non comunicando troppo attraverso la presentazione di essa, per far sì che la pagina anche senza la parte grafica comunichi gli stessi messaggi.

- *Interattiva (operable)*: dopo aver percepito un'informazione è spesso necessario svolgere azioni, solitamente con il mouse, per interagire con una pagina web.
  Il problema del mouse è che non è accessibile da persone non vedenti che sono costrette ad utilizzare la tastiera, quindi i siti web, per essere fruibili da queste persone, devono offrire possibilità di input attraverso metodi alternativi, dare più tempo per l'interazione con il sito e avere metodi di recovery nel caso di errore.

- *Comprensibile (understandable)*: un contenuto web percepibile e interagibile non è accessibile se non è anche comprensibile, per far sì che lo sia si ricorre all'utilizzo di un linguaggio più semplice possibile, adattandosi al pubblico a cui è rivolto e inoltre, per meglio renderne facile la comprensione, si possono aggiungere informazioni espresse in modi alternativi a quelli standard in maniera che ci siano più possibilità che il contenuto venga recepito con correttezza.

[3] http://www.dirtywork.it/blog/accessibilita-siti-web.

- *Duratura (robust)*: ogni persona ha la libertà di utilizzare la tecnologia che più la aggrada per consultare un sito web, differenti sistemi operativi, differenti browser ecc. e per ottenere una buona robustezza di un sito e dei contenuti al suo interno, gli sviluppatori devono garantirne il supporto per il più lungo periodo possibile, dato che anche in questo caso, se una persona può percepire, interagire e capire un contenuto, ma non può accedere ad esso per problemi di requisiti minimi del contenuto, allora quel sito non è accessibile.

Utilizzando principi come linee guida anziché regole tecniche, la versione 2.0 risulta nuova e più flessibile di quanto lo fosse la precedente versione, aumentando la consapevolezza negli sviluppatori di quali fossero i veri scopi della progettazione di siti web accessibili, aprendo le porte a numerose tecniche diverse con lo stesso obiettivo e aumentando così la varietà e la precisione delle implementazioni.

Successivamente un'altra evoluzione ha investito il mondo del web, creando nuovi modi di sviluppare i siti avvalendosi di tecnologie per rendere la fruizione dei contenuti più rapida ed intuitiva: si tratta delle pagine dinamiche e AJAX.

2.2: Il problema delle pagine dinamiche e AJAX

Negli ultimi anni si è diffuso un nuovo modo di sviluppare siti web che rende le pagine più dinamiche, più veloci e di maggior semplicità di utilizzo, questa tecnologia permette di realizzare contenuti migliori denominati Rich Internet Application.

Il metodo si ottiene con l'unione di diverse tecnologie web che permettono interazioni tra il client e il server in maniera dinamica senza la necessità di ricaricamenti della pagina e prende il nome di Asynchronous Javascript and XML (AJAX).

Infatti utilizzando AJAX vengono modificate e aggiornate solamente le parti delle pagine web interessate dalla modifica e non tutta la pagina, un esempio può essere fatto prendendo le Web App che offrono servizi di news feed: in quei casi, alla pubblicazione di una nuova notizia, non è necessario aggiornare l'intera pagina, ma basterà aggiungere quella nuova in cima all'elenco delle news.

Questo comportamento, oltre a rendere più immediata la fruizione di un

servizio di quel tipo, genera anche un risparmio notevole sui dati consumati dal sito per poter essere consultato: nel caso in cui non si utilizzi AJAX, all'arrivo di una nuova notizia, si è costretti a ricaricare la pagina per intero, richiedendo nuovamente il contenuto del sito che si è già visto, solo per ottenere quella nuova notizia in più; in caso contrario, utilizzando AJAX, verrebbe richiesto al server solamente la nuova notizia che andrà ad aggiungersi al contenuto già scaricato in precedenza accedendo per la prima volta alla pagina.

Questo ha dato il via al cosiddetto Web 2.0, denominato così proprio per le differenze marcate che lo contraddistinguono dal web così come era conosciuto prima, capace di portare una ventata di aria fresca in un momento di stallo che durava da tempo nel mondo di Internet.
Il miglioramento è tangibile e molto utile per le persone prive di disabilità, ma si può dire lo stesso per chi ha problemi di vista o altro genere?
Purtroppo no, infatti se da un lato le pagine diventano più ricche e piacevoli da consultare, dall'altro perdono di accessibilità in maniera abbastanza consistente, soprattutto per quanto riguarda i contenuti dinamici derivanti da aggiornamenti frequenti.
Esistono infatti tre situazioni nelle quali, senza aiuti, diventa impossibile per un non vedente interpretare correttamente la pagina [HGB09]:

- *Notifiche*: un utente non può rendersi conto dell'aggiornamento del contenuto e quindi non riesce a localizzare la posizione del possibile nuovo testo.

- *Layout*: l'utente può rendersi conto dopo del tempo che la struttura della pagina è improvvisamente cambiata ed è costretto a rileggere tutta la pagina per capire in che modo è cambiata.

- *Feedback di conferma*: quando un utente esegue azioni attraverso pulsanti o bottoni non ha la certezza che quello che si è eseguito sia andato a buon fine o meno.

Sono tutte e tre situazioni molto scomode e frustranti per un utente che ha bisogno di usare un determinato servizio, come per esempio un feed di notizie nel quale, all'arrivo di una nuova notizia, una persona non vedente non può accorgersi né della notifica né del cambiamento del layout della pagina ed è costretto ad impiegare molto più tempo del dovuto.

Oltre a questi tre grossi problemi, rimangono non risolte altre problematiche che diminuiscono l'accessibilità di un sito web, come per esempio la difficoltà nel capire qual è la parte principale del documento e quali sono quelle secondarie oppure se, durante la compilazione di un form, un campo è obbligatorio o meno ai fini di poter inviare il contenuto.
Per arginare questi problemi è allora necessario adottare delle contromisure attraverso l'adozione di nuovi metodi di progettazione delle pagine web.

2.3: La nascita di ARIA

Il W3C, in seguito a queste problematiche, ha introdotto nuovi documenti che spiegano in maniera specifica come affrontare il problema dell'accessibilità nel contesto delle pagine web dinamiche sviluppate attraverso AJAX. Questi documenti prendono il nome di WAI-ARIA (Accessible Rich Internet Application).
WAI-ARIA è in sintesi una specifica, cioè una raccolta di indicazioni rivolta agli sviluppatori che definisce una serie di attributi HTML aggiuntivi da poter essere inseriti all'interno dei tag classici di HTML, in modo da aumentarne il valore semantico e di conseguenza migliorare l'accessibilità dove è necessario.
Questo avviene attraverso tre caratteristiche aggiuntive principali[4]:

- *Ruoli*: definiscono cosa sia un elemento e quale sia la sua funzione all'interno della pagina.
- *Proprietà*: aggiunge un significato extra agli elementi specificandone una proprietà, per esempio se un campo di un form è richiesto o meno.
- *Stati*: come dice la parola si tratta di proprietà momentanee che possono variare nel tempo, ma sono utili all'utente per capire in che stato si trova un certo elemento.

Queste tre caratteristiche migliorano l'accessibilità perché danno modo agli sviluppatori di dettagliare più correttamente le caratteristiche di ogni elemento della pagina facendo in modo che le tecnologie di assistenza possano descrivere meglio gli elementi della pagina, per esempio attraverso l'attribuzione del ruolo "main" ad un elemento, si può dare l'indicazione al

[4] https://developer.mozilla.org/it/docs/Learn/Accessibilit%C3%A0/WAI-ARIA_basics.

sintetizzatore vocale di avvisare l'utente che quella parte di documento contiene il contenuto principale e così con tutte le altre aree della pagina, risolvendo uno dei problemi precedentemente ravvisati.
Inoltre la combinazione di esse rende accessibili anche le più recenti Web App che si basano sulla chat o sugli aggiornamenti live delle pagine, che senza questo tipo di supporto sarebbero completamente inutilizzabili da persone con disabilità visiva.
È comunque importante prestare attenzione a come vengono aggiunti gli attributi ARIA al codice HTML perché vanno utilizzati solo quando necessario e soprattutto non usati a sproposito, altrimenti, come vedremo attraverso un esempio nel capitolo 4, si rischia di creare più problemi rispetto a quelli che si tenta di risolvere, facendo interpretare male all'utente un contenuto.

Questa analisi delle linee guida attualmente disponibili e delle problematiche che tentano di risolvere, mi ha permesso di indirizzare meglio le mie ricerche verso quali sono gli strumenti che, combinati con le linee guida, aiutano gli sviluppatori a progettare le pagine web in modo da essere consultabili da qualsiasi categoria di utente.

Capitolo 3

Strumenti per la progettazione di siti web accessibili

Fin'ora ho parlato del significato di accessibilità nei siti Internet, chi ne soffre di più e quali tecniche, linee guida e principi sono stati indicati per migliorare la situazione.
Introduco allora gli strumenti che sono a disposizione degli sviluppatori per progettare i siti web e per controllarne l'accessibilità, ma non prima di aver fatto una ricerca ad-hoc attraverso la comunità scientifica per individuare quali di questi abbiano un peso specifico maggiore.

3.1: La ricerca scientifica

La comunità di ricerca scientifica, ogni anno, organizza diverse conferenze internazionali per discutere e aggiornarsi sugli ultimi sviluppi e sulle nuove tecnologie disponibili per quanto riguarda il web ed io ne ho raccolte alcune per la necessità di capire meglio quali siano gli standard attuali e dove invece sono ancora presenti delle lacune, in modo da rendermi conto dei bisogni insoddisfatti degli sviluppatori web, i principali soggetti di questa dissertazione, per poi sviluppare al meglio una mia soluzione di possibile interesse per la comunità scientifica.

Sono stati condotti diversi studi che, a seguito di prove empiriche, sono arrivati a conclusioni attraverso l'impiego di utenti disabili nell'analisi dei problemi di accessibilità dei siti web, uno tra questi è un interessante studio [PPP12] del 2012 che tratta proprio i problemi che riscontrano gli utenti privi di vista su Internet: vengono scelti 32 persone di età mista e non vedenti e 16 siti web contenenti diverse tipologie di problemi di accessibilità e vengono fatti utilizzare questi siti a scopo di ricerca di quali siano i problemi, coperti o

non coperti da WCAG e quanti dei problemi descritti dalle linee guida sono risolti, analizzando inoltre i benefici portati dalle due edizioni di esse.
Il test ha evidenziato diversi risultati inaspettati:

- WCAG 1.0 copriva per il 43.3% i problemi degli utenti, dei quali solo il 5.7% erano implementati e risolti dagli sviluppatori.
- WCAG 2.0 invece copriva il 50.4% dei problemi, ma dei quali sempre una piccola percentuale dell'8.4% era risolta.
- La maggioranza dei problemi riscontrati e non coperti riguardava i contenuti nelle pagine:
    - Contenuti non trovati dove ci si aspettava
    - Contenuti trovati dove non ci si aspettava
- Dei problemi coperti ed implementati, la categoria della destinazione dei link non chiara è la più significativa numericamente, seguita dalla mancata descrizione vocale dei contenuti multimediali

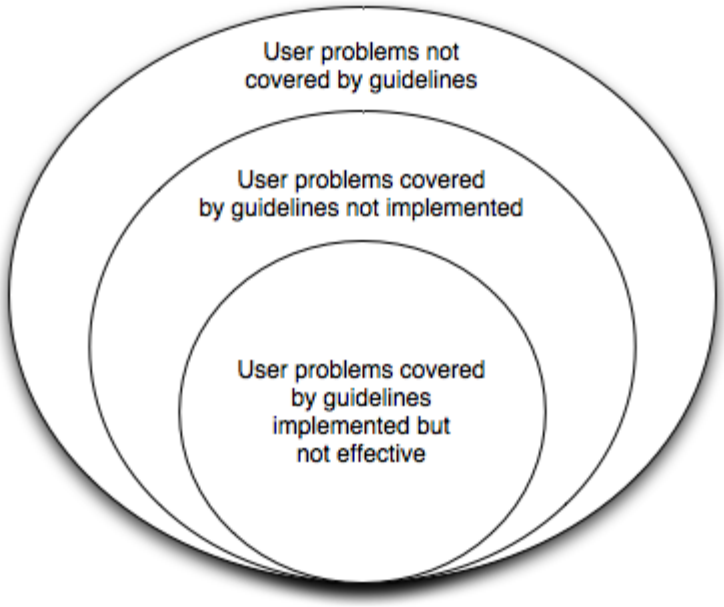

Figura 3.1: Insieme dei problemi degli utenti e copertura delle linee guida

In conclusione si può dire che le WCAG coprono la metà dei problemi di accessibilità riscontrati dagli utenti e che la versione 2.0 aumenta la copertura rispetto alla 1.0, ma in maniera poco significativa e quindi non nel

modo che ci si aspettava, nonostante l'introduzione dei quattro principi fondamentali.
Soprattutto però, le percentuali di implementazioni dei problemi rimangono molto scarse, testimonianza del fatto che gli sviluppatori web fanno ancora fatica a sviluppare siti accessibili, il che può essere dato da linee guida non chiare o non ben interpretate oppure dalla mancanza di validi strumenti di supporto alla creazione di web accessibile.
Questo dimostra che, come dice il titolo di questo studio, "Le linee guida sono solo la metà della storia", il resto lo possono fare un migliore approccio nella definizione dei principi, basandosi più sulle modalità di utilizzo del web da parte degli utenti piuttosto che sui problemi che incontrano all'interno di pagine web non corrette e un valido supporto dato da strumenti di verifica dell'accessibilità aggiornati e di buona fattura.

Partendo da questo studio, ne introduco un altro, del 2010, che prosegue il discorso delle linee guida WCAG, dell'accessibilità e di come sia possibile aumentare la quantità di problemi coperti da norme e istruzioni in modo da poter essere risolti, combinando tre fattori [DC10]:

- *Le regole WCAG*: linee guida discusse in precedenza.
- *Le linee guida dell'Universal Design (UD)*: metodologia progettuale con la quale si cerca di realizzare prodotti accessibili a tutte le categorie di persone.
- *Lo standard internazionale ISO 9241*: standard che descrive le linee guida per quanto concerne l'interazione uomo-macchina e l'ergonomia.

Per realizzare questa combinazione, viene fatta una mappatura delle regole UD alle WCAG aggiungendo qualche riferimento ad ISO 0241 e alcuni commenti integrativi per spiegare meglio l'unione, quindi ad ogni regola UD possono corrispondere più regole WCAG o viceversa, anche se essendo WCAG specifico per le pagine web è più probabile il primo caso.
Nello svolgere il mapping si sono ottenuti buoni risultati di congruenza con solo una piccola percentuale di principi WCAG mappati in differenti principi UD, ma questo non ha un significato negativo, vuol dire che lo sviluppatore dovrà prestare più attenzione nell'indirizzare la regola WCAG nei corretti principi UD.

UD guideline	WCAG success criterion
2.4. Provide adaptability to the user's pace.	1.4.2. Audio Control
	2.2.1. Timing Adjustable
	2.2.2. Pause, Stop, Hide
	2.2.3. No Timing
	2.2.4. Interruptions
	2.2.5. Re-authenticating
3.4 Arrange information consistent with its importance	2.4.10 Section headings
4.4 Differentiate elements in ways that can be described	

Figura 3.2: Esempio di mapping tra UD e WCAG

Successivamente, per valutare il lavoro svolto, sono stati usati metodi automatici di valutazione come AChecker, ma anche studenti, laureandi, laureati ed esperti nell'accessibilità, dove il loro compito era quello di compilare un questionario per recensire il risultato prodotto dalla valutazione automatica di alcuni siti web attraverso le regole risultanti dalla combinazione dei tre fattori elencati in precedenza.
Al termine della valutazione si sono riscontrati risultati positivi, in quanto l'approccio di combinazione delle linee guida, della valutazione automatica e della verifica di tale valutazione da parte di persone più o meno esperte, ha portato allo sviluppo di ragionamenti e pensieri volti ad analizzare più attentamente il prodotto degli analizzatori automatici, evitando così di fermarsi a correggere meccanicamente gli errori evidenziati e portando ad un livello più alto lo sviluppo di materiale accessibile.

Un ulteriore spunto su come progettare siti ben accessibili deriva da un altro studio un po' più datato, del 2008 [LBO08], dove utilizzando come prima un numero di persone non vedenti per verificare l'accessibilità di un sito web, vengono fatte loro eseguire una ricerca e una navigazione all'interno di un sito.
Come suggerito dallo studio precedente, qui non ci si focalizza sulle linee guida, ma si cercano risposte alla domanda di una migliore accessibilità

attraverso altre vie e, in questo caso, la caratteristica particolare di questo studio sta nel fatto che ricerca e navigazione non vengono fatte soltanto attraverso un'interfaccia grafica classica (GUI), ma anche con l'utilizzo di un'interfaccia testuale migliorata (ETI) utilizzando nove linee guida che vengono definite nell'articolo per costruirla nella maniera migliore, andando poi ad operare non su linee guida basate sui problemi degli utenti, ma sulla rappresentazione delle informazioni dei siti web.
La conseguenza è che gli utenti non vedenti, utilizzando una ETI, riescono a compiere ricerche attraverso il medesimo sito in maniera molto più efficiente di quanto riescano a fare con una GUI, ma quando si tratta di effettuare una navigazione della pagina, i risultati praticamente coincidono e sono spesso dovuti ad un problema di etichette e nomi usati per descrivere le parti del contenuto.
Da questo si può trarre che l'interfaccia giusta in alcuni casi fa la differenza, ma anche che l'accuratezza delle etichette e del testo usato per descrivere parte del contenuto, gioca un ruolo fondamentale in termini di accessibilità e usabilità e questo è un punto debole di molti siti web, che ancora oggi crea diverse difficoltà.

Dalle tre ricerche descritte emergono diversi aspetti:

- Le linee guida WCAG non coprono ogni problema delle persone con disabilità ed è fondamentale migliorare l'approccio all'accessibilità e gli strumenti per verificarla.
- Combinando diverse metodologie e standard si riescono ad ottenere linee guida migliori.
- Ragionando maggiormente sui risultati degli strumenti di valutazione di accessibilità è possibile aumentare la qualità delle correzioni dei problemi di accessibilità.
- L'importanza della modalità con cui viene presentato un contenuto, sia nella forma che nella precisione del contenuto, è fondamentale.

A questo punto ho abbastanza elementi per orientarmi meglio verso la creazione di un sito web accessibile, per questo motivo cito un altro caso di studio, del 2017, dove vengono stilati e analizzati i passi necessari allo sviluppo del sito web di un'autorità locale del governo australiano [CF17].

La ricerca propone questi passi da seguire:

1. *Identificare i problemi della versione attuale del sito*: questo primo passo è molto importante per rendersi conto su cosa è necessario lavorare per migliorare l'accessibilità del sito. In particolare questa fase è composta da due parti: la prima basata sui feedback prodotti dagli utenti utilizzatori del sito, mentre la seconda più tecnica e rivolta a determinare le parti del sito che non soddisfano le linee guida WCAG.

2. *Scegliere uno sviluppatore che conosca i principi e gli standard per l'accessibilità*: è chiaramente importante partire da una base solida che consenta di essere all'avanguardia nella conoscenza della materia dell'accessibilità.

3. *Fornire addestramenti all'accessibilità per gli sviluppatori e i creatori dei contenuti del sito*

4. *Creare un piano di sviluppo del progetto con dei checkpoint di accessibilità*: in questa fase si fa in modo che progredendo con lo sviluppo non si lascino indietro parti sviluppate in maniera poco accessibile in modo che non gravino sul resto del lavoro, quindi frequentemente vengono fatti dei controlli sull'accessibilità.

5. *Identificare nuovi problemi e trovare le soluzioni più appropriate*: durante lo sviluppo della pagina web si possono creare problemi nuovi, è necessario trovarli e risolverli al meglio.

6. *Determinare la qualità del progetto creato*: può essere determinata verificando la longevità del sito rispetto alle linee guida WCAG, ma anche facendo un confronto tra l'accessibilità del vecchio sito rispetto a quello nuovo.

7. *Pianificare i futuri sviluppi e miglioramenti*: è una fase fondamentale per la riuscita di un buon sito web accessibile perché Internet e le sue tecnologie sono in continua evoluzione ed è quindi indispensabile programmare fin da subito nuove verifiche di accessibilità e nuovi piani per sistemarle in modo da aumentare la longevità del sito e del lavoro svolto.

Quest'ultimo studio dimostra la difficoltà e il maggiore impegno necessario allo sviluppo di un sito web accessibile.

Rimane però un ultimo problema da affrontare, cioè quello sulla valutazione dell'accessibilità dei siti web RIA (Rich Internet Application) che rappresentano la stragrande maggioranza delle pagine web ad ora online e che richiedono un trattamento diverso rispetto ai contenuti web statici e privi di aggiornamenti lato client.
Quello che più rende inefficienti gli strumenti di valutazione dell'accessibilità descritti in precedenza, è proprio la dinamicità dei siti che si vanno a valutare, il che rende altamente superficiali le analisi fatte da questi software.

A supporto di questo, uno studio del 2013 [FBC13] è riuscito ad ottenere dei risultati portando a termine un test sulla quantità di stati analizzati dai normali strumenti di valutazione dell'accessibilità delle pagine web.
Uno stato è semplicemente l'insieme dei contenuti che in quel momento la pagina possiede e può nel tempo cambiare come rappresentato nella figura sottostante, per esempio attraverso l'interazione dell'utente con un bottone che, come spesso accade, causa la modifica del contenuto della pagina, creando quindi un nuovo stato.

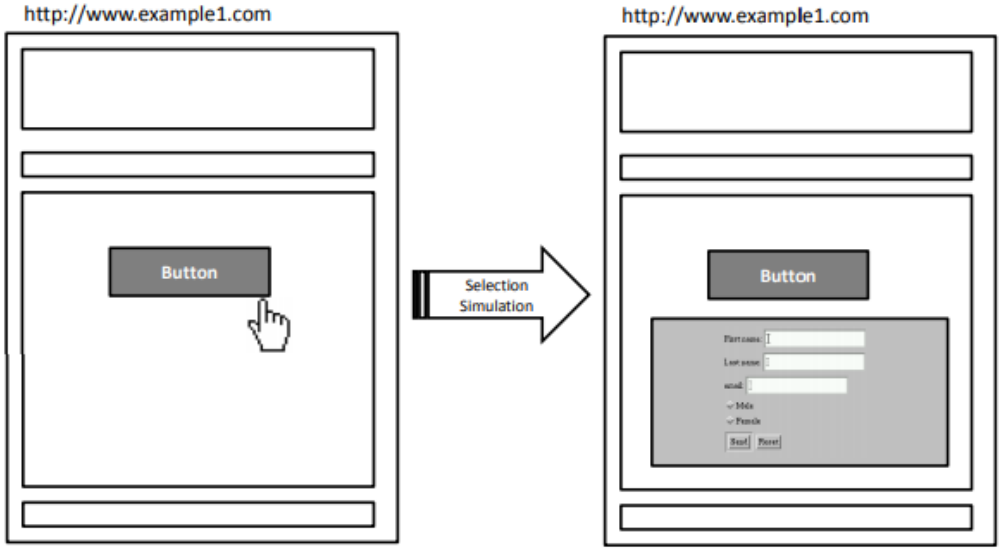

Figura 3.3: Esempio di generazione di un nuovo stato

Per avere questi risultati sono stati utilizzati un Web Processing Simulator ed un Interaction Simulator: il primo, dato un URI (risorsa generica come una pagina web) controlla che ci siano degli stati, in caso positivo passa il corrispondente DOM (struttura del documento web) al secondo, il quale simula tutti i possibili stati in cui la pagina potrà trovarsi.
In seguito sono state fatte tre analisi in momenti diversi (Figura 3.4):

- Analisi effettuata (E1) prima che il web browser processi la pagina (pagina caricata, ma non ancora visibile all'utente), con entrambi i simulatori spenti (Browser Processing e Interaction)
- Analisi effettuata (E2) dopo che il web browser ha processato la pagina (pagina visibile all'utente), con solo il simulatore Browser Processing acceso
- Analisi effettuata (E3) dopo che il web browser ha processato la pagina e con entrambi i simulatori accesi

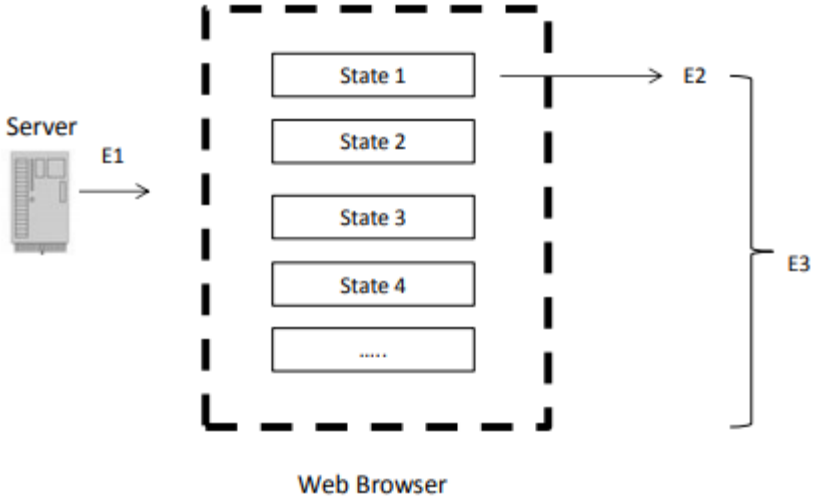

Figura 3.4: Rappresentazione dei tre tipi di valutazione

In questo modo sono stati analizzati più di 8000 siti web 2.0 (RIA) e utilizzando queste analisi si è riscontrato che gli strumenti di valutazione classici trascurano il 92% dei possibili stati che può avere una pagina, di conseguenza i problemi di accessibilità che possono avvenire in quegli stati sono ignorati.

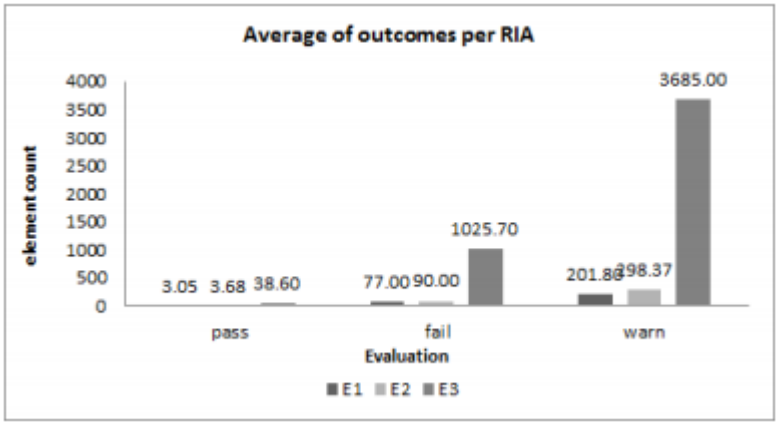

Figura 3.5: Media dei risultati per ogni sito basati sulle regole del W3C

Dalla figura 3.5 è possibile notare come infatti, analizzando meno stati, ci si trova di fronte ad un'enorme mancanza di dati non presi in considerazione sia nella fase E1, dove viene analizzata la pagina ancor prima dell'analisi del web browser, che in E2, nella quale la pagina è processata dal web browser, ma senza analizzare ulteriori stati.

Concludendo il discorso sulla progettazione di un sito web, la comunità scientifica ha trovato numerosi spunti, alcuni dei quali li ho raccolti qui per arrivare alla conclusione che per sviluppare un sito web accessibile sono necessari diversi passi e conoscenze, ma in ultimo luogo anche un buono strumento di verifica dell'accessibilità combinato con dei test reali del sito fatti attraverso persone con disabilità, per unire i risultati ottenendo più accuratezza e dettaglio.

3.2: Strumenti open-source e commerciali

Esistono diversi strumenti per analizzare i contenuti web e verificare che siano raggiungibili da tutti, ma la necessità principale è quella di potersi fidare del fatto che vengano realmente controllati tutti gli elementi della pagina.
Gli strumenti utili alla verifica di un sito sono essenzialmente di due categorie:

- *Checker*: strumenti principali usati dagli sviluppatori per rilevare automaticamente i problemi del sito

- *Lettori di schermo*: solitamente utilizzato da persone con disabilità visiva per poter navigare e visitare i siti, può però essere un metodo per lo sviluppatore per immedesimarsi nelle situazioni che un utente disabile vive normalmente, per capire meglio come avviene la lettura della pagina dal loro punto di vista

Partendo dai checker, come spiegato precedentemente, si è costretti a fare i conti con un limite che è quello di non poter prendere automaticamente una decisione su certe tipologie di situazioni, come affermare con certezza se un link o il testo alternativo di un'immagine siano chiari o meno.
Quindi è importante dare allo sviluppatore la possibilità di interagire con i risultati del test.

Uno di questi strumenti si chiama **AChecker**[5] [GQ10], è open-source, le API del servizio sono disponibili sul sito Internet corrispondente ed è attualmente disponibile come estensione per browser[6].
Questo checker permette di interagire con i risultati prodotti da una valutazione, dando la possibilità di risolvere le questioni non decidibili dal valutatore autonomamente e inoltre visualizza allo sviluppatore che cosa è stato controllato offrendo loro strumenti per aggiungere o modificare alcuni controlli di accessibilità o linee guida.

[5] Il tool è disponibile all'indirizzo: https://achecker.ca/checker/index.php.

[6] Un'estensione del browser è un'aggiunta o plug-in che estende le funzionalità del browser stesso. Attraverso l'autorizzazione dell'utente utilizza tecnologie web come HTML, JavaScript e CSS.

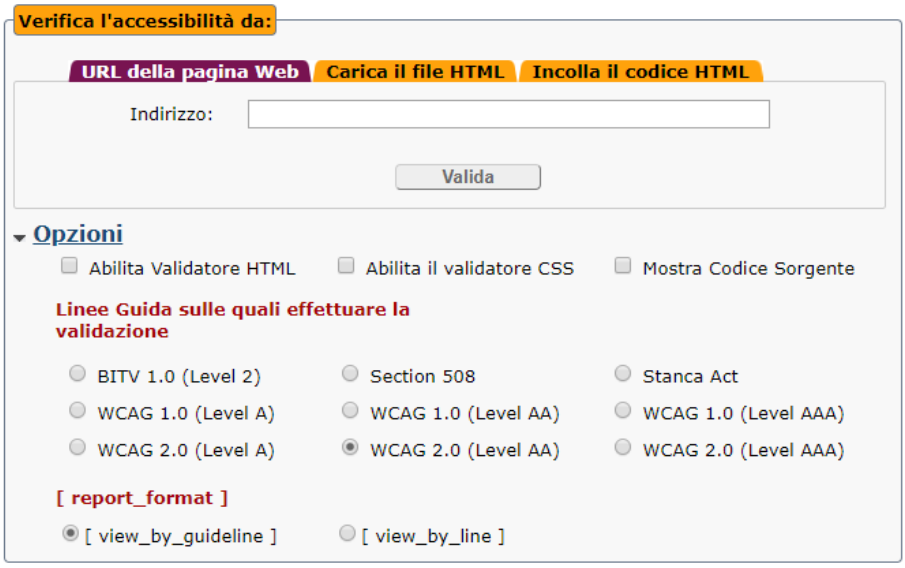

Figura 3.6: Maschera di inserimento della pagina da controllare con opzioni sottostanti

Come si può vedere dall'immagine sopra, è possibile inserire la pagina da verificare in vari modi, dopodiché si passa facoltativamente alla selezione delle linee guida da utilizzare e di altre opzioni utili.
Una volta effettuata la validazione verrà presentata una schermata contenente titolo, spiegazione e codice interessato, il tutto organizzato con diversi tab, dove i più importanti sono quelli che distinguono gli errori per certezza della valutazione, dividendo il tutto in errori certi, errori probabili ed errori potenziali.
Sfortunatamente però questo strumento, pur essendo open source, non riceve aggiornamenti da parecchio tempo, l'interfaccia è ormai datata e l'estensione per browser è un semplice collegamento al sito ufficiale del validatore.
Oltre a questo, manca una possibilità di poter vedere rapidamente le parti inesatte, infatti la validazione produce un output privo di collegamenti al codice del sito web per una rapida modifica.
Questo primo esempio credo sia però utile a far capire al lettore quali siano le funzioni basilari di uno strumento di valutazione essendo AChecker tra i primi strumenti resi disponibili.

Google stessa ha progettato e messo in commercio dal 2017 un suo strumento di validazione automatica denominato **Lighthouse**[7], anch'esso open source e presente nel Chrome Web Store come estensione, permette come AChecker di

[7] Estensione Chrome attivabile attraverso il Chrome Web Store e documentazione disponibile all'indirizzo https://developers.google.com/web/tools/lighthouse.

visualizzare la lista di tutti i problemi della pagina con le spiegazioni degli errori e come fare per risolverli.
Anche qui avviene una suddivisione per problemi risolvibili e problemi che necessitano dell'intervento manuale dello sviluppatore per essere valutati (Figura 3.7), ma c'è un vantaggio rispetto ad AChecker perché grazie alle pagine di supporto Google, è possibile visualizzare una documentazione dettagliata del problema attraverso un link che porta ad esse.
Inoltre riporta un punteggio generale espresso in centesimi al termine della valutazione, in modo che sia più semplice capire l'attuale accessibilità del proprio sito ed è inoltre possibile scegliere se valutare il sito in versione desktop oppure in versione mobile.

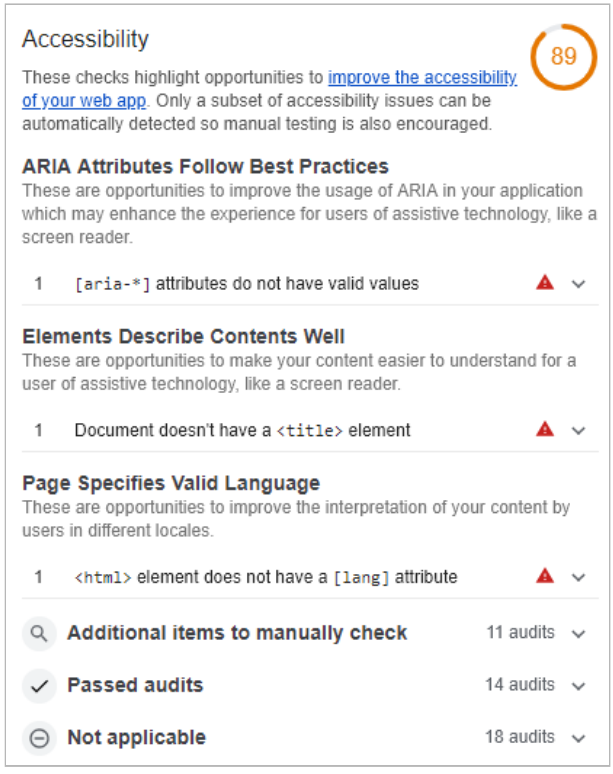

Figura 3.7: Esempio di scansione di Lighthouse

Quello che però rende tutto più comodo e semplice, è il fatto che non è necessario spostarsi da una scheda all'altra del browser per visualizzare i risultati dell'analisi perché il software è presente direttamente nel pannello sviluppatore di Chrome (premendo F12) a fianco della pagina, il che rende molto più agevole il controllo dato che premendo sul codice che il valutatore ha evidenziato, si viene reindirizzati al tab del pannello sviluppatore che visualizza gli elementi HTML per evidenziare l'elemento oggetto dell'analisi.

Questo tool permette di fare anche altre analisi al sito al di fuori della accessibilità e soprattutto è largamente supportato dalla comunità e da Google stessa e questo è un valore molto grande per uno strumento che opera nel campo delle tecnologie web dove le novità si susseguono molto rapidamente.
Rispetto ad AChecker quindi ci si trova di fronte ad uno strumento molto più a spasso con i tempi e in grado di effettuare analisi accurate e mantenendo una grande flessibilità di utilizzo, tuttavia non permette di effettuare quelle scelte fini come l'inclusione di una linea guida WCAG rispetto ad un altra e risulta quindi poco trasparente.

In ultimo, **WAVE**[8], rappresenta forse lo strumento più valido e completo da diversi punti di vista, racchiudendo svariate caratteristiche che lo rendono ottimo per questa funzione:

- È un'estensione per browser e, come Lighthouse, si posiziona a fianco della pagina web per un comodo accesso
- Come in precedenza vengono segnalati gli errori (in rosso) che non soddisfano le linee guida, questa volta potendo applicare un filtro (in alto in figura 3.9) sulle linee guida e inoltre ogni errore o avvertimento (in giallo) ha la sua icona specifica in modo che sia facilmente riconoscibile

[8] La documentazione è disponibile sulla pagina ufficiale: https://wave.webaim.org.

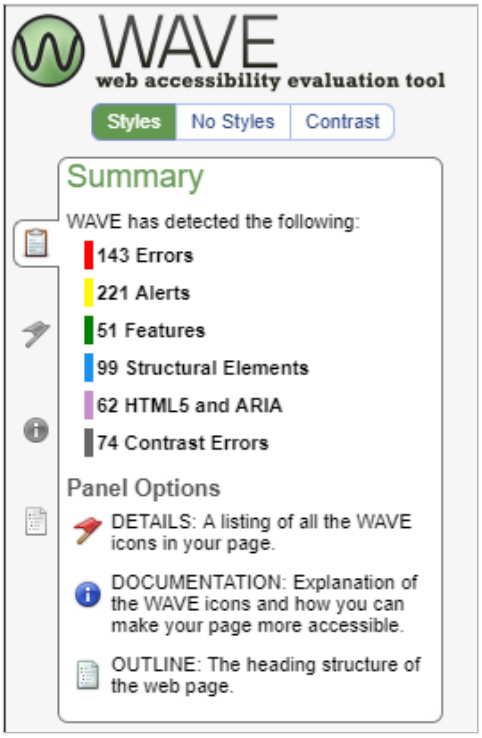

Figura 3.8: visualizzazione generale della valutazione

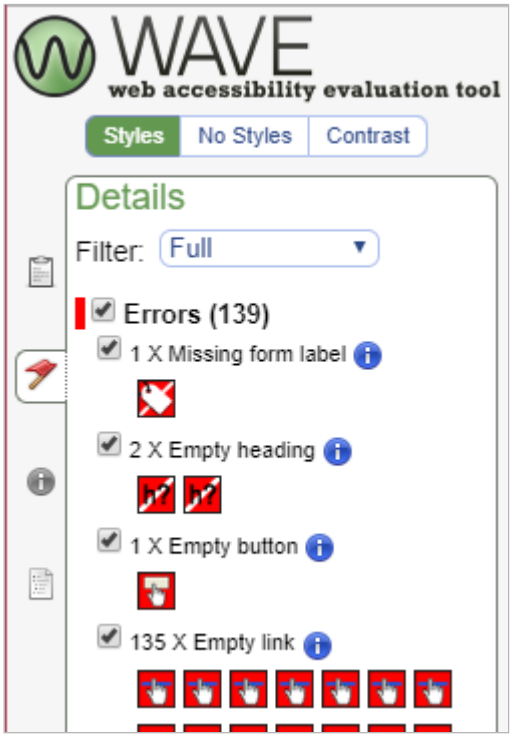

Figura 3.9: visualizzazione specifica degli errori

- L'icona personalizzata per ogni problema diventa ancora più utile qui perché WAVE aggiunge queste icone all'interno della pagina web e sono quindi visualizzabili immediatamente a colpo d'occhio potendo scegliere quali elementi visualizzare e quali no, inoltre viene evidenziato il codice ARIA scritto correttamente (in verde) a fianco degli oggetti corrispondenti

Figura 3.10: Visualizzazione delle icone di errore, avvertimento e metadati ARIA all'interno della pagina

- Consente anche di disabilitare gli stili della pagina che si sta analizzando in modo da rendere visibili tutte le icone di errore che potrebbero non esserlo se il contenuto è nascosto via CSS (foglio di stile), oppure evidenziare gli errori che riguardano il contrasto dei colori per disabilità cromatiche

WAVE è sicuramente tra i checker più ricchi di funzionalità sul mercato, tuttavia permangono i limiti discussi in precedenza degli analizzatori automatici cioè che non assicurano che un sito sia accessibile se non presenta errori, semplicemente considerando le linee guida non sono state trovate incongruenze, ma questo non fuga i dubbi che ci possano essere problematiche se il sito viene sottoposto all'utilizzo di un vero utente non vedente.

Analizzando invece i sintetizzatori vocali utilizzati come strumento di verifica dell'accessibilità anziché come metodo di lettura per disabili, ho avuto modo di testare ChromeVox[9], estensione di Chrome, con la quale ho potuto rendermi conto di come sia più complessa e macchinosa la navigazione all'interno di una pagina web se si è privi di vista. Questo sintetizzatore vocale è comunque simile ad altri, ma credo sia importante far notare questa opportunità di verifica dell'accessibilità di un sito web.

L'analisi degli strumenti e dei risultati evidenziati dalla comunità scientifica è stata fondamentale per comprendere quali siano i punti forti e le pecche delle tecnologie e dei metodi di sviluppo disponibili ad oggi, questo mi ha permesso di sviluppare uno strumento che può essere utile in determinate circostanze, laddove le modalità attuali mancano o non sono sufficienti.

[9] Estensione scaricabile all'indirizzo:
https://chrome.google.com/webstore/detail/chromevox/kgejglhpjiefppelpmljglcjbhoiplfn?hl=it.

Capitolo 4

SahARIAn

Lo strumento di validazione dell'accessibilità che propongo in questa dissertazione è strutturato in maniera differente da quelli descritti precedentemente e opera in modo altrettanto opposto, offrendo un punto di vista nuovo dal quale osservare e prendere spunto per migliorare l'accessibilità dei siti web.

SahARIAn, come Lighthouse e WAVE, è un'estensione per Chrome e Firefox e per realizzarla è stato necessario consultare il manuale fornito da Google[10], nel quale è spiegato passo passo cosa sono, come funzionano e come è possibile sviluppare una.
Ancora non rilasciata, SahARIAn è in via di sviluppo e, a differenza degli strumenti precedenti, non è un validatore automatico non occupandosi di verificare la correttezza del codice rispetto alle linee guida.
Il suo compito principale è quello di rappresentare visivamente le mancanze della notazione ARIA commesse durante la scrittura della pagina, modificandone la visualizzazione e trasformando le interazioni da mouse in input da tastiera, simulando così l'interazione da parte di un utente non vedente.
La rappresentazione offerta da SahARIAn, rende di facile lettura un possibile errore di design della struttura della pagina rispetto ai requisiti ARIA grazie alla modifica della visualizzazione dell'interfaccia del sito web e al cambiamento delle interazioni da mouse a tastiera.
Queste manipolazioni danno modo allo sviluppatore di essere più a contatto con la reale struttura della pagina web, rendendosi conto di come potrebbe essere interpretata da una persona non vedente.
Le modalità con le quali queste modifiche avvengono saranno descritte più

[10] https://developer.chrome.com/extensions.

precisamente nel capitolo successivo.
L'utilizzo è semplice: una volta installato, sarà possibile accedervi tramite la barra degli strumenti del browser cliccando sull'icona corrispondente, la quale aprirà il pop-up[11] (Figura 4.1) di comando di SahARIAn, dal quale si può scegliere tra le due modalità di visualizzazione disponibili:

- *Visualizzazione Min*: altera la visualizzazione della pagina originale rimpiazzando tutte le immagini con una di default, racchiudendo il contenuto in una cornice grigia e presentando l'interfaccia in maniera semplificata, ponendo enfasi sulla struttura e non sulla presentazione di essa.

- *Visualizzazione Max*: esegue le stesse modifiche all'interfaccia della modalità Min, ma in aggiunta evidenzia in rosso le zone più critiche a livello di accessibilità che presentano pesanti mancanze di notazioni ARIA.

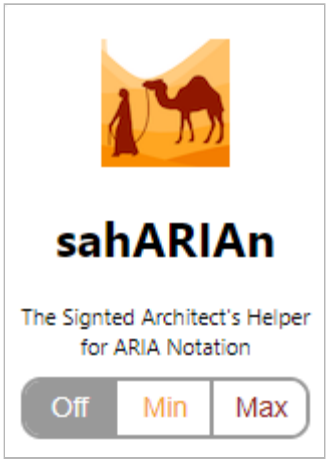

Figura 4.1: Finestra di comando di SahARIAn

Il funzionamento tecnico di queste modalità e il come è stato possibile effettuare le operazioni di cambio di stili e trasformazione degli input, verranno descritti in maniera più ampia nel capitolo successivo.
Ora mi focalizzo di più sulle conseguenze che sviluppa questo tipo di approccio e come è stato possibile attuarlo.

Per svilupparla al meglio e provarne l'utilità ho creato una demo (Figura 4.2) di un sito web che mi ha permesso di valutare l'utilizzo di questo plugin e

[11] Finestra che si apre sullo schermo del computer durante la navigazione in Internet.: "un pop-up pubblicitario".

confrontarlo con gli strumenti descritti in precedenza.
Di seguito analizzerò qualche esempio per mostrare più esplicitamente il funzionamento e i casi che si possono verificare durante l'analisi dell'accessibilità di un sito.
L'esempio da me realizzato rappresenta un ipotetico servizio di ricerca di ristoranti, organizzati per città, dove per ogni ristorante è presente un form nel quale è possibile effettuare diverse operazioni fittizie come prenotare o scegliere le pietanze dal menu.
Per sviluppare il più correttamente possibile i fogli di stile che SahARIAn applicherà una volta attivato, ho scritto questa demo inserendo quasi la totalità dei ruoli, proprietà e stati che ARIA introduce nella sua documentazione, così da poter scrivere regole corrette per molte situazioni che possono verificarsi e per essere sicuro di usare nella maniera giusta le regole ARIA mi sono servito anche degli esempi disponibili sul sito ufficiale del W3C[12], adattandoli al mio utilizzo e modificandoli dove necessario.
Questa è stata una parte abbastanza complessa e laboriosa perché ha richiesto molta attenzione nel cercare di non commettere errori, non essendo io un esperto nello sviluppo di siti web accessibili utilizzando la notazione ARIA, ma affacciandomi al tema per la prima volta.

Ho effettuato alcuni casi d'uso utilizzando la demo sviluppata e altri siti, riporto quindi i risultati confrontandoli fra loro.
In questo primo esempio, analizzo il caso in cui la pagina web è già abbastanza ricca di attributi ARIA che la rendono ben accessibile in molte sue parti:

[12] https://www.w3.org/TR/wai-aria-practices/examples.

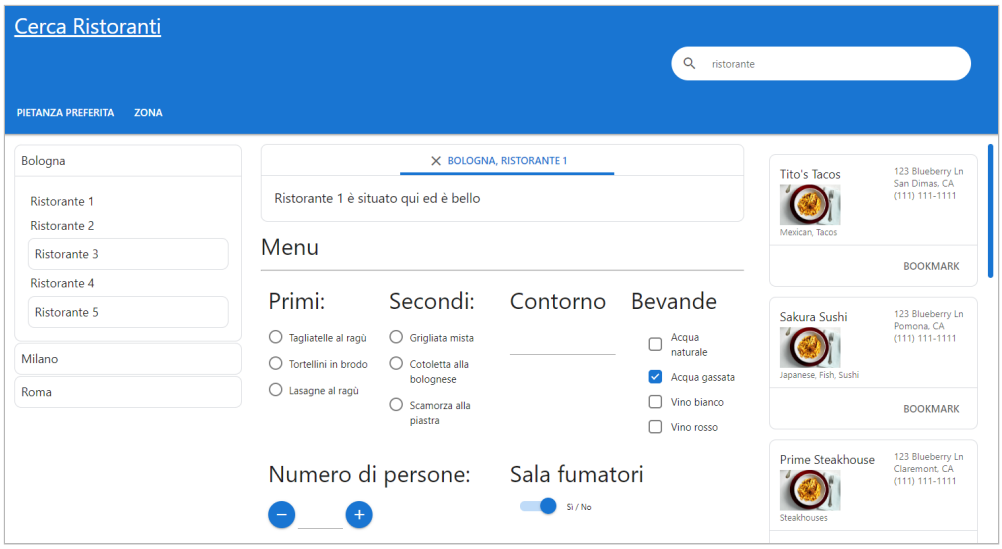

Figura 4.2: Schermata della demo con SahARIAn disabilitato

In figura 4.2 è presente il sito web così come l'ho creato, a questo punto attivando dall'apposita finestra (Figura 4.1) la modalità Min, la pagina subirà un caricamento dopo il quale verrà presentata spoglia dell'iniziale foglio di stile, che come detto in precedenza è sostituito da quello di SahARIAn.
A questo punto si può già avere una chiara idea di come la mia pagina web sia ben accessibile o meno da come viene rappresentata in figura 4.3, dove ovviamente i colori e l'accuratezza grafica sono spariti, avendo creato uno stile spoglio per evidenziare il più possibile le mancanze ARIA, ma la struttura della pagina precedente è ben riconoscibile grazie al fatto che è ricca di metadati ARIA.
Infatti se attivando SahARIAn ci si ritrova davanti ad una pagina "spoglia", ma "ordinata", questo è l'indice che si è fatto un buon lavoro nel giusto utilizzo delle tre caratteristiche di ARIA (ruoli, proprietà e stati), combinandole nella maniera più opportuna.

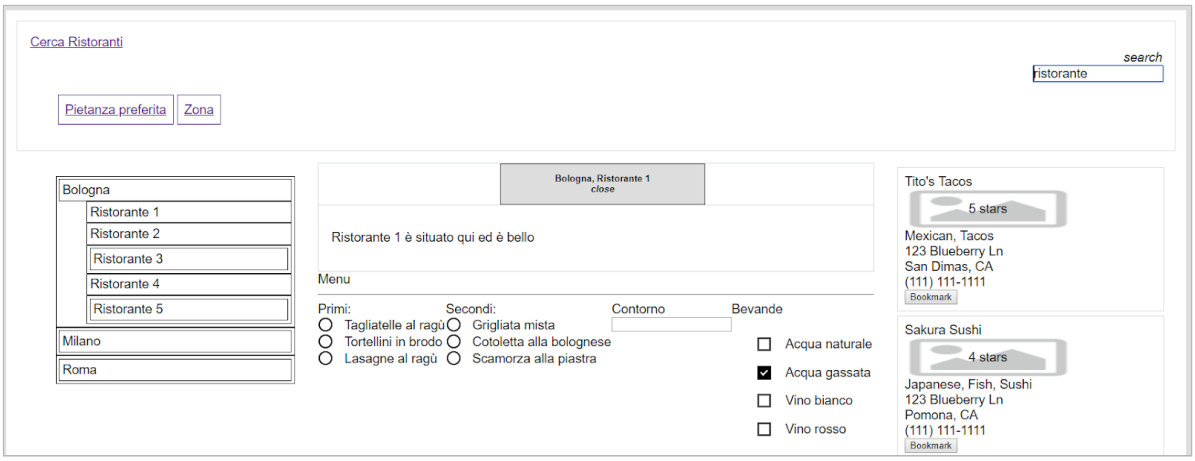

Figura 4.3: Schermata della demo con SahARIAn attivato in modalità Min

Dopo aver provato la demo in modalità Min, ho voluto testare ancora più nel profondo il sito utilizzando la modalità Max (Figura 4.4), la quale ha evidenziato che pur essendo ben accessibile, presenta comunque alcune problematiche da sistemare.
Ho voluto effettuare allora un test utilizzando Lighthouse per aver un risultato numerico dell'accessibilità e la validazione ha prodotto un punteggio di 89/100, confermando quindi che ci fossero diverse cose da mettere a punto, per esempio alcune proprietà ARIA di certi elementi non erano coerenti, ma che tutto sommato, visto il punteggio e visto il risultato della prova in modalità Min, il sito da me realizzato è stato ben costruito.

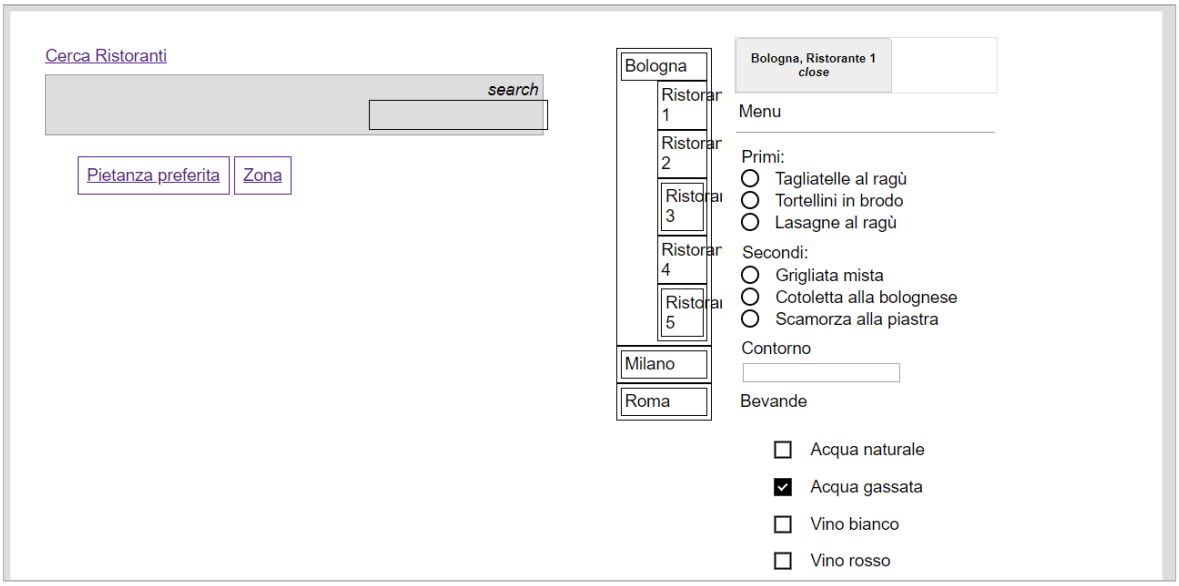

Figura 4.4: Schermata della demo con SahARIAn attivato in modalità Max

Ho poi voluto fare una seconda prova utilizzando la stessa demo precedente, alla quale però ho volutamente omesso o posto erratamente alcuni attributi ARIA, scambiando per esempio il ruolo *treegrid* con il ruolo *tree* (nel menu laterale sinistro) o eliminando il ruolo feed nell'elenco dei ristoranti a destra, tanto per citarne alcuni.
Il risultato è rappresentato in figura 4.5 dove, al contrario della figura precedente, è evidente come si faccia più fatica a capire la struttura della pagina web o comunque ci si rende conto a colpo d'occhio che qualcosa non funziona come dovrebbe, nonostante gli effettivi contenuti siano i medesimi del caso precedente.
A questo punto, trovandosi di fronte ad una situazione del genere, si capirebbe senza particolari difficoltà che qualcosa andrebbe rivisto.

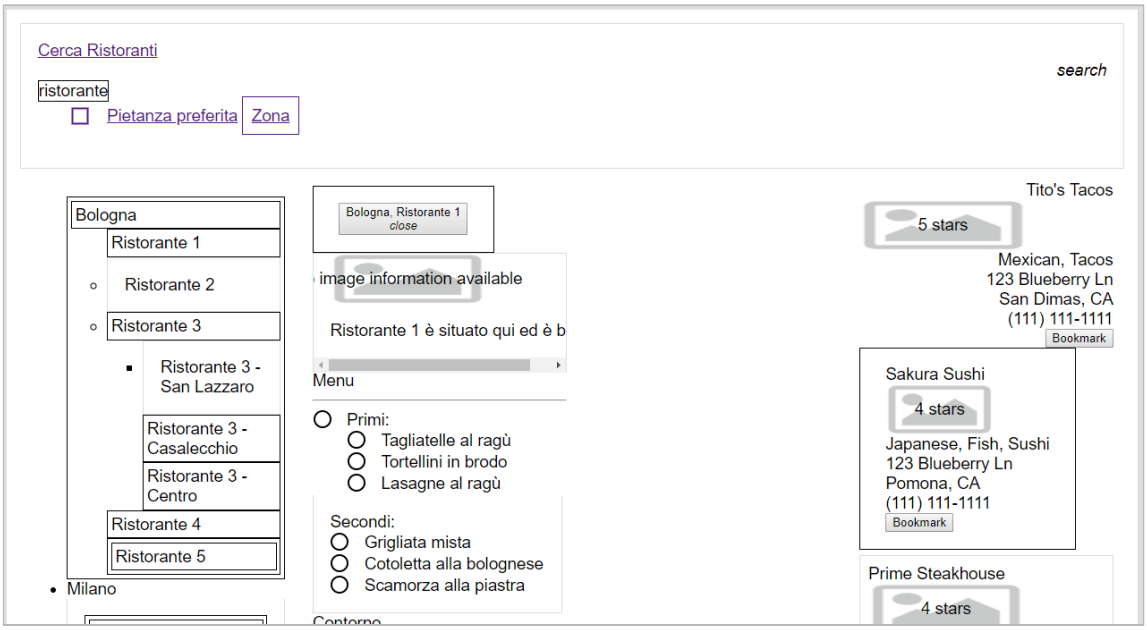

Figura 4.5: Schermata della demo con attributi ARIA e HTML5 errati o mancanti

Come prima, ho voluto provare la modalità Max e il risultato è stato in linea con le aspettative (Figura 4.6), evidenziando in rosso le parti più compromesse e in ogni caso non rendendo facile la comprensione della struttura della pagina.
In particolare, quando una zona è evidenziata di rosso, significa che per il primo elemento figlio del body, contenitore della pagina, non è stato usato nessun ruolo landmark[13] di ARIA che specifichi meglio il suo significato

[13] I landmark ARIA sono attributi aggiuntivi per gli elementi della pagina utili a definire aree specifiche, importanti per una navigazione rapida da parte di utenti non vedenti.

all'interno della pagina e che quindi possa essere interpretato da un sintetizzatore vocale: per esempio invece di usare il tag <main> o il ruolo main, utilizzato per identificare il corpo principale del documento, si utilizza che è un contenitore generico, non aggiungendo altro; questo crea un problema importante ad un utente non vedente che leggendo la pagina, non ha modo di rendersi conto che sta leggendo il corpo principale del documento se non magari attraverso intuizioni, ma certo non è la via migliore.
Essendo questo un aspetto molto importante dell'accessibilità ad un documento, questa mancanza è sottolineata in modo particolare.

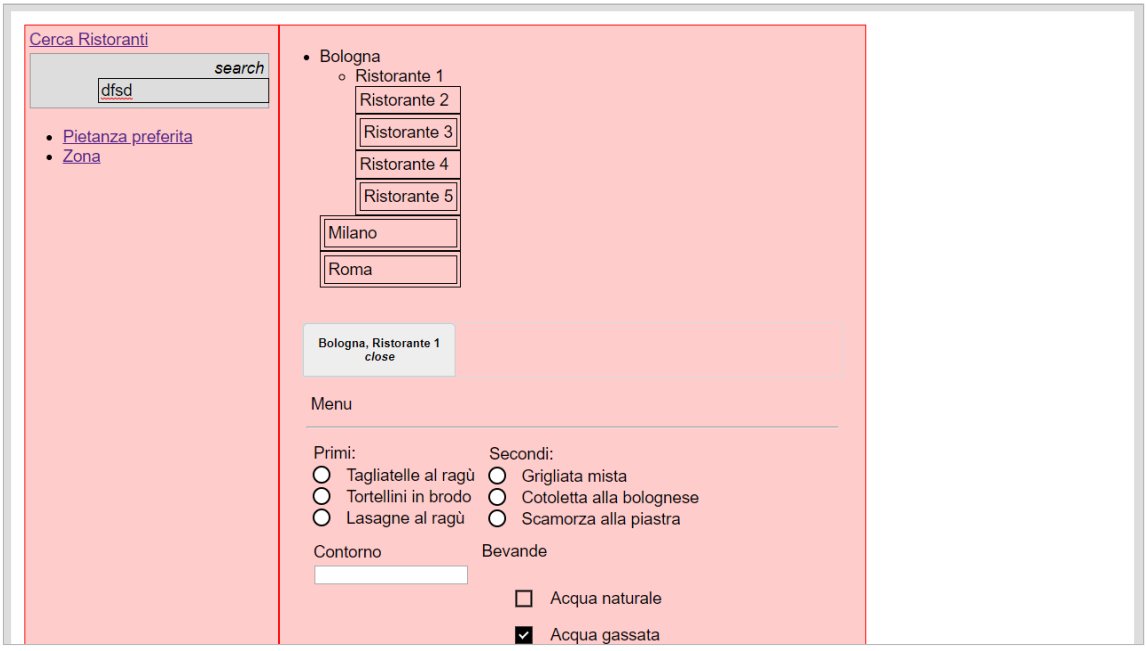

Figura 4.6: Schermata della demo con attributi ARIA e HTML5 errati o mancanti

Oltre alla demo da me realizzata, come dicevo, ho potuto provare SahARIAn su altre pagine web per verificarne diversi comportamenti e capire come sviluppare al meglio questo strumento.

Per portare un secondo esempio, ho scelto la pagina di HDblog.it, sito che frequento diverse volte e mi incuriosiva conoscerne l'accessibilità.

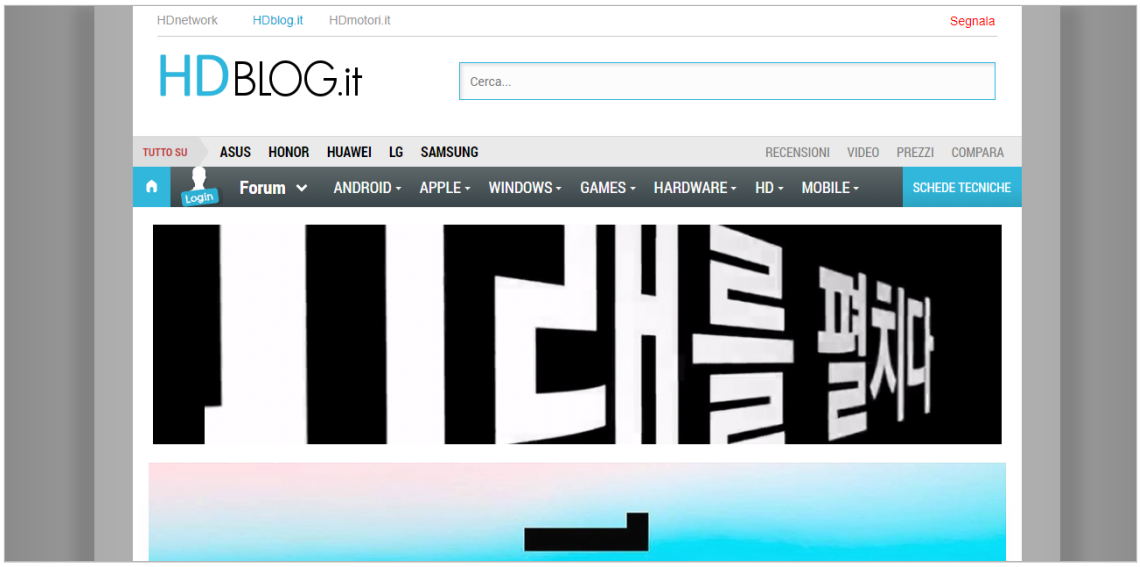

Figura 4.7: Home page di HDblog.it

A questo punto svolgo lo stesso procedimento fatto in precedenza.

Ma il risultato è nettamente peggiore come si può vedere in figura 4.8, il layout risulta errato e non c'è modo di riconoscere una traccia del sito originario.

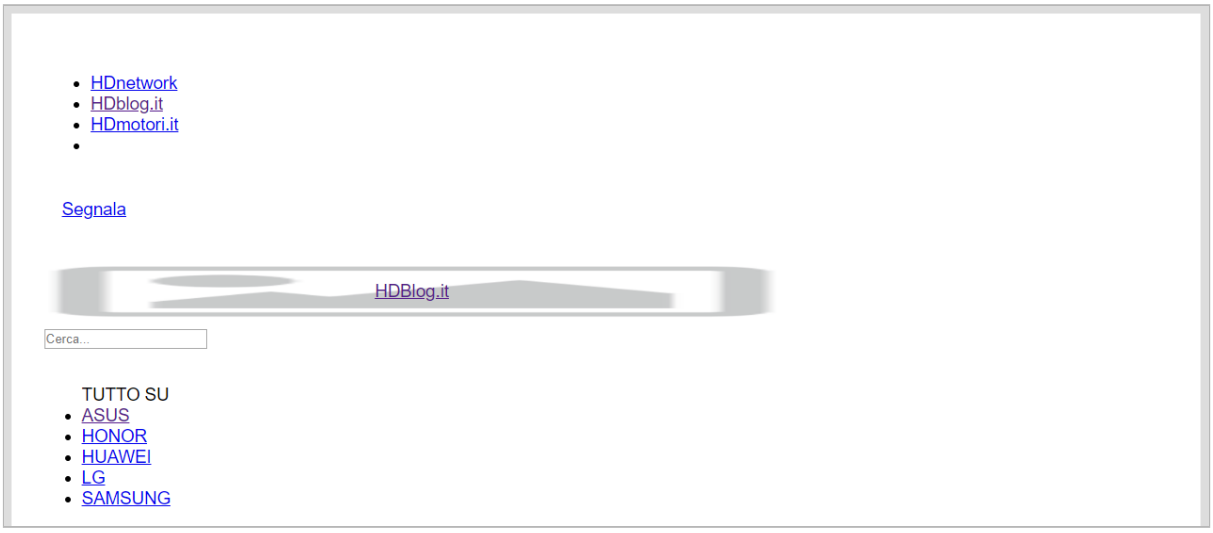

Figura 4.8: Home page di HDblog.it con SahARIAn attivato in modalità Min

Provando la modalità Max, la situazione peggiora ulteriormente come si può notare nella figura 4.9.

Figura 4.9: Home page di HDblog.it con SahARIAn attivato in modalità Max

A seguito di questi risultati abbastanza negativi ho quindi voluto fare una prova verificando HDblog con uno strumento di valutazione automatica.
Per comodità ho scelto Lighthouse per avere immediatamente un risultato numerico che valutasse l'accessibilità e, come mi aspettavo, il risultato è stato di 47/100, mentre la mia demo raggiunge gli 89 punti su 100.
È quindi possibile fare un paragone tra i metodi di valutazione, misurando l'efficacia di un sito attraverso entrambe le tecnologie per avere sicuramente un'idea ancora più ricca di quale sia la reale accessibilità di un sito web.

In realtà ho testato SahARIAn anche su numerosi altri siti, ma i risultati ottenuti sono simili a quelli appena descritti, con variazioni da pagina a pagina in relazione al livello di accessibilità e alla notazione ARIA utilizzata.
Quello che però posso dire in più, è che capita spesso di imbattersi in situazioni di scarsa implementazione delle linee guida ARIA, nonostante siano ormai passati diversi anni dalla sua introduzione.

Capitolo 5

Architettura di SahARIAn

In questo capitolo descriverò più precisamente la realizzazione del mio progetto di tesi, discutendo delle scelte implementative effettuate.
SahARIAn è come detto un'estensione che genera un documento HTML autonomo e autosufficiente che, attraverso la tecnica del Code injection, permette un'interazione con la pagina designata che si vuole valutare, con la quale è possibile dialogare scambiando dati con essa attraverso l'utilizzo dell'interfaccia a pop-up (Figura 4.1).

Code injection Letteralmente "iniezione di codice", è una particolare tecnica di programmazione che consiste nell'inserire una parte di codice all'interno di un altro software per influenzarne il funzionamento.
Solitamente si tratta di un procedimento che assume una connotazione negativa, in quanto è facile vederne i pericoli che possono scaturirsi dall'uso scorretto, soprattutto per quel che riguarda la sicurezza su Internet.
È per esempio possibile che la Code injection venga utilizzata per alterare il contenuto di un database, facendo perdere una grande quantità di dati, oppure per installare codice maligno all'interno di un server o attaccare utenti di Internet attraverso HTML/Script injection[14].
Questo metodo, ovviamente, non è solo uno strumento per cracker con intenzioni maligne, ma può essere impiegato anche per buone cause, per esempio introducendo funzionalità utili che in precedenza non erano disponibili.
SahARIAn rientra quindi nell'ultimo caso e, nella fattispecie, carica una serie di fogli di stile personalizzati per le notazioni ARIA che vengono aggiunti al DOM della pagina ospite.
I fogli di stile CSS sono caricati attraverso l'iniezione del file *saharian.js*

[14] https://en.wikipedia.org/wiki/Code_injection.

all'interno della pagina web visualizzata al momento dell'attivazione dell'estensione, Il plug-in è composto di due parti principali:

- *Pop-up*: interfaccia attraverso la quale è possibile scegliere quale modalità di visualizzazione utilizzare (Figura 4.1). Non è costituito da parti particolarmente interessanti tali da essere descritte più in dettaglio.

- *File saharian.js*: file iniettato all'interno della pagina web visualizzata al momento dell'attivazione dell'estensione. Questo file viene quindi eseguito nel sito ospite e permette di effettuare operazioni attraverso la finestra a pop-up suddetta. Di seguito descrivo più precisamente quali operazioni vengono svolte.

saharian.js Al suo interno è definita la funzione principale del file che riceve come input un parametro booleano, che, a seconda del suo valore, fa sì che SahARIAn attivi la visualizzazione desiderata o meno.

function activate(force)

Si possono distinguere quindi i due casi: *force* è vero oppure è falso. Nel caso sia vero verranno eseguite diverse operazioni che elenco qui di seguito:

- *Disattivazione delle immagini*: in questa parte vengono rimpiazzate le immagini con una di default (Figura 4.8, dietro alla scritta HDblog) e questa procedura viene eseguita ogni mezzo secondo, per assicurarsi che eventuali immagini o pubblicità successive al momento di attivazione di SahARIAn vengano sostituite.

- *Disattivazione degli stili*: tutti gli stili all'interno dei tag del codice HTML vengono disattivati, ad eccezione degli attributi display e visibility che vengono invece mantenuti, la motivazione sta nel fatto che questi due attributi non rappresentano una possibile problematica di accessibilità, in quanto la scelta di rendere invisibile un elemento è di carattere implementativo e non un modo per nascondere problemi strutturali della pagina. Questo processo viene eseguito ogni mezzo secondo per evitare che qualche stile possa essere reimpostato dal codice della pagina da analizzare nel corso del tempo.

- *Attivazione di un bypass di eventi (Inject Event Bypass)*: a questo punto si procede con la trasformazione di tutti gli eventi provenienti dal mouse,

in eventi da tastiera. Questo serve per dare la possibilità ad uno sviluppatore di usare comunque un mouse per comodità, ma ,attraverso questo bypass, può anche rendersi conto se un evento da tastiera non viene gestito correttamente, infatti, se mal gestito, l'evento non verrà eseguito.

- *Disattivazione dei fogli di stile*: la funzione legge e genera delle copie dei fogli di stile appartenenti alla pagina web, per poi disattivarne le regole, anche in questo caso con l'eccezione di display e visibility.

- *Aggiunta dei fogli di stile specializzati*: vengono aggiunti i fogli di stile di entrambe le modalità (Min e Max) attivando solo il foglio Min, nel caso si attivi la medesima modalità, oppure entrambe, nel caso si selezioni la modalità Max.

Nel caso opposto, dove *force* è falsa, avviene la disattivazione di tutti i processi appena descritti, se precedentemente erano stati attivati, in maniera contraria a come sono stati elencati, procedendo quindi con la rimozione dei fogli di stile specializzati, per poi riattivare i fogli di stile originali, disattivare il bypass degli eventi provocati dal mouse, rendere di nuovo operativi gli stili all'interno dei tag HTML e rimpiazzare le immagini disattivate in precedenza, in modo da far ritornare la pagina al suo stato iniziale.
La funzione esegue tutte queste operazioni attraverso una IIFE[15], la quale restituisce un oggetto formato da due elementi importanti:

- *status*: closure che restituisce *status*, variabile definita dalla IIFE, composta da tre valori che identificano rispettivamente il livello attuale di visualizzazione (off, Min o Max), la necessità di rimpiazzare le immagini della pagina e il bisogno di disattivare gli stili.

- *activate*: funzione asincrona che gestisce le chiamate delle funzioni di attivazione e disattivazione per i vari livelli di SahARIAn. Il cambio di visualizzazione avviene infatti in maniera asincrona, in quanto una volta attivata l'estensione in modalità Min o Max, per tutte le pagine visitate successivamente, verranno effettuate le modifiche sopracitate.

saharian.js esegue, non appena la pagina viene caricata, la funzione activate in maniera corrispondente al valore selezionato attraverso il pop-up: se per

[15] Immediately-invoked Function Expression: Funzione anonima creata ed immediatamente invocata. Serve per creare oggetti non ripetibili dotati di closure, quindi con stato interno non visibile all'esterno.

esempio SahARIAn è disattivato, verrà invocata activate con parametro *force* falso, dopodiché nel caso venga attivato, un listener[16] di Chrome dichiarato dopo la activate(), si occuperà di invocarla nuovamente con valore vero, in modo da far attivare la catena di eventi descritti.

Passando invece ad analizzare il codice CSS che viene aggiunto ai fogli di stile, disattivati, della pagina una volta che si attiva l'estensione è necessario distinguerne le funzioni:

- *Min*: contiene tutti i ruoli, proprietà e stati presenti all'interno della demo realizzata (Figura 4.2) e di conseguenza le regole sono quasi sempre del tipo [role="radio"][aria-checked="true"]{ *regole* }
- *Max*: le sue regole si attivano in aggiunte a quelle di Min, ma evidenziando in maniera marcata, attraverso una colorazione rossa, gli errori sui ruoli di landmark di ARIA.

Per poter installare SahARIAn, non essendo ancora disponibile sullo store di Chrome, è necessario, una volta recati al pannello delle estensioni del browser, attivare la modalità sviluppatore che permette un'installazione manuale delle estensioni, quindi selezionare la cartella contenente tutti i file del plug-in e ultimare la procedura.

[16] Elemento di programmazione che utilizza un osservatore per controllare gli eventi generati da un oggetto, in modo da effettuare operazioni al verificarsi di un certo evento: https://it.wikipedia.org/wiki/Observer_pattern.

Capitolo 6

Conclusioni e sviluppi futuri

Al termine di questa dissertazione ho ben compreso come l'accessibilità sia un tema centrale di questo momento storico, è parte fondamentale di Internet ed è un obiettivo da raggiungere per dare la possibilità a tutti di usufruire degli stessi servizi al fine di abbattere, almeno per quanto riguarda Internet, le barriere attualmente presenti sul web.

Discutere di quali siano le attuali linee guida e le problematiche che tentano di risolvere, è stato utile a chiarire quale sia la reale situazione dell'accessibilità nel mondo in termini di strumenti ad oggi disponibili, scoprendo anche che le ultime tecnologie hanno portato miglioramenti, ma anche ulteriori problemi.

A questo punto è stato possibile sviluppare una ricerca basata sulle attuali modalità di progettazione di pagine web accessibili, studiate e discusse nelle conferenze scientifiche più importanti del settore, per cercare di carpire le informazioni più utili in modo da poter colmare i buchi vuoti lasciati dalle tecnologie attualmente disponibili sul mercato.

Questo mi ha permesso di sviluppare al meglio SahARIAn, lo strumento utilizzato in questa tesi per sostenere il fatto che una rappresentazione visiva delle notazione di accessibilità, possa essere utile al fine di agevolare gli sviluppatori nella creazione di pagine più accessibili.
Il suo sviluppo è stato per me una nuova esperienza e un lavoro interessante, soprattutto nella fase di creazione della demo per poter testare SahARIAn, questo mi ha permesso di capire più a fondo quali siano le difficoltà che incontra una persona con difficoltà visiva e, allo stesso tempo, quali siano le sfide più ardue con le quali gli sviluppatori di siti web hanno a che fare ogni volta che progettano un sito web.

Al termine di questo sviluppo posso dire con certezza che il sistema realizzato consente una facile ed immediata analisi dell'accessibilità di un sito web.

Concludo questo elaborato analizzando i limiti di SahARIAn che pecca in termini di consigli e assistenza in fase di sviluppo e sistemazione degli errori fatti.
Com'è infatti evidente dal funzionamento, non genera nessun elenco di errori o possibili mancanze, come invece fanno i metodi automatici e questo è chiaramente un suo limite che non lo farebbe preferire in maniera decisa a metodi già esistenti, ma magari solamente come un qualcosa di aggiuntivo o in fase iniziale, per controllare immediatamente e in maniera più rapida l'accessibilità del sito, oppure in fase finale, per verificare che il risultato positivo di uno strumento automatico sia effettivamente supportato anche da una visualizzazione coerente data da SahARIAn.

Quello che mi propongo di aggiungere allora è proprio questo, dare allo sviluppatore un'opportunità di visione delle parti che possono essere migliorate aggiungendo consigli e modalità su come migliorarle, in particolare avendo uno stile simile a WAVE, con segnalazione attraverso icone disposte nei punti critici della pagina web e magari unire anche la comodità e praticità della valutazione offerta da Lighthouse, con i veloci link alle risorse utili per capire di che errore si tratta e quali possono essere le soluzioni migliori per risolverli.
In secondo luogo, sarebbe utile sviluppare una funzionalità di SahARIAn che permetta di scegliere tra le modalità di lettura della pagina offerte dai sintetizzatori vocali, descritte nell'introduzione. Attraverso un selettore della modalità document e di quella application, l'estensione darebbe allo sviluppatore un ulteriore strumento sicuramente utile a testare la navigazione e l'interazione che un utente può svolgere all'interno del sito, in modo da capire se possono esserci problemi su quel lato una volta risolte le mancanze a livello strutturale.
Un altro punto da migliorare sarà la demo, necessaria per indirizzare lo sviluppo dell'estensione in maniera corretta, aggiungendo i ruoli rimasti esclusi in questa fase e dedicando più tempo anche a proprietà e stati, correggendo gli errori evidenziati dagli strumenti automatici e da SahARIAn nel capitolo 4.
Questa demo una volta strutturata al meglio potrà anche essere un punto di

riferimento dal quale prendere spunto per avere un'idea di com'è strutturata e composta una pagina accessibile.
Inoltre ci sono ancora alcuni errori da sistemare per migliorare le modalità di visualizzazione e fare sì che queste possano essere ancora più accurate nella rappresentazione visuale della pagina web, in modo poi da rendere SahARIAn più affidabile e utile.

Bibliografia

- [DC10] Dell Anhol Almeida L, Calani Baranauskas MC. 2010. Universal Design Principles Combined with Web Accessibility Guidelines: A Case Study. IHC '10 Proceedings of the IX Symposium on Human Factors in Computing Systems
- [CF17] Conway V, Fitzpatrick K. 2017. Creating Accessible Local Government: The Process. W4A 2017 conference
- [DPS04] Di Blas N, Paolini P, Speroni M. 2004. "Usable Accessibility" to the Web for blind Users. Conference paper
- [FBC13] Fernandes N, Batista AS, Costa D, Duarte C, Carriço L. 2013. Three Web Accessibility Evaluation Perspectives for RIA. W4A 2013 conference
- [GQ10] Gay G, Qi Li C. 2010. AChecker: Open, Interactive, Customizable, Web Accessibility Checking. Conference Paper
- [HGB09] Hailpern J, Guarino Reid L, Boardman R, Annam S. 2009. WEB 2.0: Blind to an Accessible New World. WWW'09 Madrid conference
- [LBO08] Leuthold S, Bargas-Avila JA, Opwis K. 2008. Beyond web content accessibility guidelines: Design of enhanced text user interfaces for blind Internet users. Department of Psychology, University of Basel paper
- [PPP12] Power C, Pimenta Freire A, Petrie H, Swallow D. 2012. Guidelines are Only Half of the Story: Accessibility Problems Encountered by Blind Users on the Web. CHI 2012 conference, May 5–10, 2012

Risorse online:

- Dirty Work - Web e accessibilità, 2016: http://www.dirtywork.it/blog/accessibilita-siti-web
- MDN Web Docs - Basi della tecnologia WAI-ARIA, 2018: https://developer.mozilla.org/it/docs/Learn/Accessibilit%C3%A0/WAI-ARIA_basics
- W3C - Index of ARIA Design Pattern Examples, 2019: https://www.w3.org/TR/wai-aria-practices/examples
- AChecker, 2011: https://achecker.ca/checker/index.php
- Google Developers - Lighthouse, 2018: https://developers.google.com/web/tools/lighthouse/
- WAVE, 2019: https://wave.webaim.org/
- Chrome Web Store - ChromeVox, 2019: https://chrome.google.com/webstore/detail/chromevox/kgejglhpjiefppelpmljglcjbhoiplfn?hl=it
- Chrome Developer - Extensions: https://developer.chrome.com/extensions.
- Wikipedia. Code injection - Wikipedia, enciclopedia libera, 2019: https://en.wikipedia.org/wiki/Code_injection
- Wikipedia. Observer Pattern - Wikipedia, enciclopedia libera, 2018: https://it.wikipedia.org/wiki/Observer_pattern

Ringraziamenti

Ringrazio il mio relatore Fabio Vitali per avermi dato la possibilità di lavorare ad un progetto importante e stimolante come questo.

Ringrazio i miei genitori che creano sempre le premesse per far sì che possa esprimermi al meglio in ciò che faccio, sono orgoglioso di loro

Ringrazio la Nonna per tutto quello che mi dà ogni giorno tra storie, racconti di ogni tipo e tante tante tagliatelle al ragù, perché il pranzo dalla nonna è unico

Ringrazio la mia fidanzata, Cristina, per essermi sempre vicino dandomi i consigli giusti e infondendomi la tranquillità necessaria per fare meglio, questo è molto importante per me

Ringrazio i miei amici di sempre, Francesco, Antonio e Tom per tutto quello che hanno fatto per me nel corso di questi tanti anni e per tutti i bei momenti passati assieme tra risate, PlayStation e mangiate, siete insostituibili

Ringrazio Osso, Cala e Drus per le belle serate, i tantissimi zigomi alti, le ceffe e l'aiuto che mi hanno dato durante questi tre anni, senza il loro supporto non sarei riuscito a concludere il percorso in questo modo

Ringrazio poi tutte le persone con le quali ho condiviso dei bei momenti, siete veramente in tanti e questo mi fa molto piacere

Vi voglio bene