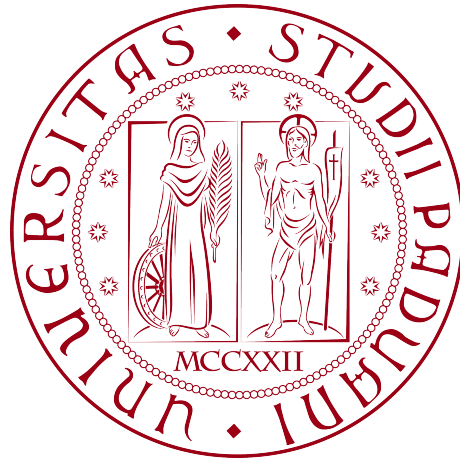


Università degli Studi di Padova

DIPARTIMENTO DI MATEMATICA “TULLIO LEVI-CIVITA”

CORSO DI LAUREA IN INFORMATICA



***SviluppAbile: un'estensione per sviluppare
pagine web accessibili con l'aiuto di un chatbot***

Tesi di Laurea

Relatrice

Prof.ssa Ombretta Gaggi

Laureanda

Elena Chilese

Matricola 2008074

ANNO ACCADEMICO 2024-2025

“Dedicata ad Elena, che mai si è arresa...”

Ringraziamenti

Per prima cosa desidero ringraziare la professoressa Ombretta Gaggi, relatrice della mia tesi e tutor del mio stage, per l'assistenza e il sostegno fornitimi durante tutto il percorso di stage e nella stesura del lavoro.

Un sincero grazie va anche a Salvatore Gatto per il supporto e l'aiuto durante lo stage.

Con profondo affetto ringrazio i miei genitori e, in particolar modo, mia mamma, per la pazienza e la forza con cui mi ha sempre sostenuta anche nei momenti più difficili. Un ringraziamento va anche ai miei zii e soprattutto a mio nonno Adriano, per la vicinanza e l'affetto che non mi ha mai fatto mancare.

Un grazie speciale a Michele, che mi è stato accanto fin dall'inizio del percorso universitario, diventando non solo un prezioso punto di riferimento in ogni occasione, ma anche il mio compagno di avventure e il mio migliore amico.

Ringrazio di cuore tutti i miei amici e i compagni di università (in particolare Matteo, Elena ed Erica) per la loro vicinanza e il sostegno dimostrato in questi anni.

Desidero esprimere la mia gratitudine inoltre ai colleghi e ai professori dell'ITE Fusinieri, che hanno contribuito al mio percorso con il loro supporto e i loro insegnamenti.

Padova, Settembre 2025

Elena Chilesse

Sommario

Il presente documento descrive il lavoro svolto durante il periodo di stage, della durata di trecento ore svolto presso il Dipartimento di Matematica “Tullio Levi-Civita” dell’Università di Padova.

Il prodotto finale dello stage è un’estensione web *Chrome-based_G* per aiutare gli sviluppatori web nella creazione di pagine web accessibili.

Gli obiettivi del progetto sono:

- Studio dell’accessibilità web;
- Studio di Manifest V3;
- Studio delle possibili *API_G* per l’integrazione dell’ *IA_G*;
- Creazione di un’estensione web che integri un *chatbot_G AI_G* per analizzare il codice sorgente della pagina web ed offrire risposte adeguate.

Tabella dei contenuti

1	Introduzione	1
1.1	L'idea del progetto	1
1.2	Analisi delle soluzioni già presenti	3
1.3	Organizzazione del testo	4
2	Analisi dei requisiti	5
2.1	Obiettivo del progetto	5
2.2	Casi d'uso	5
2.3	Tracciamento dei requisiti	5
3	Tecnologie	6
3.1	Linguaggi	6
3.1.1	HTML	6
3.1.2	CSS	7
3.1.3	Prism.js	8
3.1.4	JavaScript	8
3.2	Tecnologie e strumenti	10
3.2.1	Chrome Extension	10
3.2.1.1	Manifest V3	10
3.2.2	Ollama	13
3.2.3	GitHub	15
3.2.4	VSCode	15
3.2.5	Total Validator	16
3.2.6	WAVE	17

TABELLA DEI CONTENUTI

3.2.7	Lighthouse	17
4	Sviluppo del progetto di stage	19
4.1	Web design	19
4.2	Sviluppo	20
4.2.1	PoC	20
4.2.2	Prodotto finale	21
4.3	Interazione con l'AI	21
4.4	Filtraggio risposta generata	24
4.5	Problematiche riscontrate	24
5	Test	25
5.1	Test “Analisi assistita”	25
5.1.1	F1-score	25
5.1.2	Sito web: SudokuWorld	26
5.1.2.1	Total Validator	26
5.1.2.2	Lighthouse	27
5.1.2.3	SviluppAbile	27
5.1.2.4	Resoconto finale	31
5.2	Test “Modalità guidata”	33
6	Conclusioni	39
6.1	Consuntivo finale	39
6.2	Competenze acquisite	39
6.3	Valutazione personale	39
	Bibliografia	ii

Elenco delle figure

3.1	Logo HTML5	7
3.2	Logo CSS3	8
3.3	Logo JavaScript	10
3.4	Logo Ollama	15
3.5	Logo GitHub	15
3.6	Logo VSCode	16
3.7	Logo Total Validator	17
3.8	Logo WAVE	17
3.9	Logo Lighthouse	18
4.1	Logo dell'estensione <i>SviluppAbile</i>	21
5.1	Analisi di Total Validator sul sito web SudokuWorld	26
5.2	Analisi di Lighthouse sul sito web SudokuWorld	27
5.3	Pagina HTML base per effettuare i test della modalità guidata . . .	33

Elenco delle tabelle

3.1	Confronto modelli Ollama utilizzati	14
-----	---	----

Capitolo 1

Introduzione

L'accessibilità web rappresenta un aspetto fondamentale per garantire che tutte le persone, indipendentemente dalle loro abilità o disabilità, possano utilizzare i contenuti digitali in modo efficace. Le linee guida [WCAG_G](#) (*Web Content Accessibility Guidelines*) sono uno standard internazionale, esse definiscono i criteri tecnici per migliorare l'accessibilità dei siti web, intervenendo su aspetti quali la struttura semantica, la navigazione tramite tastiera, il contrasto cromatico dei colori e la compatibilità con tecnologie assistive.

Un sito accessibile non solo migliora l'esperienza d'uso per persone con disabilità visive, motorie o cognitive, ma estende anche la fruibilità a un pubblico più ampio, contribuendo a un web più inclusivo e universale.

Nonostante ciò, spesso la realizzazione pratica di siti accessibili incontra difficoltà dovute a scarsa conoscenza delle best-practices dello sviluppo web accessibile, delle normative in questione e della scarsa diffusione di strumenti automatizzati efficaci.

1.1 L'idea del progetto

Il progetto sviluppato durante il mio stage si inserisce all'interno di un'iniziativa molto più ampia intitolata *“Supporting Accessibility Auditing and HTML Validator using Large Language Models”* a cui partecipano e collaborano docenti e stagisti dell'Università di Padova e dell'Università di Bologna.

Tale progetto nasce dall'esigenza di affrontare il problema dell'accessibilità web,

diritto fondamentale per tutti i cittadini (in particolare per le persone con disabilità) che devono poter accedere alle informazioni sul web senza barriere.

Nonostante le normative nazionali e internazionali (come la *Direttiva Europea sull'Accessibilità Web* e l' *European Accessibility Act*) impongano requisiti chiari, una buona parte dei siti web presenta ancora numerose criticità di accessibilità. [17]

In questo contesto, il progetto mira a valutare, testare e sfruttare le capacità dei *Large Language Models* (LLM_G) per supportare l'audit dell'accessibilità e la validazione del codice $HTML_G$.

L'obiettivo finale è sviluppare strumenti innovativi che utilizzino modelli di *intelligenza artificiale* $_G$, come $ChatGPT_G$, per fornire un supporto interattivo agli sviluppatori, aiutandoli a identificare e correggere problematiche di accessibilità in modo più efficace e comprensibile rispetto ai metodi tradizionali. Attraverso questa integrazione, si intende migliorare la qualità e soprattutto la chiarezza dei risultati e favorire una cultura più diffusa sull'accessibilità digitale.

Inoltre, si vuole anche valutare se gli LLM siano effettivamente in grado di svolgere questo tipo di lavoro.

L'estensione "*SviluppAbile*" è un primo esempio di strumento per lo sviluppo guidato di pagine web accessibili. Essa nasce dall'esigenza di supportare gli sviluppatori web nel creare pagine accessibili in modo semplice e interattivo, sfruttando le potenzialità dell'*intelligenza artificiale* $_G$.

L'estensione *Chrome-based* $_G$ sviluppata offre due modalità principali:

- una modalità di analisi assistita, che permette di ispezionare il DOM_G di una pagina web e di porre domande specifiche sull'accessibilità, ottenendo risposte chiare e contestualizzate;
- una modalità di sviluppo guidato, in cui l'utente riceve suggerimenti di codice accessibile in tempo reale, visualizzati in una colonna centrale e pronti per essere copiati o scaricati.

Questo duplice approccio consente di integrare nel flusso di lavoro quotidiano degli sviluppatori un valido supporto basato su *AI_G*, con l'obiettivo di facilitare il rispetto delle linee guida *WCAG_G* e migliorare la qualità complessiva del codice.

1.2 Analisi delle soluzioni già presenti

Sul mercato esistono diversi strumenti e *plugin_G* dedicati alla verifica dell'accessibilità delle pagine web, come Total Validator [3.2.5], WAVE [3.2.6] e Lighthouse [3.2.7], che offrono principalmente funzionalità di scansione automatica e generazione di report.

Questi strumenti, tuttavia, forniscono spesso una panoramica statica dei problemi riscontrati; indicano chiaramente dove si trova l'errore, ma i risultati sono generalmente formulati in modo formale e destinati ad un esperto di accessibilità. Di conseguenza, l'utente non specialista deve interpretare autonomamente le informazioni, con il rischio di applicare soluzioni non sempre pienamente conformi alle linee guida.

Alcuni software integrano suggerimenti o tutorial, ma raramente utilizzano modelli di *intelligenza artificiale_G* in grado di interagire con lo sviluppatore in linguaggio naturale, rispondendo a domande specifiche o guidandolo direttamente nella scrittura del codice.

“SviluppAbile” si distingue per l'integrazione di una chat *IA_G* che funge da assistente personale, e per la duplice modalità operativa che combina l'analisi in tempo reale con un supporto diretto allo sviluppo, colmando così una lacuna importante nelle soluzioni attualmente disponibili.

1.3 Organizzazione del testo

Il documento è diviso in sei capitoli e illustra in maniera dettagliata l'esperienza di stage svolta.

Il [secondo capitolo](#) illustra le informazioni raccolte in fase di analisi del progetto tramite requisiti e casi d'uso.

Il [terzo capitolo](#) approfondisce le tecnologie e gli strumenti utilizzati per la realizzazione del progetto.

Il [quarto capitolo](#) descrive lo sviluppo dell'estensione, evidenziando le scelte progettuali e le soluzioni implementative.

Il [quinto capitolo](#) presenta i test effettuati, volti a valutare l'effettiva utilità dell'estensione rispetto agli strumenti tradizionali di validazione del codice accessibile.

Il [sesto capitolo](#) riassume i risultati ottenuti e le possibili evoluzioni future del progetto.

Riguardo la stesura del testo, relativamente al documento sono state adottate le seguenti convenzioni tipografiche:

- Gli acronimi, le abbreviazioni e i termini ambigui o di uso non comune menzionati vengono definiti nel glossario, situato alla fine del presente documento;
- Per i termini riportati nel glossario viene utilizzata la seguente nomenclatura: *Application Program Interface_G*;
- I termini in lingua straniera o facenti parti del gergo tecnico sono evidenziati con il carattere *corsivo*;
- Gli esempi di codice, le domande poste all'IA e le righe di codice utilizzano il formato `monospace`.

Capitolo 2

Analisi dei requisiti

2.1 Obiettivo del progetto

L'obiettivo principale del progetto è la realizzazione di un'estensione web interattiva con chatbot integrato, pensata per supportare gli sviluppatori nell'individuazione e nella correzione degli errori di accessibilità presenti nelle pagine web.

Il progetto si propone di rispondere alle seguenti esigenze:

- Fornire un linguaggio chiaro e facilmente comprensibile, anche per chi non ha conoscenze approfondite delle linee guida di accessibilità;
- Offrire un supporto immediato e contestuale durante lo sviluppo del codice;
- Facilitare il miglioramento della qualità e dell'accessibilità dei siti web, promuovendo buone pratiche di progettazione inclusiva.

2.2 Casi d'uso

2.3 Tracciamento dei requisiti

Capitolo 3

Tecnologie

3.1 Linguaggi

3.1.1 HTML

HTML_G (*HyperText Markup Language*) è il linguaggio di marcatura standard utilizzato per strutturare i contenuti delle pagine web. Non si tratta di un linguaggio di programmazione, ma di uno strumento che consente di definire gli elementi presenti su una pagina (come titoli, testi, immagini, link, *form_G*) assegnando loro un significato semantico tramite l'uso di tag.

Introdotta nel 1991 da Tim Berners-Lee, *HTML_G* si è evoluta nel tempo attraverso diverse versioni, fino a diventare uno standard globale supervisionato dal *World Wide Web Consortium* (*W3C_G*).

La struttura di un file *HTML_G* si basa su una serie di elementi, rappresentati attraverso tag, che permettono di definire la natura e la funzione dei contenuti all'interno di una pagina web. Ogni tag descrive una tipologia specifica di contenuto o un comportamento associato, contribuendo all'organizzazione semantica del documento. I tag sono delimitati da parentesi angolari (< >) e, nella maggior parte dei casi, sono utilizzati in coppie: un tag di apertura e uno di chiusura, quest'ultimo contraddistinto da una barra (</tag>). Alcuni elementi, tuttavia, non necessitano di chiusura e sono detti auto-chiudenti, come ad esempio per l'inserimento di immagini, o
 per i ritorni a capo. Questa struttura gerarchica

e nidificata consente al *browser_G* di interpretare correttamente i contenuti, garantendo coerenza tra presentazione e significato.

La versione attuale, HTML5 (logo in figura 3.1), è stata rilasciata in modo stabile nel 2014.

HTML5 rappresenta un'evoluzione del linguaggio di *markup_G* con l'obiettivo di ridurre la dipendenza da *plugin_G* esterni, introducendo nuove funzionalità native. Tra le principali novità vi sono gli elementi `<canvas>`, `<video>` e `<audio>`, nuovi tag semantici per strutturare i contenuti (come articoli, intestazioni, sezioni) e controlli avanzati per i *form_G* (email, *URL_G*, numeri, date, ricerca).

Vengono inoltre introdotti strumenti per la memorizzazione locale dei dati sul client tramite `localStorage` e `sessionStorage`.



Figura 3.1: Logo HTML5

3.1.2 CSS

CSS_G (*Cascading Style Sheets*) è un linguaggio utilizzato per definire la presentazione dei documenti *HTML_G* e *XML_G*. Introdotto nel 1996 da Håkon Wium Lie, consente di separare la struttura dei contenuti dalla loro formattazione, migliorando la chiarezza del codice e facilitandone la manutenzione.

I fogli di stile permettono di specificare, tramite regole composte da selettori, proprietà e valori, l'aspetto degli elementi *HTML_G*: layout, colori, font, margini, spaziature, effetti visivi, animazioni e molto altro. Le regole di stile possono essere inline (tramite tag `<style>` o con attributi `style`) oppure riportate in file esterni `.css` riutilizzabili. La prima soluzione non mantiene però una netta separazione tra

struttura e contenuto, non rispettando quindi le best-practice del settore.

L'attuale versione, CSS3 (logo in figura 3.2), è modulare e introduce importanti funzionalità come `flexbox`, `grid`, media query per il responsive design, transizioni, trasformazioni e nuovi selettori.



Figura 3.2: Logo CSS3

3.1.3 Prism.js

Prism.js è una libreria JavaScript leggera ed estensibile progettata per evidenziare la sintassi del codice sorgente in modo chiaro e leggibile, rispettando gli standard moderni del web.

Supporta un'ampia gamma di linguaggi di programmazione e di *markup_G*, ed è facilmente personalizzabile tramite temi e *plugin_G*.

Grazie alla sua efficienza e semplicità di integrazione, viene utilizzata in milioni di siti web per migliorare la leggibilità del codice, rendendolo più comprensibile sia a sviluppatori che a utenti. E' disponibile in diverse varianti, sia per il white-mode che per il dark-mode.

Nella mia estensione, Prism.js è stato integrato per colorare il codice sorgente del *DOM_G*, facilitando l'analisi e la comprensione della struttura *HTML_G*.

3.1.4 JavaScript

JavaScript (logo in figura 3.3) è un linguaggio di programmazione dinamico, interpretato ed orientato agli oggetti, progettato originariamente per rendere le pagine web interattive.

È stato sviluppato nel 1995 da Brendan Eich per Netscape con il nome LiveScript, e

successivamente rinominato in JavaScript per motivi di marketing, pur non avendo legami diretti con il linguaggio Java. La sua standardizzazione è gestita da *ECMA International*, sotto il nome ECMAScript.

Concepito per essere eseguito direttamente all'interno del *browser_G*, JavaScript consente di scrivere *script_G* incorporati nel codice *HTML_G*, eseguibili senza compilazione al momento del caricamento della pagina. Questi *script_G* permettono di manipolare il *DOM_G* (Document Object Model), reagire agli eventi dell'utente (come click, input da tastiera o movimenti del mouse), modificare dinamicamente il contenuto della pagina e gestire funzionalità come validazioni, messaggi, animazioni, e interazioni *asincrone_G*.

Oltre alla manipolazione di elementi visivi, JavaScript permette di sfruttare numerose *API_G* messe a disposizione dal *browser_G*, dalla gestione del mouse e del touch, alla manipolazione delle immagini, fino alla gestione locale dei dati attraverso meccanismi come il LocalStorage. Questa versatilità lo rende uno strumento fondamentale per lo sviluppo di applicazioni web moderne.

JavaScript si è evoluto da semplice linguaggio *client-side_G* a tecnologia completa e versatile, oggi utilizzabile anche su *server_G* (ad esempio con *Node.js*) e in ambienti esterni ai *browser_G* grazie all'uso di diversi motori JavaScript i quali permettono l'esecuzione di codice *JS_G* con alte prestazioni, rendendo il linguaggio adatto anche a contesti *backend_G*, applicazioni desktop e mobile.

Oggi JavaScript rappresenta uno dei tre pilastri fondamentali del web, insieme a *HTML_G* e *CSS_G*, ed è completamente integrato con essi. Esistono anche linguaggi alternativi (come TypeScript o CoffeeScript) che vengono compilati in JavaScript, offrendo funzionalità aggiuntive mantenendo la compatibilità con l'ambiente JavaScript standard.



Figura 3.3: Logo JavaScript

3.2 Tecnologie e strumenti

3.2.1 Chrome Extension

Le estensioni di Chrome sono piccoli programmi software che consentono di arricchire e personalizzare il *browser*_G, aggiungendo funzionalità aggiuntive o migliorando quelle già esistenti.

Grazie ad esse è possibile modificare l'interfaccia utente, automatizzare attività ripetitive, integrare servizi esterni e rendere più efficiente la navigazione sul web. La distribuzione delle estensioni avviene principalmente tramite il *Chrome Web Store*, ma durante le fasi di sviluppo è possibile installarle manualmente utilizzando la modalità sviluppatore disponibile nel *browser*_G.

3.2.1.1 Manifest V3

Manifest V3 (*MV3*_G) è la specifica più recente per la creazione di estensioni per Chrome e altri *browser*_G basati su *Chromium*.

Rispetto alla precedente Manifest V2, l'ultima versione introduce diverse modifiche per migliorare la sicurezza, le prestazioni e la privacy delle estensioni. [5]

Questa nuova versione è una risposta alle criticità emerse con il Manifest V2, e introduce cambiamenti strutturali significativi nel modo in cui le estensioni interagiscono con il *browser*_G e le pagine web.

E' importante notare che Manifest V3 non è retrocompatibile con le estensioni V2 senza modifiche sostanziali e che la versione è tuttora in evoluzione, poiché alcune

funzionalità sono ancora in fase di implementazione.

Il file `manifest.json` in *MV3_G* continua a rappresentare il cuore dell'estensione, definendo metadati fondamentali come nome, versione, permessi richiesti e *script_G* da eseguire. Tuttavia, rispetto a *MV2_G*, *MV3_G* impone restrizioni più rigide su quali *API_G* possono essere utilizzate e introduce un modello di esecuzione più efficiente e sicuro.

Gli elementi principali di un'estensione basata su Manifest V3 sono:

- **Manifest:** come nelle versioni precedenti, il file `manifest.json` contiene tutte le informazioni necessarie per il funzionamento dell'estensione, inclusi i permessi richiesti. Tra i permessi più rilevanti vi sono:
 - `downloads`: permette all'estensione di avviare e gestire download di file, ad esempio per salvare contenuti generati o analizzati;
 - `scripting`: consente l'esecuzione di *script_G* personalizzati nelle pagine web, utile per analizzare o modificare il *DOM_G*;
 - `activeTab`: garantisce l'accesso temporaneo alla scheda attiva dopo un'interazione dell'utente con l'estensione;
 - `tabs`: fornisce informazioni sulle schede del *browser_G* e permette di interagire con esse, come ottenere *URL_G* o titoli;
 - `storage`: consente di memorizzare e recuperare dati locali, ad esempio impostazioni o risultati di analisi.
- **Service Worker:** in Manifest V3, i tradizionali background *script_G* sono sostituiti da service worker. Questi sono *script_G* che vengono eseguiti in background ma solo quando necessario, rispondendo a eventi come le richieste di rete o azioni dell'utente. A differenza dei background *script_G* di *MV2_G*, i service worker non sono persistenti e vengono sospesi quando inattivi, riducendo così l'utilizzo di risorse e migliorando le prestazioni complessive. I

service worker non possono accedere direttamente al *DOM_G* delle pagine, ma comunicano con i content *script_G* tramite messaggi.

- **Content Script:** vengono iniettati direttamente nelle pagine web e possono interagire con il *DOM_G*. In *MV3_G*, anche i content *script_G* sono soggetti a restrizioni più severe per evitare conflitti con il contenuto della pagina e migliorare la sicurezza.

Il Manifest V3 introduce modifiche significative nell'ecosistema delle estensioni per *browser_G*, con effetti sia positivi sia problematici.

Aspetti positivi

Tra i vantaggi più evidenti vi è il blocco del codice remoto, che impedisce alle estensioni di scaricare ed eseguire codice non verificato dopo l'installazione. Questa misura contribuisce a ridurre alcuni rischi di sicurezza, proteggendo gli utenti da esecuzioni di codice potenzialmente dannoso.

Aspetti critici

Il nuovo standard non elimina completamente i rischi: le *API_G* continuano a consentire la raccolta di dati sensibili (come la cronologia di navigazione) lasciando alcune vulnerabilità inalterate.

Dal punto di vista dello sviluppo, *MV3_G* comporta limitazioni importanti: l'*API_G* *webRequest*, utilizzata per modificare dinamicamente le richieste di rete, è stata sostituita da *declarativeNetRequest*, più rigida e meno flessibile, riducendo le possibilità di innovazione.

Inoltre, le pagine background persistenti vengono sostituite dai service worker, che presentano vincoli sulle *API_G* disponibili e problemi di compatibilità con funzioni avanzate (come *WebSockets_G*, *WebAssembly_G* o la localizzazione delle estensioni). La migrazione a *MV3_G* si è dimostrata complessa e molte funzionalità di estensioni pre-esistenti rischiano di rompersi o di subire un peggioramento delle prestazioni.

Infine, il processo decisionale di Google è stato percepito come scarsamente collaborativo, poiché l'implementazione del nuovo standard è stata imposta senza un reale dialogo con la comunità degli sviluppatori. [6]

In sintesi, sebbene il blocco del codice remoto rappresenti un progresso in termini di sicurezza, Manifest V3 introduce restrizioni che limitano innovazione, privacy e performance, portando l'*Electronic Frontier Foundation* a definirlo un atto ostile verso gli utenti, frutto di una posizione dominante di Google nell'ecosistema delle estensioni web.

3.2.2 Ollama

Ollama (logo in figura 3.4) è una piattaforma che consente di eseguire localmente modelli di *intelligenza artificiale*_G avanzati, sviluppati in collaborazione con OpenAI e altri partner, mantenendo alte prestazioni anche senza l'uso del *cloud computing*_G. Oltre ai modelli gpt-oss-20B e gpt-oss-120B, progettati rispettivamente per scenari a bassa *latenza*_G o per applicazioni generali ad alto livello di ragionamento, Ollama supporta numerosi altri modelli *open weight*_G, tra cui Mistral (utilizzato solo all'inizio a causa delle ridotte capacità del mio pc personale), Llama 3.1:8B e Llama 3.2:3B, che ho utilizzato nel mio progetto. Nella tabella sottostante 3.1 vengono messi a confronto i vari modelli di Ollama utilizzati nei test dell'estensione *SviluppAbile*, sulla base di parametri quali la capacità di ragionamento complesso, la comprensione linguistica, la conoscenza generale, la velocità di risposta e i requisiti hardware.

Tabella 3.1: Confronto modelli Ollama utilizzati

Versione	Llama 3.1:8b	Llama 3.2:3b	Mistral
Ragionamento complesso	Molto elevato, ottimo su coding, Q&A, compiti multilingue complessi	Buono ma inferiore ai modelli più grandi, migliore per compiti rapidi	Limitato rispetto a Llama 8b, pensato per compiti specifici
Comprensione linguistica	Alta precisione e coerenza, anche in testi lunghi	Discreta, ma può generare errori grammaticali o essere meno coerente in testi lunghi	Buona solo in testi brevi e semplici
Conoscenza generale	Ampia copertura di argomenti	Più ridotta per via del minor numero di parametri	Limitata
Velocità di risposta	Più lenta su hardware modesto	Rapida con HW adeguato	Molto rapida
Requisiti hardware	Almeno 8 GB RAM + richiede <i>GPU_C</i> di fascia media/alta	Almeno 4 GB RAM	Almeno 2 GB RAM

La piattaforma integra funzionalità avanzate (function calling, navigazione web integrata, esecuzione di codice Python, output strutturati), accesso completo al processo di ragionamento del modello, regolazione dello sforzo computazionale e possibilità di fine-tuning per personalizzare le prestazioni. Grazie al supporto nativo per la *quantizzazione_C* in formato MXFP4, Ollama riesce a ridurre drasticamente il consumo di memoria, permettendo l'esecuzione di modelli molto grandi anche su sistemi con risorse limitate. La licenza Apache 2.0 garantisce libertà di utilizzo e personalizzazione, mentre l'ottimizzazione per *GPU_C* NVIDIA GeForce RTX e RTX PRO assicura alte prestazioni in locale. .



Figura 3.4: Logo Ollama

3.2.3 GitHub

GitHub (logo in figura 3.5) è una piattaforma di hosting basata su *Git*, il sistema di controllo di versione distribuito ideato da Linus Torvalds. Lanciata nel 2008, GitHub si è affermata come uno degli strumenti principali per la collaborazione nello sviluppo software, consentendo a sviluppatori di tutto il mondo di condividere, modificare e mantenere progetti in modo coordinato.

La piattaforma offre funzionalità avanzate per il versionamento del codice, la gestione dei rami (branching), il tracciamento delle modifiche e la revisione collaborativa tramite pull request.

Oltre a essere uno strumento tecnico, GitHub ha favorito la nascita di una vera e propria comunità *open source*, nella quale il codice è accessibile, riutilizzabile e migliorabile da chiunque.

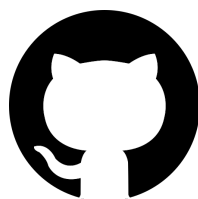


Figura 3.5: Logo GitHub

3.2.4 VSCode

Visual Studio Code (logo in figura 3.6), conosciuto semplicemente come VSCode, è un editor di codice sorgente sviluppato da Microsoft e distribuito gratuitamente. Introdotto nel 2015, ha rapidamente conquistato una posizione di rilievo tra gli

strumenti preferiti dagli sviluppatori grazie alla sua combinazione di leggerezza, flessibilità e ricchezza di funzionalità.

Compatibile con i principali sistemi operativi (Windows, macOS e Linux), VSCode supporta evidenziazione della sintassi per numerosi linguaggi, suggerimenti intelligenti e completamento automatico del codice, strumenti di *debugging* integrati e gestione del controllo di versione tramite *Git*.

Inoltre, grazie a un vasto marketplace di estensioni, può essere facilmente personalizzato per adattarsi a diversi flussi di lavoro e tecnologie, rendendolo un ambiente di sviluppo versatile e ampiamente diffuso.



Figura 3.6: Logo VSCode

3.2.5 Total Validator

Total Validator (logo in figura 3.7) è uno strumento di convalida web che permette di verificare contemporaneamente l'accessibilità secondo le linee guida WCAG 2.0, 2.1 e 2.2, la conformità alla Section 508 statunitense e alle specifiche ARIA.

Consente inoltre di controllare la validità del codice HTML rispetto a numerosi standard, incluso l'HTML5, e la correttezza dei CSS in conformità con molte specifiche del W3C.

Tra le sue funzioni principali rientrano anche la rilevazione di link interrotti e la verifica ortografica multilingue. Lo strumento permette di analizzare pagine offline, protette da autenticazione o generate dinamicamente tramite JavaScript, risultando così particolarmente flessibile.

È disponibile per Windows, macOS e Linux e, dal suo lancio nel 2005, è stato adottato da milioni di siti web, comprese istituzioni pubbliche e governative.



Figura 3.7: Logo Total Validator

3.2.6 WAVE

WAVE (*Web Accessibility Evaluation Tool*) è uno strumento sviluppato da WebAIM per supportare la valutazione dell'accessibilità dei siti web (logo in figura 3.8). Consente di individuare automaticamente molte violazioni delle WCAG e, al tempo stesso, facilita l'analisi manuale da parte degli sviluppatori.

È disponibile come servizio online e come estensione per i principali browser, permettendo di testare anche pagine locali o protette. I risultati vengono visualizzati direttamente sulla pagina tramite icone e pannelli che segnalano errori, avvisi e caratteristiche accessibili, fornendo indicazioni utili per le correzioni.

Per contesti più complessi, WAVE offre inoltre un'API che ne consente l'integrazione nei processi di sviluppo e di testing automatizzato.



Figura 3.8: Logo WAVE

3.2.7 Lighthouse

Lighthouse (logo in figura 3.9) è uno strumento open-source sviluppato da Google per valutare la qualità delle pagine web. Esegue controlli automatici su performance, accessibilità, SEO e progressive web app, restituendo report dettagliati con punteggi e suggerimenti pratici.

Si può utilizzare tramite Chrome DevTools, riga di comando (CLI), modulo Node o interfaccia web, consentendo anche audit su pagine protette o locali. Lighthouse quindi supporta gli sviluppatori nel miglioramento della velocità di caricamento, dell'usabilità e della visibilità dei siti.



Figura 3.9: Logo Lighthouse

Capitolo 4

Sviluppo del progetto di stage

4.1 Web design

Il design dell'estensione è stato sviluppato con un approccio *desktop-first_G*, in considerazione del target principale costituito da sviluppatori web che operano prevalentemente su schermi di grandi dimensioni.

Tale scelta consente di privilegiare la chiarezza e l'ampiezza degli spazi di lavoro, garantendo una disposizione ottimale dei pannelli e delle funzionalità principali. Particolare attenzione è stata posta all'accessibilità cromatica, mediante uno studio accurato dei contrasti tra testo, sfondo ed elementi interattivi, al fine di assicurare leggibilità anche in condizioni visive differenti.

È stata inoltre implementata la possibilità di personalizzare l'esperienza visiva tramite un pulsante dedicato alla selezione della modalità giorno/notte, che permette all'utente di adattare l'interfaccia alle proprie preferenze e alle condizioni ambientali di utilizzo.

Nonostante l'approccio iniziale privilegi i dispositivi desktop, il layout rimane responsivo grazie a soluzioni flessibili che mantengono l'usabilità anche su schermi di dimensioni ridotte. Questa scelta assicura un'esperienza coerente ed accessibile in diversi contesti d'uso.

4.2 Sviluppo

4.2.1 PoC

L'obiettivo iniziale era la realizzazione di un *Proof of Concept_G* (*PoC_G*) che permettesse di verificare la fattibilità tecnica dell'estensione proposta. In questa fase preliminare, l'attenzione era rivolta principalmente a due aspetti fondamentali: il recupero del codice sorgente della pagina web e l'integrazione con un'*API_G* di *intelligenza artificiale_G*, necessaria per avviare i primi test di analisi e generazione di suggerimenti.

Il *PoC_G* non teneva conto di aspetti come la scalabilità, l'ottimizzazione delle performance o la gestione di casi d'uso complessi, ma si limitava a dimostrare che le funzionalità di base potessero essere effettivamente implementate.

La struttura iniziale era volutamente semplice in quanto conteneva solamente funzioni basilari per:

- `script.js`: responsabile dell'avvio dell'estensione;
- `analysis.js`: deputato all'estrazione del codice sorgente della pagina web e alla sua messa a disposizione per ulteriori elaborazioni;
- `service_worker.js`: gestisce le richieste verso il modello di *intelligenza artificiale_G*, inviando il codice recuperato e ricevendo in risposta i suggerimenti generati.

Questa versione sperimentale costituiva una base minima ma solida su cui successivamente è stato possibile costruire le funzionalità avanzate, affinare l'interfaccia e integrare logiche più complesse per la gestione dei risultati.

Da notare il fatto che Chrome blocca l'esecuzione di Ollama nell'estensione. Per avviare correttamente l'estensione è quindi necessario aprire una pagina chrome da terminale utilizzando i seguenti comandi per disabilitare alcuni blocchi di chrome

stesso.

```
cd Program Files_Google_Chrome_Application  
chrome.exe -disable-web-security -user-data-dir="C:_temp_chrome"
```

4.2.2 Prodotto finale



Figura 4.1: Logo dell'estensione *SviluppAbile*

4.3 Interazione con l'AI

L'estensione sviluppata prevede un'integrazione diretta con Ollama. Questa scelta garantisce all'utente il pieno controllo sui dati, riducendo i rischi legati alla trasmissione di informazioni sensibili verso servizi esterni e permettendo un utilizzo anche in assenza di connessione Internet stabile. L'interazione con l'*intelligenza artificiale*_G avviene attraverso chiamate HTTP a un endpoint locale, al quale vengono inviati prompt costruiti dinamicamente in base alle esigenze del flusso operativo.

```
1  async function inviaPrompt(prompt) {  
2      const res = await  
3          fetch('http://localhost:11434/api/generate', {  
4              method: 'POST',  
5              headers: { 'Content-Type': 'application/json' },  
6              body: JSON.stringify({  
7                  model: "llama3.1:8b",  
8                  prompt,  
9                  stream: false  
10             })  
11     });
```

```
11 |   return await res.json();
12 | }
```

Listing 4.1: Funzione di interazione con Ollama

Sono stati definiti tre casi d'uso principali per la generazione dei prompt:

- l'invio congiunto del *DOM_G* della pagina e della domanda dell'utente, utile per ricevere spiegazioni e indicazioni sulle righe che verranno poi evidenziate in quanto utili alla comprensione della risposta;

```
1 |   const promptPrincipale =
2 |     'Sei un assistente che analizza codice HTML.
      Rispondi alla domanda in modo chiaro ma
      conciso, usando al massimo 5-6 frasi. ' +
3 |     'Evita ripetizioni o spiegazioni troppo
      generiche. ' +
4 |     'Alla fine della risposta, su una nuova riga,
      scrivi le righe eventualmente utilizzate per
      la risposta nel seguente formato:\n\n' +
5 |     '##RIGHE##\n{"righe":
      [elenco_di_numeri_di_riga]}\n\n' +
6 |     'Codice HTML:\n${codice}\n\nDomanda: ${domanda}';
```

Listing 4.2: Prompt per la generazione di risposta e righe da evidenziare

- l'invio della sola domanda per generare ulteriori domande piu' approfondite da consigliare all'utente;

```
1 |   const promptSuccessiva =
2 |     'Suggerisci 1 o 2 domande piu' specifiche
      sull'accessibilita' o sull'analisi del codice,
      ' +
3 |     'partendo dalla seguente domanda:\n\n' +
```

```
4      'Rispondi solo con il seguente formato JSON:\n\n'
      +
5      '##DOMANDA##\n{ "domande": ["prima domanda",
      "seconda...", "terza..."] }\n\n' +
6      'Domanda iniziale: ${domanda}';
```

Listing 4.3: Prompt per la generazione di domande successive

- l'invio del *DOM_G* insieme alla richiesta di modifica, scenario in cui l'IA produce sia una risposta argomentata sia un blocco di codice pronto per essere inserito nel pannello centrale della pagina nella modalità di sviluppo guidato.

```
1      const promptCodice =
2      'Sei un assistente che aiuta a rendere
      accessibile il codice HTML. ' +
3      'Rispondi in massimo 5-6 frasi chiare e
      tecniche. ' +
4      'Se suggerisci del codice, racchiudilo tra i
      marcatori \'##CODICE##\' come mostrato di
      seguito:\n\n' +
5      '##CODICE##\n<codice HTML da inserire o
      modificare>\n##FINECODICE##\n\n' +
6      'Codice HTML:\n${codice}\n\nDomanda:
      ${domanda}';
```

Listing 4.4: Prompt per la generazione di risposta e codice *HTML_G* accessibile

Questa diversificazione consente di mantenere un approccio modulare e adattabile, rendendo l'assistente in grado di rispondere a necessità differenti con un unico modello sottostante.

4.4 Filtraggio risposta generata

Un aspetto fondamentale del funzionamento dell'estensione riguarda il filtraggio e la rielaborazione della risposta generata dall'*intelligenza artificiale*_G. Le risposte restituite da Ollama, infatti, non vengono mostrate direttamente all'utente, ma sono sottoposte ad un processo di parsing e di pulizia.

In primo luogo, l'estensione distingue le diverse tipologie di output attese: la risposta vera e propria alla domanda inserita, le eventuali domande suggerite e gli eventuali blocchi di codice generati da visualizzare e/o scaricare. A tal fine vengono utilizzati marcatori testuali inseriti nel prompt (ad esempio `##DOMANDE##` o `##CODICE##`), che consentono di individuare con precisione le sezioni rilevanti all'interno della risposta.

Una volta ricevuto l'output, funzioni dedicate come `estraiRigheDaRisposta` ed `estraiDomandeSuggerite` applicano espressioni regolari per isolare le parti utili, scartando elementi ridondanti o formattazioni non necessarie.

Il filtraggio consente anche di separare le informazioni in blocchi distinti, in modo che ciascun contenuto possa essere mostrato nella pagina web nel pannello appropriato (ad esempio, suggerimenti testuali nella chat e codice sorgente nel riquadro centrale). Questo approccio riduce il carico cognitivo per l'utente, che non si trova di fronte a una risposta grezza e complessa, ma ad un output strutturato e facilmente navigabile.

Inoltre la possibilità di visualizzare alcune righe di codice evidenziate consente una comprensione più immediata del codice sorgente analizzato rendendo più intuitivo il processo di revisione del codice.

4.5 Problematiche riscontrate

Capitolo 5

Test

5.1 Test “Analisi assistita”

Per effettuare i test dell'estensione sono stati utilizzati alcuni siti realizzati per l'ultima edizione del concorso *Accattivante Accessibile*. I test hanno preso in considerazione gli errori individuati da TV insieme a quelli segnalati dalla docente referente e sono stati messi a confronto con gli errori rilevati da *SviluppAbile*. Infine, ho confrontato i risultati elaborando un resoconto per ciascun sito analizzato e, successivamente, ho applicato la metrica F1-score per ottenere un report oggettivo.

5.1.1 F1-score

La F_1 -score è una metrica utilizzata per valutare i modelli di classificazione, sia binari che multi-classe. Essa combina in un unico valore la *precision* (quanto le predizioni positive sono corrette) e il *recall* (quanti casi positivi reali sono stati identificati), calcolandone la media armonica:

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Questa formulazione assicura che il punteggio sia elevato solo quando entrambe le metriche hanno valori alti, penalizzando fortemente gli squilibri. La F_1 -score risulta particolarmente utile in presenza di dati sbilanciati, poiché fornisce una valutazione più affidabile.

5.1.2 Sito web: SudokuWorld

5.1.2.1 Total Validator



Figura 5.1: Analisi di Total Validator sul sito web SudokuWorld

Come visibile nella figura 5.1 è possibile vedere che lo strumento TV_G trova ben 15 errori di accessibilità.

Gli errori principali sono:

- E913 - SC 2.4.6: Le etichette dei controlli dei $form_G$ devono essere univoche. Utilizzare etichette univoche consente agli utenti di distinguere i vari controlli presenti sulla stessa pagina quando vengono letti da uno screen reader. In alternativa, è possibile inserirli all'interno di diversi `<fieldset>` con `<legend>` univoci, poiché il testo del `<legend>` viene letto insieme all'etichetta del controllo. Vedi WCAG2 Success Criterion 2.4.6.
- P883 - SC 1.3.1: Nidificare correttamente le intestazioni ($H1 > H2 > H3$). Gli elementi di intestazione devono essere ordinati in modo gerarchico. Ad esempio, un elemento H2 dovrebbe seguire un H1, un H3 dovrebbe seguire

un H2, e così via. Gli sviluppatori non devono saltare livelli né utilizzare le intestazioni solo per scopi di presentazione. Vedi WCAG2 Failure F43.

5.1.2.2 Lighthouse

Il report generato dallo strumento Lighthouse ha restituito un punteggio di 100/100 nella sezione Accessibility, indicando che, secondo le metriche automatiche adottate, non sono stati rilevati errori o problematiche di conformità (vedi figura 5.2).

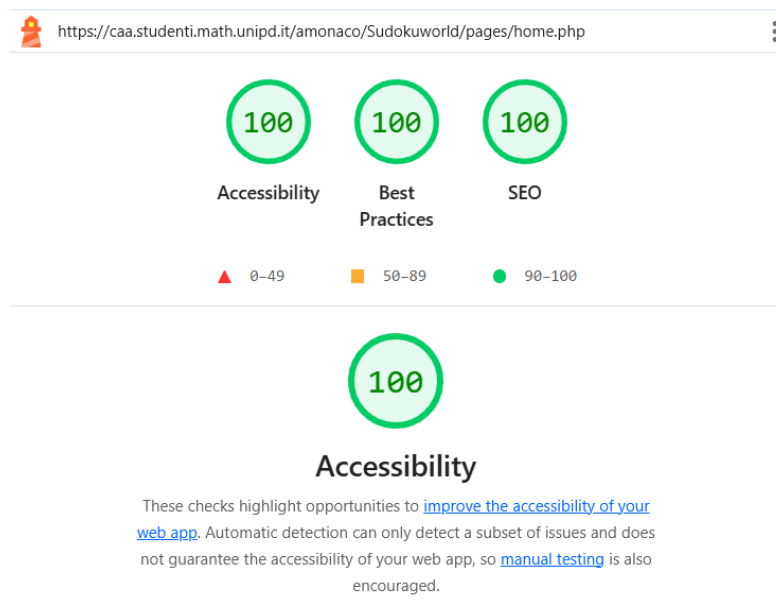


Figura 5.2: Analisi di Lighthouse sul sito web SudokuWorld

5.1.2.3 SviluppAbile

Di seguito vengono riportati i test effettuati con l'estensione *SviluppAbile*.

(pagina home.php)

Il mio codice HTML rispetta le linee guida WCAG?

Il tuo codice HTML sembra essere in parte conforme alle linee guida WCAG (Web Content Accessibility Guidelines), ma potrebbe migliorare ulteriormente per ga-

rantire una maggiore accessibilità. Alcune aree che richiedono ulteriore attenzione includono:

- L'attributo 'aria-label' non è utilizzato in modo coerente nei controlli di navigazione.
- Le etichette dei pulsanti "Cerca" e "Profilo" potrebbero essere più descrittive.

Tuttavia, il tuo codice include molti elementi positivi come l'utilizzo di 'alt' per immagini, la presenza di un titolo ('<title>') e di metatag di descrizione ('<meta name="description">'). Questi sono passaggi fondamentali per migliorare la comprensione del contenuto da parte dei motori di ricerca e degli assistenti.

Per garantire una maggiore accessibilità, considera di applicare ulteriormente le linee guida WCAG 2.1, che offrono standard e raccomandazioni per creare esperienze web più inclusive.

La pagina non rispetta la norma W3C WCAG 2.1 che richiede di nidificare correttamente le intestazioni. Non sono presenti elementi HTML con attributo "h1", mentre ci sono intestazioni con grado inferiore.

Vi sono errori riguardanti i link?

Il codice HTML contiene due carousel circolari:

1. Il primo è definito come "specialProductsCarousel" e si trova nella sezione '<main>'.
2. Il secondo è chiamato "bestSellersCarousel" ed è anch'esso presente all'interno della stessa sezione.

Quanto ai controlli sui link visitati, non sono implementati controlli specifici per la traccia dei link visitati sul sito web.

(pagina sudoku.php)

Il mio codice HTML rispetta le linee guida WCAG?

Il tuo codice HTML non rispetta pienamente le linee guida WCAG (Web Content Accessibility Guidelines) per diverse ragioni:

1. ****La navigazione è troppo complessa****: la barra di navigazione contiene molti elementi e link, potrebbe essere difficile per gli utenti disabili utilizzare l'interfaccia.
2. ****L'accessibilità dei form è insufficiente****: non sono presenti attributi accessibili per i campi di input e submit, rendendo la navigazione dei form difficoltosa per gli utenti che usano dispositivi di assistenza.
3. ****La lettura dei testi in basso risulta difficile****: alcuni elementi come il timer sono troppo piccoli per essere letti facilmente dagli schermi con alto contrasto o da dispositivi di assistenza.
4. ****Non ci sono alternative al linguaggio naturale****: le immagini contengono solo icone e simboli, che potrebbero non essere compresi da utenti analfabetici.

Aggiungere attributi accessibili ai campi input e submit:

* Modifica il codice HTML come segue:

```
<form class="d-flex align-items-center my-2 my-lg-0 ms-auto"
action="/amonaco/Sudokuworld/pages/search.php" method="GET">
    <label for="searched-product" class="visually-hidden">Cerca</label>

    <input class="form-control me-2" type="search" name="searched-product"
id="searched-product" placeholder="Cerca" value="" aria-label="Search"
/>

    <button class="btn btn-outline-success" type="submit" aria-label="Submit
the form">Cerca</button>

</form>
```

Aggiungere tag alt alle immagini: assicurati di aggiungere il contenuto alternativo per ogni immagine.

* Modifica il codice HTML come segue:

```
<a class="nav-link" href="/amonaco/Sudokuworld/pages/sudoku.php">
    <i class="bi bi-grid-3x3-gap-fill"></i>
</a>
```

Aggiungere tag aria: per fornire informazioni aggiuntive alle pagine web, puoi utilizzare gli attributi aria.

* Modifica il codice HTML come segue:

```
<h1 class="container border border-black p-2 mb-4 rounded text-center"
id="sudokuTitle" role="heading" aria-level="1">!!  Sudoku !!</h1>
```

Controlla l'accessibilità con strumenti di testing: prima della pubblicazione, assicurati di utilizzare strumenti come Lighthouse o WAVE per valutare la tua pagina web e risolvere eventuali problemi.

Ricorda che un sito web accessibile è più facile da utilizzare per tutti gli utenti, compresi quelli con disabilità.

DOMANDE SUGGERITE:

"Quale è il criterio WCAG 2.1 che riguarda la navigazione con tastiera?"

"Come posso utilizzare gli strumenti di analisi del codice per identificare le aree dell'accessibilità da migliorare nel mio sito web HTML?"

"Come posso utilizzare gli strumenti di analisi del codice per identificare le aree dell'accessibilità da migliorare nel mio sito web HTML?"

Per identificare le aree dell'accessibilità da migliorare nel tuo sito web HTML, puoi utilizzare gli strumenti di analisi come W3C Validator e Lighthouse. Il W3C Validator verifica la correttezza del codice HTML e fornisce suggerimenti per migliorare l'accessibilità. Lighthouse è uno strumento open-source che valuta l'accessibilità, le prestazioni e la sicurezza del tuo sito web.

"Come posso utilizzare le liste di controllo di WCAG (Web Content Accessibility Guidelines) per migliorare l'accessibilità nel mio sito web?"

Le Liste di Controllo (LC) di WCAG 2.1 forniscono un quadro dettagliato per valutare e migliorare l'accessibilità del tuo sito web.

Per iniziare, leggi attentamente le LC relative alle sezioni più rilevanti per il tuo sito:

- * Guida per i fornitori di contenuti (G): informazioni generali sull'accessibilità.
- * Accessibilità delle pagine Web: principi e strategie per raggiungere l'accessibilità.

In particolare, è importante analizzare le seguenti sezioni:

- Principio 1 (Azienda responsabile): il tuo sito deve essere progettato in modo da non richiedere agli utenti di utilizzare dispositivi assistivi o attrezzature specializzate.
- Principio 2 (Interfacce, componenti e layout di pagina): l'accesso alle funzionalità del tuo sito deve essere possibile con dispositivi e browser diversi.
- Principio 3 (Informazioni e comunicazione): il contenuto del tuo sito deve essere chiaro, comprensibile ed accessibile a tutti.

5.1.2.4 Resoconto finale

SviluppAble riesce a riconoscere i principali problemi di struttura semantica e *form_G*.

Ha individuato gli stessi problemi evidenziati da *TV_G*, in particolare:

1. Struttura delle intestazioni (WCAG 1.3.1): in diverse pagine è stata rilevata la mancata nidificazione corretta degli heading (<h1> > <h2> > <h3>).
2. Accessibilità dei *form_G*: problemi relativi a etichette non uniche e campi privi di label/aria-label sono stati segnalati da entrambe le analisi.
3. Mancanza di pulsanti submit o pulsanti non correttamente etichettati.

PRO:

Rispetto alla valutazione di *TV_G*, *SviluppAble* fornisce descrizioni dettagliate dei problemi e suggerimenti di correzione con esempi di codice, non limitandosi alla sola segnalazione dell'errore. Inoltre individua aspetti migliorabili non segnalati da

TV_G, come l'uso non coerente di aria-label nei controlli di navigazione, etichette di pulsanti poco descrittive, mancanza di alternative testuali esaustive per immagini già dotate di alt.

Rispetto allo strumento Lighthouse, il quale assegna un punteggio del 100% sull'accessibilità, *SviluppAbile* individua errori e aiuta a correggerli.

Continuando a fare domande nella chat si possono approfondire man-mano i vari aspetti dell'accessibilità, cosicché lo sviluppatore possa migliorare la propria pagina web.

CONTRO:

Rispetto alla valutazione di *TV_G* e di un esperto di accessibilità web, *SviluppAbile* non segnala alcuni dettagli di esperienza utente (ad esempio mancanza di presentazione del sito, estetica troppo predefinita, assenza di padding/margini); problemi di navigazione post-login/logout o di comportamento del carosello, che sono più legati alla logica di funzionamento che al solo *HTML_G*.

SviluppAbile necessita di domande specifiche e in alcuni casi del caricamento degli errori individuati da altri strumenti (Total Validator) per l'individuazioni di alcuni errori.

Inoltre, essa non ha accesso al file di stile, quindi non può controllare la parte relativa ai colori/link visitati.

Da notare che Ollama usa la parola "disabili", termonilogia nè esatta nè tantomeno rispettosa. Il termine corretto è utenti con disabilità.

CONCLUSIONE:

Il confronto evidenzia che *SviluppAbile* è efficace nell'individuare gran parte delle non conformità WCAG 2.1 di tipo tecnico e fornisce un supporto pratico agli sviluppatori grazie ai suggerimenti di modifica.

La valutazione del sito web effettuata durante il concorso, seppur meno dettagliata sul piano tecnico, integra invece aspetti legati alla fruibilità reale del sito e alla

presentazione dei contenuti, che attualmente richiedono ancora un’analisi manuale.

5.2 Test “Modalità guidata”

Per effettuare i test utili a dimostrare l’efficacia di *SviluppAbile*, ho creato una semplice pagina *HTML_G* (vedi figura 5.3).

```
1 <!DOCTYPE html>
2 <html><head>
3 <title>Pagina di test</title>
4 </head>
5 <body>
6 <h1>Test</h1>
7 <p>Pagina creata per testare l'estensione
  SviluppAbile</p>
8 </body></html>
```

Figura 5.3: Pagina HTML base per effettuare i test della modalità guidata

Come si crea un <head> che contenga tutti gli elementi utili per una pagina accessibile?

Per creare un head accessibile, assicurati di includere i seguenti elementi:

1. ****Metatag Titolo (title)****: Utilizzato dal browser come titolo della finestra della pagina.
2. ****Metatag Descrizione (description)****: Descrive il contenuto della pagina e appare nella barra degli strumenti del browser e nei risultati dei motori di ricerca.
3. ****Metatag Chiave (keywords)****: Non più importante come in passato, ma puoi utilizzarla se necessario per le ricerche.

Inoltre, assicurati di utilizzare un markup semantico corretto per le sezioni della pagina. Ad esempio, puoi utilizzare ‘<header>’, ‘<nav>’, ‘<main>’, ‘<section>’, ‘<article>’, ‘<footer>’ in luogo di semplici div.

Infine, assicurati di aggiungere un elemento ‘alt’ a tutte le immagini con una descrizione del contenuto dell’immagine.

CODICE SUGGERITO:

```
1 <head>
2 <title>Pagina di test</title>
3 <meta charset="UTF-8">
4 <meta name="viewport" content="width=device-width, initial-scale=1.0">
5 <meta name="description" content="Pagina creata per testare l'estensione
SviluppAbile">
6 <meta name="keywords" content="test,sviluppo accessibile">
7 </head>
```

Come posso ora creare dei paragrafi che rispettino le linee guida BEM?

Per creare paragrafi in base alle linee guida BEM, puoi utilizzare la sintassi seguente:

1. Definisci un classico ‘.block__element’ per il contenuto del paragrafo: ‘class="block__element"’
2. Utilizza ‘.block__element-modificatore’ per aggiungere modificatori: ‘class="block__element-modificatore"’

In questo modo, i paragrafi saranno distinti e facili da identificare tramite la tecnica BEM.

Nota: Assicurati di sostituire ‘-modificatore’ con il nome del modificatore effettivo nel tuo progetto.

CODICE SUGGERITO:

```
1 <h1 class="block__title">Titolo</h1>
2 <p class="block__text -grande">Testo di esempio con un modificatore per
il colore.</p>
```

```
3 <p class="block__text -rosso">Testo di esempio con altro contenuto.</p>
```

Come posso inserire immagini accessibili?

Inserire immagini accessibili significa garantire che le informazioni contenute nelle immagini siano anche disponibili ai utenti con disabilità visiva. Ciò può essere fatto utilizzando l'attributo 'alt' all'interno dell'elemento '':

In questo esempio, l'attributo 'alt' contiene una descrizione breve e concisa della foto. Se il browser non è in grado di visualizzare l'immagine (ad esempio, se si utilizza un lettore schermo), la descrizione contenuta nell'attributo 'alt' verrà visualizzata al posto dell'immagine.

Inoltre, per immagini decorative o grafici che non contengono informazioni importanti, è consigliabile utilizzare l'attributo 'alt=""', in modo da eliminare il testo alternativo dalla navigazione.

CODICE SUGGERITO:

```
1 
```

Come si crea una table rispettando le linee guida WCAG?

Per creare una tabella accessibile secondo le linee guida WCAG, è importante fornire un'esperienza utente inclusiva e facilitare l'accessibilità a tutte le persone, in particolare alle persone con disabilità visive che utilizzano strumenti di lettura ad alta voce o tastiera. Ecco alcuni passaggi per creare una tabella accessibile:

1. **Istruzioni per l'utilizzo della tabella**: fornire un'introduzione sulle informazioni presentate nella tabella e quali sono le colonne principali, aiutando gli utenti che utilizzano strumenti di lettura ad alta voce.
2. **Tabella semantica con role="presentation" impostato su false**: questo ti permetterà a un assistente di lettura di comprendere il contenuto della tabella e fornirlo correttamente all'utente.

Nota che ogni elemento della tabella (celle, intestazioni) deve avere un id o un attributo aria-*, per fornire una descrizione dell'elemento e aiutare gli utenti con disabilità visive a comprendere il contenuto.

CODICE SUGGERITO:

```
1 <table>
2 <caption>Esempio di tabella accessibile</caption>
3 <thead>
4 <tr role="row">
5 <th scope="col" id="nome">Nome</th>
6 <th scope="col" id="età">Età</th>
7 </tr>
8 </thead>
9 <tbody>
10 <tr role="row">
11 <td headers="nome">Mario Rossi</td>
12 <td headers="età">32 anni</td>
13 </tr>
14 <tr role="row">
15 <td headers="nome">Giovanni Bianchi</td>
16 <td headers="età">35 anni</td>
17 </tr>
18 </tbody>
19 </table>
```

Come posso utilizzare gli attributi ARIA per migliorare l'accessibilità degli elementi interattivi?

Gli attributi ARIA (Accessible Rich Internet Applications) sono stati sviluppati per rendere le applicazioni web accessibili a tutti, nonostante eventuali disabilità. Sono utilizzati per aggiungere funzionalità di navigazione e controllo agli elementi interattivi come i bottoni e gli elementi dropdown.

Per migliorare l'accessibilità degli elementi interattivi tramite ARIA, ti suggerisco di utilizzare le seguenti strategie:

1. Utilizza il role di un elemento (ad esempio 'button', 'menuitem') per specificarne la funzione.
2. Aggiungi gli attributi aria-label e aria-describedby per fornire una descrizione dell'elemento interattivo.
3. Utilizza l'attributo aria-haspopup per indicare se un elemento è un menu o una lista.

In questo modo, l'elemento interattivo sarà più facile da navigare e controllare tramite strumenti assistivi come screen reader e mouse navigator.

Per ulteriori informazioni sugli attributi ARIA, ti consiglio di consultare la documentazione ufficiale dell'W3C.

CODICE SUGGERITO:

```
1 <!DOCTYPE html>
2 <html><head>
3 <title>Pagina di test</title>
4 </head>
5 <body>
6 <!-- Utilizzo del role e degli attributi ARIA per il bottone -->
7 <button role="button" aria-label="Clicca per eseguire l'azione"> Esegui
```

```
azione </button>

8 <!-- Utilizzo del role e degli attributi ARIA per il menu ->
9 <ul role="menu">
10 <li role="presentation">
11 <a href="#" aria-label="Opzione 1" aria-haspopup="true">Opzione 1</a>
12 <ul role="sottomenu">
13 <li role="presentation"><a href="#" aria-label="Sottopagina 1.1">Sottopagina
1.1</a></li>
14 <li role="presentation"><a href="#" aria-label="Sottopagina 1.2">Sottopagina
1.2</a></li>
15 </ul></li></ul>
16 </body></html>
```

Capitolo 6

Conclusioni

6.1 Consuntivo finale

6.2 Competenze acquisite

6.3 Valutazione personale

Bibliografia

Siti

- [1] URL: <https://kinsta.com/it/blog/creare-estensione-chrome/>.
- [2] URL: <https://www.levelaccess.com/resources/must-have-wcag-2-1-checklist/>.
- [3] URL: <https://wcag.it/>.
- [4] URL: <https://www.w3.org/WAI/fundamentals/>.
- [5] URL: <https://developer.chrome.com/docs/extensions/develop/migrate/what-is-mv3?hl=it> (cit. a p. 10).
- [6] URL: <https://www.eff.org/deeplinks/2021/12/googles-manifest-v3-still-hurts-privacy-security-innovation> (cit. a p. 13).
- [7] URL: <https://developer.chrome.com/docs/extensions/develop/migrate/api-calls?hl=it>.
- [8] URL: <https://www.w3schools.com/html/>.
- [9] URL: <https://www.html.it/guide/guida-html/>.
- [10] URL: <https://www.01net.it/le-principali-novita-di-html5/>.
- [11] URL: <https://www.css3.info/>.
- [12] URL: <https://prismjs.com/>.
- [13] URL: <https://ollama.com/blog/gpt-oss>.
- [14] URL: <https://ollama.com/library/mistral>.
- [15] URL: <https://ollama.com/library/llama3>.

- [16] URL: <https://www.microsoft.com/it-it/>.
- [17] URL: <https://digital-strategy.ec.europa.eu/it/policies/web-accessibility-directive-standards-and-harmonisation> (cit. a p. 2).
- [18] *Application Programming Interface*. URL: https://www.treccani.it/enciclopedia/api_%28Lessico-del-XXI-Secolo%29/.
- [19] *Asincrono*. URL: <https://www.treccani.it/vocabolario/asincrono/>.
- [20] *Backend*. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end.
- [21] *Browser*. URL: <https://www.treccani.it/vocabolario/browser/>.
- [22] *Chatbot*. URL: https://it.wikipedia.org/wiki/Chat_bot.
- [23] *ChatGPT*. URL: <https://www.treccani.it/enciclopedia/eol-chatgpt/>.
- [24] *Client-side*. URL: https://it.wikipedia.org/wiki/Lato_client.
- [25] *Cloud computing*. URL: [https://www.treccani.it/vocabolario/cloud-computing_res-4d54e944-8996-11e8-a7cb-00271042e8d9_\(Neologismi\)/](https://www.treccani.it/vocabolario/cloud-computing_res-4d54e944-8996-11e8-a7cb-00271042e8d9_(Neologismi)/).
- [26] *Debugging*. URL: <https://www.treccani.it/vocabolario/debugging/>.
- [27] *Desktop-first*. URL: <https://hvdig.co.uk/web-agency/mobile-first-vs-desktop-first>.
- [28] *Document Object Model*. URL: https://it.wikipedia.org/wiki/Document_Object_Model.
- [29] *eXtensible Markup Language*. URL: <https://it.wikipedia.org/wiki/XML>.
- [30] *Form*. URL: [https://www.treccani.it/vocabolario/form_\(Neologismi\)/](https://www.treccani.it/vocabolario/form_(Neologismi)/).
- [31] *GIT*. URL: [https://it.wikipedia.org/wiki/Git_\(software\)](https://it.wikipedia.org/wiki/Git_(software)).
- [32] *Graphics Processing Unit*. URL: <https://www.treccani.it/vocabolario/url/>.

- [33] *Intelligenza Artificiale*. URL: [https://www.treccani.it/enciclopedia/intelligenza-artificiale_\(Enciclopedia-della-Matematica\)/](https://www.treccani.it/enciclopedia/intelligenza-artificiale_(Enciclopedia-della-Matematica)/).
- [34] *Large Language Model*. URL: [https://www.treccani.it/vocabolario/neo-modello-linguistico-di-grandi-dimensioni_\(Neologismi\)/](https://www.treccani.it/vocabolario/neo-modello-linguistico-di-grandi-dimensioni_(Neologismi)/).
- [35] *Latenza*. URL: <https://www.treccani.it/vocabolario/latenza/>.
- [36] *Markup*. URL: https://it.wikipedia.org/wiki/Linguaggio_di_markup.
- [37] *Open source*. URL: <https://www.treccani.it/vocabolario/open-source/>.
- [38] *Open Weight*. URL: <https://invezz.com/it/notizie/2025/08/05/openai-rilascia-i-modelli-open-weight-cosa-sono-e-perche-cambia-tutto/>.
- [39] *Plugin*. URL: [https://it.wikipedia.org/wiki/Plugin_\(informatica\)](https://it.wikipedia.org/wiki/Plugin_(informatica)).
- [40] *Proof of Concept*. URL: https://it.wikipedia.org/wiki/Proof_of_concept.
- [41] *Quantizzazione*. URL: <https://www.treccani.it/vocabolario/quantizzazione/>.
- [42] *Script*. URL: <https://www.treccani.it/vocabolario/script/>.
- [43] *Server*. URL: <https://www.treccani.it/vocabolario/server/>.
- [44] *Transmission Control Protocol*. URL: https://it.wikipedia.org/wiki/Transmission_Control_Protocol.
- [45] *Uniform Resource Locator*. URL: <https://www.treccani.it/vocabolario/url/>.
- [46] *WebAssembly*. URL: <https://it.wikipedia.org/wiki/WebAssembly>.
- [47] *WebSocket*. URL: <https://it.wikipedia.org/wiki/WebSocket>.
- [48] *World Wide Web*. URL: https://it.wikipedia.org/wiki/World_Wide_Web.

CAPITOLO 6. BIBLIOGRAFIA

- [49] *World Wide Web Consortium*. URL: https://it.wikipedia.org/wiki/World_Wide_Web_Consortium.