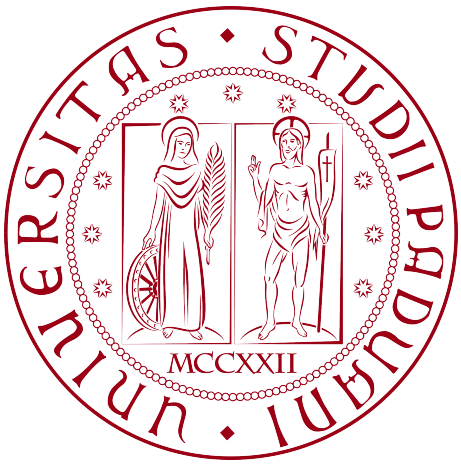
Università degli Studi di Padova

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 Triennale

Relatrice

Prof.ssa Ombretta Gaggi

Laureanda

Elena Chilese

Matricola 2008074

Anno Accademico 2024-2025

Ringraziamenti

Padova, Settembre 2025 *Elena Chilese*

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

Elenco delle figure

Elenco delle tabelle

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.
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, 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.

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.
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 WAVE, Axe e Lighthouse, che offrono principalmente funzionalità di scansione automatica e generazione di report.
Questi strumenti tuttavia spesso forniscono una panoramica statica e generica dei problemi riscontrati; i risultati tendono ad essere formulati in modo formale e talvolta poco comprensibile per l'utente, il quale è quindi costretto a interpretare autonomamente i dati, rischiando di applicare soluzioni non sempre pienamente conformi alle linee guida.
Alcuni software integrano suggerimenti o tutorial, ma raramente utilizzano modelli di intelligenza artificiale 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. Fornirsce inoltre una valutazione personale dell'esperienza di stage interno.

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 e le righe di codice utilizzano il formato `monospace`.

Capitolo 2

Analisi dei requisiti

2.1 Obiettivo del progetto

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.
Introdotto nel 1991 da Tim Berners-Lee, *HTML*$_G$ si è evoluto 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 `<img>` per l'inserimento di immagini, o `<br>` per i ritorni a capo. Questa struttura gerar-

chica e nidificata consente al *browser*$_G$ di interpretare correttamente i contenuti, garantendo coerenza tra presentazione e significato.

La versione attuale, HTML5, 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, 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 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 Live-

Script, 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.
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, 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 comporta limitazioni importanti: l'`API 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, WebAssembly 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 presta-

zioni. 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.

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 è una piattaforma che consente di eseguire localmente modelli di intelligenza artificiale 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, 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*.

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 GPU 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*$_G$ in formato MXFP4, Ollama riesce a ridurre dra-

sticamente 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*$_G$ NVIDIA GeForce RTX e RTX PRO assicura alte prestazioni in locale. .

Figura 3.4: Logo Ollama

3.2.3 GitHub

GitHub 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*$_G$, nella quale il codice è accessibile, riutilizzabile e migliorabile da chiunque.

Figura 3.5: Logo GitHub

3.2.4 VSCode

Visual Studio Code, 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*$_G$ 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

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 (POC) 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 di intelligenza artificiale, necessaria per avviare i primi test di analisi e generazione di suggerimenti.
Il POC 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, 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.

4.2.2 Prodotto finale

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 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
        fetch('http://localhost:11434/api/generate', {
3      method: 'POST',
4      headers: { 'Content-Type': 'application/json' },
5      body: JSON.stringify({
6        model: "llama3.1:8b",
7        prompt,
8        stream: false
9      })
10   });
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 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 insieme alla richiesta di modifica, scenario in cui l'IA produce sia una risposta argomentata sia un blocco di codice pronto per es-

sere 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 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. 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”

5.1.1 Sito web: SudokuWorld

5.1.1.1 Total Validator

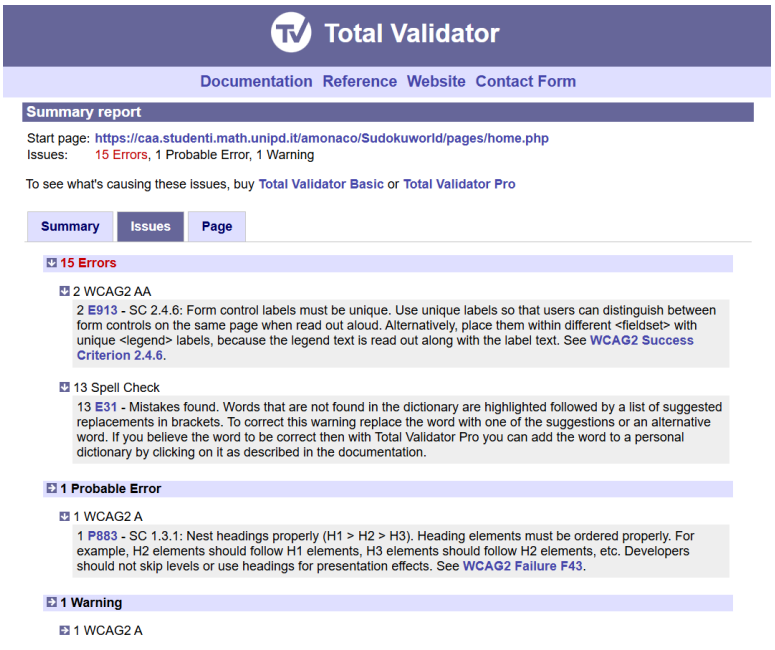


Figura 5.1: Analisi di Total Validator sul sito web SudokuWorld

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

Gli errori principali sono:

- E913 - SC 2.4.6: Le etichette dei controlli dei form 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.

- E31 - Sono stati rilevati errori ortografici. Le parole non presenti nel dizionario vengono evidenziate e accompagnate da un elenco di possibili sostituzioni (in parentesi).

- 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.

5.1.1.2 SviluppAbile

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

5.1.1.3 Resoconto finale

5.2 Test "Modalità guidata"

Capitolo 6

Conclusioni

6.1 Consuntivo finale

6.2 Competenze acquisite

6.3 Valutazione personale

Acronimi e abbreviazioni

AI *Artificial Intelligence*$_G$. iv, 3

API *Application Program Interface*$_G$. iv, 9, 11, 12

CSS Cascading Style Sheets. 7, 9

DOM Document Object Model. 2, 8, 9, 11, 12

GPU *Graphics Processing Unit*$_G$. 15

HTML HyperText Markup Language. 2, 6–9

IA *Intelligenza Artificiale*$_G$. iv, 3

JS JavaScript. 9

LLM *Large Language Model*$_G$. 2

MV2 Manifest V2. 11

MV3 Manifest V3. 10–12

URL *Uniform Resource Locator*$_G$. 7, 11

W3C *World Wide Web Consortium*$_G$. 6

WCAG Web Content Accessibility Guidelines. 1, 3

XML *Extensible Markup Language*$_G$. 7

Glossario

API $_G$ in informatica con il termine *Application Programming Interface* (ing. interfaccia di programmazione di un'applicazione) si indicano regole e specifiche per la comunicazione tra *software*.

Tali regole fungono da interfaccia tra i vari *software* e ne facilitano l'interazione, allo stesso modo in cui l'interfaccia utente facilita l'interazione tra uomo e *computer*.

[1] . 4

Asincrono non sincrono, che non avviene o si manifesta cioè nel medesimo tempo, o, in senso più tecnico, che manca di sincronismo.

[2] . 9

Backend con il termine *"backend"* si intende la parte non visibile all'utente di un programma, che elabora e gestisce i dati generati dall'interfaccia grafica.

[3] . 9

Browser nel linguaggio informatico, programma di un computer che permette il collegamento alla rete Internet e mediante il quale si può navigare da un sito telematico all'altro.

[4] . 7, 9–11

Chatbot è un software progettato per simulare una conversazione con un essere umano.

[5] . iv

ChatGPT implementazione del modello di intelligenza artificiale (IA) sviluppato da OpenAI, basato sull'architettura GPT-3 (Generative Pre-trained

Transformer 3) e progettato per comprendere e generare testo in linguaggio naturale.

[**site:chatgpt-treccani**] . 2

Chrome-based un *browser*$_G$ o software che utilizza la base del codice sorgente del progetto open-source Chromium, la stessa foundation di Google Chrome.

. iv, 2

Client-side in informatica, nell'ambito delle reti di calcolatori, il termine lato client (client-side in inglese) indica le operazioni di elaborazione effettuate da un client in un'architettura client-server.

[7] . 9

Cloud computing la tecnologia che, sotto forma di servizio offerto dal provider al cliente, permette di memorizzare e di elaborare i dati e i programmi di un utente grazie all'utilizzo di risorse hardware o software distribuite in rete.

[8] . 13

Debugging nel linguaggio dell'informatica, operazione di messa a punto di un programma, un'applicazione, ecc., consistente nella ricerca (di norma effettuata dall'elaboratore) e nella correzione (talvolta automatica) degli errori di procedura, relativi al tipo di linguaggio impiegato, che impediscono o rendono difettosa l'elaborazione.

[9] . 16

Desktop-first significa progettare il sito web avendo come obiettivo principale l'esperienza su desktop. Sebbene sarebbe facile pensare che il desktop-first sia un approccio ormai superato, in realtà esistono diverse ragioni per cui molti scelgono ancora di partire in grande e poi ridimensionare.

[10] . 17

Form in Internet, modulo telematico a disposizione dell'utente per l'inserimento e l'invio di dati.
[13] . 6, 7

Latenza in informatica, il tempo impiegato da un'informazione per andare da un'unità all'altra di un sistema, in partic. da un sensore al relativo elaboratore (è detto anche l. di risposta).
[17] . 13

Markup un linguaggio di *markup* (in italiano linguaggio di marcatura o linguaggio di formattazione) è un insieme di regole che descrivono i meccanismi di rappresentazione (strutturali, semantici, presentazionali) o d'impaginazione di un testo.
[18] . 7, 8

Open source in informatica, software non protetto da copyright, il cui codice sorgente è lasciato alla disponibilità degli utenti e quindi liberamente modificabile.
[19] . 15

Plugin in campo informatico è un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie.
[20] . 3, 7, 8

Quantizzazione in elettronica e nella tecnica delle telecomunicazioni, l'operazione, attuata con un quantizzatore, consistente nel suddividere il campo di variabilità di una grandezza continua in un numero finito di intervalli (definiti a volte «quanti»), in ciascuno dei quali la grandezza è considerata costante e sostituita con un valore rappresentativo.
[21] . 14

Script in informatica, programma o sequenza di istruzioni che viene interpretata o portata a termine da un altro programma.
[22] . 9, 11, 12

Server in informatica (con riferimento a una rete di calcolatori), calcolatore che svolge funzioni di servizio per tutti i calcolatori collegati.
[23] . 9

Bibliografia

Siti

[1] *Application Programming Interface*. URL: https://www.treccani.it/enciclopedia/api_%28Lessico-del-XXI-Secolo%29/ (cit. a p. ii).

[2] *Asincrono*. URL: https://www.treccani.it/vocabolario/asincrono/ (cit. a p. ii).

[3] *Backend*. URL: https://it.wikipedia.org/wiki/Front-end_e_back-end (cit. a p. ii).

[4] *Browser*. URL: https://www.treccani.it/vocabolario/browser/ (cit. a p. ii).

[5] *Chatbot*. URL: https://it.wikipedia.org/wiki/Chat_bot (cit. a p. ii).

[6] *ChatGPT*. URL: https://www.treccani.it/enciclopedia/eol-chatgpt/.

[7] *Client-side*. URL: https://it.wikipedia.org/wiki/Lato_client (cit. a p. iii).

[8] *Cloud computing*. URL: https://www.treccani.it/vocabolario/cloud-computing_res-4d54e944-8996-11e8-a7cb-00271042e8d9_(Neologismi)/ (cit. a p. iii).

[9] *Debugging*. URL: https://www.treccani.it/vocabolario/debugging/ (cit. a p. iii).

[10] *Desktop-first*. URL: https://hvdig.co.uk/web-agency/mobile-first-vs-desktop-first (cit. a p. iii).

[11] *Document Object Model*. URL: https://it.wikipedia.org/wiki/Document_Object_Model.

[12] *eXtensible Markup Language*. URL: https://it.wikipedia.org/wiki/XML.

[13] *Form*. URL: https://www.treccani.it/vocabolario/form_(Neologismi)/ (cit. a p. iv).

[14] *Graphics Processing Unit*. URL: https://www.treccani.it/vocabolario/url/.

[15] *Intelligenza Artificiale*. URL: https://www.treccani.it/enciclopedia/intelligenza-artificiale_(Enciclopedia-della-Matematica)/.

[16] *Large Language Model*. URL: https://www.treccani.it/vocabolario/neo-modello-linguistico-di-grandi-dimensioni_(Neologismi)/.

[17] *Latenza*. URL: https://www.treccani.it/vocabolario/latenza/ (cit. a p. iv).

[18] *Markup*. URL: https://it.wikipedia.org/wiki/Linguaggio_di_markup (cit. a p. iv).

[19] *Open source*. URL: https://www.treccani.it/vocabolario/open-source/ (cit. a p. iv).

[20] *Plugin*. URL: https://it.wikipedia.org/wiki/Plugin_(informatica) (cit. a p. iv).

[21] *Quantizzazione*. URL: https://www.treccani.it/vocabolario/quantizzazione/ (cit. a p. iv).

[22] *Script*. URL: https://www.treccani.it/vocabolario/script/ (cit. a p. v).

[23] *Server*. URL: https://www.treccani.it/vocabolario/server/ (cit. a p. v).

[24] *Uniform Resource Locator*. URL: https://www.treccani.it/vocabolario/url/.

[25] *World Wide Web*. URL: https://it.wikipedia.org/wiki/World_Wide_Web.

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