UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA

LAUREA TRIENNALE IN INFORMATICA

TECNOLOGIE WEB

EasyGuitar - Accordi accessibili

Informazioni di Accesso e Contatto:

Utente Amministratore:	username: `admin`, password: `admin`
Utente Base:	username: `user`, password: `user`
Indirizzo Sito:	http://tecweb.studenti.math.unipd.it/msanguin
Referente:	aldo.bettega@studenti.unipd.it

Anno Accademico 2025/2026

Indice

1. Introduzione

Nel vasto panorama del web, la ricerca di risorse musicali — in particolare raccolte di testi e accordi per chitarra — è paradossalmente diventata un'esperienza spiacevole e poco accessibile. Le piattaforme esistenti, pur essendo ricche di contenuti, sono spesso caratterizzate da interfacce obsolete, sature di pubblicità invasiva, pop-up aggressivi e una scarsa attenzione alla leggibilità, rendendo la pratica strumentale davanti allo schermo un'operazione faticosa.

Da questa analisi critica nasce l'idea di **EasyGuitar**: un progetto sviluppato con l'intento preciso di offrire una valida alternativa, accessibile e tecnicamente curata, alle raccolte commerciali attuali.

L'obiettivo primario è stato quello di riportare al centro l'esperienza dell'utente musicista. EasyGuitar è stato concepito non come un semplice archivio dati, ma come uno strumento pensato per chi suona: un «leggio digitale» pulito, immediato e privo di distrazioni.

Il sito si propone di rendere la pratica musicale non solo possibile, ma piacevole e inclusiva. Ogni scelta progettuale — dal contrasto cromatico ottimizzato per la lettura (con supporto alla Dark Mode per le esibizioni in ambienti poco illuminati) alla navigazione semplificata per l'uso con tecnologie assistive — è stata guidata dalla volontà di abbattere le barriere digitali.

Sviluppato nell'ambito del corso di Tecnologie Web, EasyGuitar dimostra come l'applicazione rigorosa degli standard di accessibilità (WCAG) e una solida architettura tecnica possano convivere per creare un prodotto che risponde alle reali necessità di un appassionato di musica.

2. Analisi dei Requisiti

2.1. Overview

L'analisi dei requisiti è stata condotta attraverso un confronto tra i componenti del gruppo, guidato in particolare da Marco Sanguin, ideatore del sito e chitarrista amatoriale. Grazie alla sua esperienza personale come musicista, è emersa l'importanza di progettare una pagina dedicata alle canzoni che fosse pulita, chiara e piacevole alla vista. Si è resa evidente anche la necessità di creare un sito con una attenzione particolare all'accessibilità in quanto molte pagine web simile la trascurano completamente. Sono stati individuati come requisiti fondamentali la presenza di un sistema di autoscroll regolabile e una modalità di visualizzazione degli accordi pensata per rispondere alle reali esigenze di un musicista durante l'esecuzione.

2.2. Analisi dell'Utenza (Target)

L'analisi ha identificato come target primario non il neofita alla ricerca di tutorial didattici, bensì il **chitarrista appassionato**. Questo profilo di utenza possiede già alcune competenze tecniche per l'esecuzione e considera il sito web come un supporto strumentale (un «leggio digitale») piuttosto che educativo.

Le necessità fondamentali di questo utente sono:

- **Immediatezza («Suonare subito»):** L'utente desidera ridurre al minimo il tempo che intercorre tra il pensiero di una canzone e la sua esecuzione. La struttura del sito deve abbattere i tempi di caricamento e navigazione.
- **Interfaccia «Smart»:** La pagina deve essere intuitiva e priva di distrazioni cognitive. La navigazione deve fluire naturale, permettendo di trovare testi e accordi velocemente, anche durante una sessione di pratica o una performance informale.

In sintesi, EasyGuitar risponde all'esigenza di un utente che vuole concentrarsi sulla musica e non sulla tecnologia che la veicola.

2.3. Dettaglio delle Funzionalità

Di seguito viene riportata la disamina delle funzionalità implementate, suddivise per tipologia di attore che interagisce con il sistema.

2.3.1. Funzionalità del Visitatore (Accesso Pubblico)

Il sistema garantisce l'accesso immediato al catalogo anche agli utenti non autenticati. Questa scelta elimina le barriere all'ingresso, permettendo la consultazione rapida delle risorse musicali senza obbligo di registrazione.

1. **Consultazione del Catalogo**
   - **Indice Artisti:** Elenco completo e ordinato alfabeticamente di tutti gli interpreti censiti in piattaforma.
   - **Scheda Artista:** Pagina di dettaglio monografica (raggiungibile da Home, Ricerca o Indice) che aggrega la biografia dell'autore e presenta il carosello interattivo di tutti i brani correlati disponibili.
   - **Indice Canzoni:** Repertorio completo dei brani ordinati alfabeticamente, che funge da punto di accesso diretto alle pagine di esecuzione.

2. **Esperienza di Esecuzione (Player)**
   - **Visualizzazione Ottimizzata:** Layout responsive studiato per garantire la massima leggibilità di testi e accordi su qualsiasi dispositivo (Desktop/Mobile), con supporto nativo alla Dark Mode per performance in ambienti scarsamente illuminati.
   - **Autoscroll Adattivo:** Funzionalità di scorrimento automatico della pagina, con velocità regolabile millimetricamente dall'utente, progettata per consentire l'esecuzione strumentale senza la necessità di staccare le mani dallo strumento per scorrere la pagina.

3. **Ricerca Avanzata**
   - **Interfaccia a Tab:** Suddivisione logica dei risultati tra «Artisti» e «Canzoni» per disambiguare rapidamente l'intento di ricerca e migliorare l'usabilità.
   - **Filtri per Competenza:** Sistema di filtraggio avanzato per Lingua e Accordi. In particolare, il filtro accordi permette all'utente di selezionare il proprio bagaglio di conoscenze (es. «Conosco solo DO, RE, SOL») e ottenere come risultato solo i

brani eseguibili con quel set specifico, configurandosi come un potente strumento didattico.

4. **Strumenti di Navigazione**
   - **Global Navigation Bar:** Menu principale situato nell'header, persistente e responsive, che evidenzia visivamente la sezione corrente.
   - **Breadcrumb Dinamica:** Elemento di orientamento gerarchico che permette all'utente di comprendere la propria posizione nella struttura del sito e risalire facilmente ai livelli superiori.
   - **Footer:** Area di navigazione secondaria contenente collegamenti rapidi alle sezioni di servizio e link di utilità.

2.3.2. Funzionalità dell'Utente Registrato

L'utente registrato eredita tutte le funzionalità pubbliche e accede a strumenti di gestione personale. Questo profilo permette di salvare e organizzare i contenuti, trasformando la semplice consultazione in un'esperienza di studio personalizzata.

1. **Accesso e Personalizzazione**
   - **Autenticazione:** Sistema di registrazione e login sicuro tramite credenziali personali.
   - **Gestione Profilo:** Personalizzazione dell'identità utente tramite selezione dell'avatar da un set predefinito di immagini tematiche.
   - **Gestione Raccolte:** Creazione, modifica ed eliminazione di playlist personalizzate per organizzare le sessioni di studio o le scalette.

2. **Operatività sulle Playlist**
   - **Consultazione:** Pagina dedicata per la visualizzazione e l'ordinamento dei brani contenuti nelle playlist.
   - **Aggiunta Rapida:** Implementazione di un modale di ricerca interno alla pagina playlist che permette l'aggiunta immediata di nuovi contenuti tramite query testuale, senza ricaricare la pagina (AJAX).
   - **Editing:** Rimozione intuitiva e immediata dei brani dalle liste.

3. **Sistema dei Preferiti**
   - **Playlist Persistente:** Una raccolta speciale di sistema, non eliminabile, dedicata all'archiviazione rapida dei brani di maggiore interesse.
   - **Accessibilità Trasversale:** L'azione di aggiunta/rimozione dai preferiti (icona «cuore») è disponibile in ogni punto dell'interfaccia: dalle card di anteprima (Home, Risultati di ricerca) alla pagina di dettaglio della singola canzone.

2.3.3. Funzionalità Amministratore (Back-office)

L'amministratore è la figura garante della qualità e dell'integrità dei contenuti della piattaforma. A differenza dell'utente standard, dispone di un accesso privilegiato a una **Dashboard Gestionale** che centralizza le operazioni di manutenzione.

1. **Gestione del Catalogo (CRUD)** L'amministratore ha pieno controllo sulle entità Artista e Canzone attraverso operazioni di Create, Read, Update e Delete:

- **Cura degli Artisti:** Possibilità di inserire nuovi interpreti o aggiornare le biografie e i dettagli di quelli esistenti tramite form validati lato server.
- **Editing dei Brani:** Interfaccia avanzata per l'inserimento e la modifica di testi e accordi. Il form di creazione è progettato per strutturare i dati in modo che siano correttamente interpretabili dal player (es. riconoscimento automatico degli accordi).
- **Manutenzione:** Rimozione immediata di contenuti errati o non conformi agli standard qualitativi della piattaforma.

2. **Moderazione dell'Utenza**
   - **Monitoraggio:** Visualizzazione tabellare della lista completa degli iscritti alla piattaforma.
   - **Revoca Accesso:** Funzionalità di eliminazione degli account utente.

3. Architettura e Progettazione

L'architettura del sistema è stata definita con l'obiettivo di garantire modularità, manutenibilità e sicurezza. Questa decisione didattica ha permesso di avere il pieno controllo sul ciclo di vita della richiesta HTTP e di implementare manualmente i pattern architetturali fondamentali.

3.1. Pattern MVC (Model-View-Controller)

Il cuore dell'applicazione è basato sul pattern architetturale MVC, che garantisce la netta separazione delle responsabilità:

- **Model (Modello):** Gestisce la logica di business e l'interazione con il database. Ogni entità (es. `Utente`, `Canzone`, `Artista`, `Playlist`) possiede una classe dedicata che astrae le operazioni SQL, restituendo oggetti o array di dati pronti per l'elaborazione.
- **View (Vista):** Si occupa esclusivamente della logica di presentazione. I file di vista ricevono i dati dal Controller e generano l'HTML finale da inviare al client. Non contengono logica di accesso ai dati, garantendo pulizia nel codice di frontend.
- **Controller:** Agisce da intermediario. Riceve l'input dell'utente (tramite il Router), invoca i metodi del Model per recuperare o modificare i dati e seleziona la View appropriata per la risposta, costruendola adeguatamente tramite dei placeholder.

Un componente fondamentale di questa architettura è il **Router Custom**. Invece di mappare ogni pagina a un singolo file PHP (es. `playlist.php`), tutte le richieste vengono convogliate a un unico **entry point** (`index.php`) che analizza l'URL, instanzia il Controller corretto ed esegue l'azione richiesta. Questo permette di avere URL «parlanti» (es. `/artista/nome-artista`) e una gestione centralizzata degli errori (404).

3.2. Struttura delle Directory

La struttura del progetto riflette la separazione logica del pattern MVC, mantenendo isolato il codice sorgente (`src`) dalle risorse accessibili pubblicamente (`public`).

```
├── public
│   ├── css
│   ├── img
│   ├── index.php
│   └── js
└── src
    ├── Controllers
    │   ├── Api
    │   └── Base
    ├── Core
    ├── Exceptions
    ├── Helpers
    ├── Models
    └── Views
        ├── components
        ├── layouts
        └── pages
```

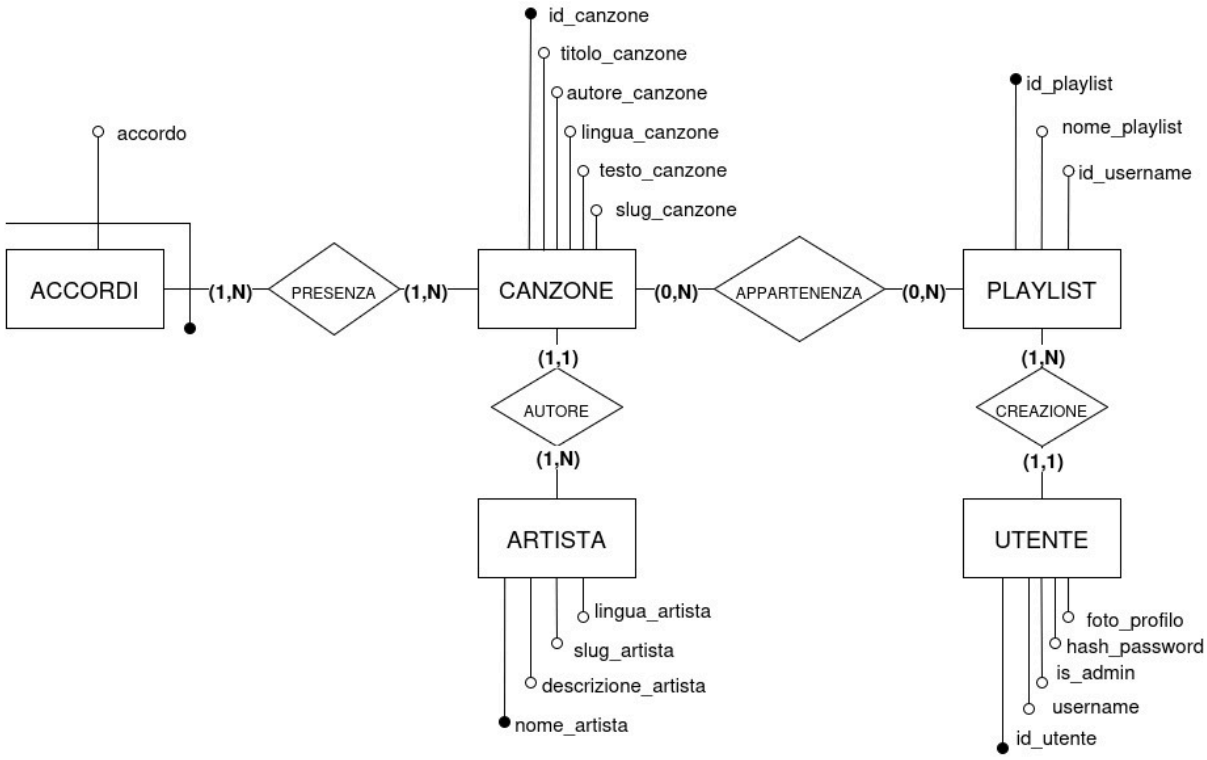
3.3. Database

La persistenza dei dati è gestita tramite MySQL, configurato con il charset `utf8mb4` per garantire il pieno supporto ai caratteri internazionali e speciali, essenziali in un catalogo musicale multilingua.

Lo schema è stato progettato per ottimizzare le relazioni e la navigazione:

- **Gestione Utenti (`utente`):** I ruoli sono definiti tramite un flag booleano `is_admin` (0 per utenti, 1 per amministratori). La sicurezza è garantita dal campo `hash_password` e la personalizzazione avviene tramite `foto_profilo` (riferimento a un set predefinito di avatar).
- **Contenuti e Routing (`artista`, `canzone`):** Entrambe le tabelle includono un campo `slug` (`slug_artista`, `slug_canzone`) utilizzato per generare URL leggibili («SEO-friendly») e puliti, gestiti dal Router custom. La relazione tra canzoni e artisti prevede vincoli di integrità referenziale (`ON DELETE CASCADE`).
- **Relazioni Molti-a-Molti:**
  - **Playlist (`canzoni_playlist`):** Tabella di giunzione che collega `playlist` e `canzone`, permettendo a una canzone di appartenere a più raccolte e viceversa.
  - **Filtri Accordi (`accordi_canzone`):** Gli accordi non sono salvati come semplice testo nella canzone, ma normalizzati in questa tabella. Ciò consente di eseguire query complesse per filtrare i brani in base agli accordi specifici che l'utente sa suonare.

Di seguito viene mostrato lo schema relazionale del database:



3.4. Design e UX (User Experience)

Il design dell'interfaccia è stato guidato dal principio **«Mobile First»**, considerando che il caso d'uso tipico prevede l'utente con lo strumento in braccio e il dispositivo (spesso smartphone o tablet) posizionato su un leggio o sulle gambe.

- **Dark Mode Funzionale:** La modalità scura non è solo una scelta estetica, ma un requisito funzionale per ridurre l'affaticamento visivo e il consumo energetico durante sessioni prolungate. La preferenza viene salvata nel `LocalStorage` del browser per garantire la persistenza tra le visite.
- **Feedback Immediato:** Ogni azione critica (aggiunta ai preferiti, eliminazione playlist) è accompagnata da feedback visivi (toast notification o cambio di stato delle icone) e semantici (aggiornamenti ARIA) per confermare l'avvenuta operazione senza interrompere il flusso di navigazione.
- **Performance Percepita:** L'uso di chiamate asincrone (AJAX) per la ricerca nei modali evita il ricaricamento completo della pagina, mantenendo l'interfaccia reattiva e «app-like».

4. Implementazione Back-End (PHP)

L'infrastruttura di back-end è stata interamente realizzata in PHP «vanilla», senza l'utilizzo di framework esterni..

4.1. Routing

La gestione delle richieste HTTP è centralizzata attraverso un router personalizzato. Attraverso il file `.htaccess` le configurazioni del server sono state riscritte per inviare tutte le richieste al file `public/index.php` (entry point unico) che agisce da Front

Controller. Il router analizza l'URI della richiesta, identifica la risorsa desiderata (es. una pagina artista o una playlist) e delega l'elaborazione al Controller specifico. Questo approccio ha permesso di implementare URL semantici («SEO-friendly») come `/artisti/nome-artista` invece di query string complesse, migliorando sia l'indicizzazione che la leggibilità per l'utente. Inoltre, il router gestisce centralmente le eccezioni, reindirizzando automaticamente verso pagine di errore 404 quando non trova un url valido.

4.2. Gestione dei dati

L'accesso e la manipolazione dei dati persistenti avvengono attraverso il layer dei **Models**, secondo il pattern MVC adottato. Ogni entità del dominio (Utente, Artista, Canzone, Playlist) dispone di una classe dedicata (es. `CanzoneModel`) che astrae la complessità delle query SQL. Questa astrazione permette ai Controller di invocare metodi ad alto livello (come `get_all_songs()` o `get_canzone_by_id()`) senza dover scrivere SQL grezzo, garantendo un codice più pulito, sicuro e manutenibile. I dati recuperati vengono restituiti sotto forma di array associativi, pronti per essere iniettati nelle Viste. Particolare attenzione è stata posta nell'uso di **Prepared Statements** per prevenire vulnerabilità di tipo SQL Injection.

4.3. Autenticazione

Il sistema di autenticazione gestisce l'identità degli utenti e la sicurezza delle sessioni. Le password degli utenti non vengono salvate in chiaro, ma vengono sottoposte a hashing prima di essere memorizzate nel database, garantendo la sicurezza delle credenziali. Il meccanismo di login verifica le credenziali fornite e, in caso di successo, inizializza una sessione PHP sicura, memorizzando l'identificativo dell'utente e il suo ruolo (amministratore o utente base). Questo sistema di ruoli permette di proteggere le rotte sensibili: il router verifica i permessi prima di concedere l'accesso alle pagine di amministrazione o di gestione del profilo, reindirizzando gli utenti non autorizzati alla pagina di login.

4.4. API Interne

Per supportare le funzionalità dinamiche del front-end senza ricaricare l'intera pagina, sono state sviluppate delle API interne in php. Queste API, gestite da specifici Controller (es. `ApiUserController`), ricevono richieste asincrone (AJAX) e restituiscono dati in formato JSON. Un esempio chiave è l'API per la ricerca rapida: quando un utente digita nel modale di aggiunta brani a una playlist, una chiamata asincrona interroga il database e restituisce in tempo reale i risultati filtrati, migliorando drasticamente l'usabilità dell'applicazione.

5. Implementazione Front-End (HTML/CSS/JS)

Il front-end di EasyGuitar è stato sviluppato con un approccio, utilizzando HTML5, CSS3 e JavaScript.

5.1. Validazione Form

La validazione dei dati inseriti dagli utenti avviene su due livelli per garantire sicurezza e usabilità. Lato client, si sfrutta la validazione nativa HTML5 (attributi `required`,

`type="email"`, ecc.) combinata con JavaScript per fornire un feedback immediato all'utente prima dell'invio dei dati. Questo previene errori banali e migliora l'esperienza d'uso. Tuttavia, per garantire la sicurezza e l'integrità dei dati, una seconda validazione rigorosa viene sempre eseguita lato server (PHP) dai Controller, assicurando che nessun dato malformato o malevolo possa raggiungere il database. In caso di errore, il sistema restituisce messaggi chiari e accessibili, associati ai campi errati.

5.2. AJAX e Fetch

L'interattività dell'interfaccia è gestita tramite JavaScript asincrono, utilizzando principalmente l'API `Fetch`. Questa tecnologia è fondamentale per le operazioni che non richiedono un cambio di contesto, come:

- L'aggiunta o rimozione di un brano dai preferiti direttamente dalle card e dalle playlist nella pagina dedicata.
- La ricerca istantanea di canzoni all'interno del modale delle playlist.
- La gestione dell'eliminazione dei contenuti da parte dell'amminstratore.
- La modifica della foto profilo per l'utente autenticato.

L'uso di AJAX permette di aggiornare solo porzioni specifiche della pagina, mantenendo l'applicazione reattiva e fluida, migliorando drasticamente l'usabilità del sito.

6. Accessibilità

Il progetto EasyGuitar è stato sviluppato ponendo l'accessibilità non come una funzionalità accessoria, ma come un requisito non funzionale primario. Il sito aderisce agli standard **WCAG 2.1** (Web Content Accessibility Guidelines) di livello **AA**, garantendo un'esperienza inclusiva per utenti con disabilità visive, motorie o cognitive.

L'obiettivo è stato quello di creare un «leggio digitale» che fosse universalmente fruibile, indipendentemente dal dispositivo di input utilizzato (mouse, tastiera, touch o screen reader) o dalle condizioni ambientali (es. scarsa illuminazione su un palco).

6.1. Navigazione

La navigazione è stata progettata per essere intuitiva, prevedibile e completa, permettendo all'utente di orientarsi nel sistema con il minimo sforzo cognitivo.

- **Struttura Gerarchica e Breadcrumb:** Per mitigare il rischio di disorientamento, ogni pagina interna (escluse Home e Landing Page) implementa un sistema di **Breadcrumb**. Questo elemento, marcato semanticamente con `nav` e `aria-label="Breadcrumb"`, permette all'utente di visualizzare istantaneamente la propria posizione nella gerarchia del sito (es. *Home > Artisti > Nome Artista*) e di risalire rapidamente ai livelli superiori.
- **Menu Responsive e Skip Links:** In ottica **Mobile First**, la barra di navigazione si adatta dinamicamente: su desktop si presenta estesa, mentre su dispositivi mobili si adatta al dispositivo. Per gli utenti che navigano da tastiera, è stato implementato un link nascosto **«Salta al contenuto principale»** (Skip Link) all'inizio del `body`. Questo permette di bypassare i blocchi ripetitivi (come l'header) e accedere

direttamente al `main`, migliorando drasticamente l'efficienza di navigazione tramite tasto `Tab`.

- **Gestione del Focus e Modali (Focus Trap):** Un'attenzione particolare è stata dedicata ai componenti interattivi sovrapposti, come il modale di ricerca nelle Playlist. All'apertura del modale:
  1. Il focus viene spostato programmaticamente all'interno della finestra di dialogo.
  2. Viene attivata una **«Focus Trap»** (trappola del focus): l'utente che naviga con `Tab` cicla esclusivamente tra gli elementi del modale, senza rischiare di interagire accidentalmente con la pagina sottostante oscurata.
  3. Alla chiusura, il focus viene riportato all'elemento che aveva originato l'azione, preservando il contesto di navigazione.

  Il modale è correttamente marcato con `role="dialog"` e `aria-modal="true"`.

- **Attributi ARIA e Feedback Semantico:** Per garantire la comprensione dello stato del sistema agli utenti non vedenti, sono stati impiegati attributi ARIA specifici:
  - `aria-current="page"`: applicato al link del menu attivo per indicare la pagina corrente.
  - `aria-live="polite"`: utilizzato nella ricerca AJAX per annunciare i risultati (o l'assenza di essi) agli screen reader senza interrompere la lettura corrente.
  - `aria-expanded`: per comunicare lo stato (aperto/chiuso) del menu mobile e dei pannelli a soffietto.

6.2. Colori e Contrasto

La gestione cromatica di EasyGuitar risponde a una doppia esigenza: estetica (identità visiva) e funzionale (leggibilità in contesti di performance).

- **Supporto Nativo alla Dark Mode:** Considerando il caso d'uso tipico del musicista (spesso in ambienti con luci soffuse), è stata implementata una modalità scura attivabile tramite toggle globale. La preferenza viene salvata nel `LocalStorage` per garantire la persistenza. Questa modalità non è una semplice inversione di colori, ma una palette studiata **ad hoc** per ridurre l'affaticamento visivo mantenendo i rapporti di contrasto corretti.

- **Rapporto di Contrasto (WCAG AA):** Tutti i testi essenziali rispettano il rapporto di contrasto minimo di **4.5:1** rispetto allo sfondo, verificato tramite strumenti come **WebAIM Contrast Checker**. Questo vale sia per il tema chiaro (testo scuro su sfondo chiaro) che per il tema scuro (testo chiaro su sfondo scuro). Particolare attenzione è stata posta ai testi colorati sugli sfondi testurizzati (es. texture legno), dove è stato applicato un overlay semitrasparente o un'ombra (`text-shadow`) per garantire la leggibilità.

- **Indipendenza dal Colore:** Le informazioni non sono mai veicolate esclusivamente tramite il colore. Ad esempio, i messaggi di errore nei form non sono solo rossi, ma sono accompagnati da un'icona distintiva e da un messaggio testuale esplicito (associato via `aria-describedby`). I link all'interno del corpo del testo sono distin-

guibili non solo per il colore differente, ma anche per la sottolineatura o il grassetto, garantendo l'accessibilità anche a utenti affetti da daltonismo o discromatopsia.

7. Test e Validazione

La fase di testing e validazione ha rappresentato uno step cruciale dell'intero ciclo di sviluppo. L'approccio adottato è stato ibrido: da una parte una verifica continua e manuale durante la stesura del codice, dall'altra una validazione automatica finale mediante strumenti dedicati. Questo doppio binario ha permesso di correggere tempestivamente errori strutturali e di garantire un buon livello di accessibilità e robustezza del prodotto finale.

7.1. Strumenti usati (Validazione Automatica)

Per la validazione del codice e la verifica della conformità agli standard, sono stati impiegati i seguenti strumenti software:

- **Total Validator Base:** Utilizzato per un'analisi complessiva delle pagine. Questo tool ha permesso di verificare simultaneamente la validità del markup HTML, la conformità agli standard CSS e il rispetto delle linee guida WCAG 2.1. È stato fondamentale per individuare errori di sintassi nidificati e attributi mancanti (come le etichette per i form). È stato principalmente usato nella fase finale, per un controllo accurato del sito.
- **Nu Html Checker (W3C):** Lo strumento ufficiale del W3C è stato impiegato per garantire che il codice rispettasse rigorosamente la sintassi HTML5 e le regole XML-like richieste dal progetto (chiusura dei tag, annidamento corretto). È stato usato durante buona parte del progetto, verso la fase finale, utile per la sua semplicità di utilizzo.
- **Colour Contrast Analyser & Color Safe:** Dati i requisiti stringenti di accessibilità visiva, questi strumenti sono stati essenziali per la definizione della palette cromatica.
  - **Colour Contrast Analyser** è stato usato per verificare puntualmente il rapporto di contrasto (minimo 4.5:1 per il livello AA) tra testo e sfondo, specialmente nella modalità **Dark Mode**.
  - **Color Safe** ha guidato la scelta di colori accessibili in fase di design, permettendo di bilanciare l'estetica con la leggibilità.

7.2. Test manuali

La validazione automatica, per quanto potente, non può sostituire il giudizio umano. Per questo motivo, sono stati condotti test manuali approfonditi per verificare la reale usabilità del sito.

- **Navigazione tramite Screen Reader (NVDA):** È stato utilizzato lo screen reader **NVDA** (NonVisual Desktop Access) per simulare l'esperienza di un utente affetto da disabilità visiva. Il test si è concentrato sulla verifica della semantica degli elementi e sul corretto funzionamento degli attributi ARIA implementati (come

`aria-live` per la ricerca dinamica). Si è verificato che le notifiche di errore e i risultati della ricerca venissero annunciati correttamente senza sovrapposizioni audio.

- **Navigazione da Tastiera:** È stato verificato che tutte le funzionalità del sito fossero fruibili senza l'ausilio del mouse. In particolare:
  - Il focus visibile è sempre presente e chiaro.
  - L'ordine di tabulazione è logico e segue la struttura visiva della pagina.
  - Le «Focus Trap» all'interno dei modali funzionano correttamente, impedendo al focus di uscire dalla finestra di dialogo attiva.
  - Il link «Salta al contenuto principale» funziona come previsto.
- **Compatibilità Browser (Cross-browser):** Il sito è stato testato sui principali browser moderni (**Google Chrome, Mozilla Firefox, Brave Browser**) per garantire che il rendering grafico e le funzionalità JavaScript fossero consistenti indipendentemente dal motore di rendering utilizzato dall'utente.
- **Responsive Design e Mobile:** Oltre all'utilizzo di dispositivi fisici, è stata sfruttata intensivamente la **Modalità Ispeziona** dei browser desktop simulando diverse risoluzioni (smartphone, tablet, landscape). Questo ha permesso di verificare:
  - Il corretto collasso del menu di navigazione in formato «Hamburger».
  - La leggibilità dei testi senza necessità di zoom.
  - Il dimensionamento adeguato delle aree cliccabili (touch targets) per evitare errori di digitazione su schermi touch.

8. Organizzazione del gruppo

Questa sezione illustra la metodologia adottata dal team per trasformare l'idea progettuale iniziale in un prodotto software finito. Il gruppo ha operato seguendo una logica di suddivisione funzionale, che ha permesso di parallelizzare lo sviluppo e ottimizzare le competenze individuali.

8.1. Suddivisione dei compiti

Per garantire una copertura completa di tutti gli aspetti dell'ingegneria web, ogni componente si è focalizzato su una specifica area di competenza, assumendone la responsabilità tecnica dall'analisi alla realizzazione.

- **Enrique Hernandez Gris:** Ha curato l'architettura strutturale delle viste (HTML5 semantico) e la logica di presentazione del catalogo. Si è occupato dell'integrazione dei dati nelle pagine principali e dello sviluppo dell'interfaccia di ricerca, garantendo la coerenza visiva tra le varie sezioni del sito.
- **Luca Slongo:** Si è specializzato nell'esperienza utente (User Experience) e nel design dell'interfaccia. Ha definito la palette cromatica in ottica di accessibilità, ha curato la responsività del sito per i dispositivi mobili e ha sviluppato componenti chiave per la navigazione come la Breadcrumb, l'Homepage e la gestione del Profilo Utente.
- **Marco Sanguin:** Ha gestito la logica dinamica lato client e l'interazione asincrona (API). Sfruttando le sue competenze musicali, ha ingegnerizzato il sistema di **Chord**

Parsing per la visualizzazione intelligente degli accordi nella pagina Canzone. Si è inoltre occupato del deployment finale dell'applicazione sul server di produzione.

- **Aldo Bettega:** Ha progettato l'architettura del database e il layer di astrazione dei dati (Models), ottimizzando le query SQL. Ha sviluppato la logica di backend per la ricerca avanzata e ha realizzato l'intero ecosistema del Pannello di Amministrazione, gestendo la sicurezza e la manipolazione dei contenuti.

8.2. Metodologia di Lavoro

L'organizzazione del team è stata improntata alla flessibilità, adottando un approccio agile che ha permesso di conciliare lo sviluppo incrementale del software con gli impegni accademici paralleli.

Per garantire un flusso di lavoro efficiente, sono stati utilizzati strumenti specifici per la gestione della comunicazione e del codice:

- **Comunicazione Sincrona:** La piattaforma **Discord** è stata il punto di riferimento per le riunioni periodiche di allineamento. Questi incontri sono stati fondamentali per monitorare i progressi, risolvere blocchi bloccanti e pianificare le attività successive.
- **Gestione Asincrona:** Il versionamento del codice è stato affidato a **Git** tramite un repository **GitHub** condiviso. Questo ha permesso di lavorare in parallelo sulle diverse feature, mantenendo uno storico delle modifiche e facilitando l'integrazione del codice (merge).

Nonostante la netta suddivisione dei compiti illustrata in precedenza, il gruppo ha mantenuto un approccio fortemente collaborativo. La specializzazione dei singoli membri non ha creato compartimenti stagni; al contrario, è stato costante il supporto trasversale (peer review e debugging congiunto), garantendo che ogni componente del team acquisisse una conoscenza approfondita dell'intero ecosistema del progetto.

9. Conclusioni e Sviluppi Futuri

La realizzazione di EasyGuitar ha rappresentato per il gruppo di lavoro un'opportunità fondamentale per declinare le conoscenze teoriche in un progetto software completo. L'esperienza ha permesso di affrontare le complessità reali dello sviluppo web, ponendo l'accento non solo sulla correttezza formale del codice, ma sull'impatto sociale della tecnologia attraverso l'accessibilità.

9.1. Competenze Acquisite

Il percorso di sviluppo ha permesso di consolidare competenze trasversali, unendo l'ingegneria del software alla sensibilità per la User Experience (UX):

- **Dominio Tecnologico:** La scelta didattica di operare senza l'ausilio di framework ha imposto una comprensione profonda del linguaggio PHP e del pattern architetturale MVC. È stata inoltre affinata la capacità di gestire l'interattività lato client (JavaScript, manipolazione del DOM) e di progettare schemi di base di dati relazionali efficienti.

- **Sensibilità Inclusiva:** Al di là degli aspetti tecnici, il progetto ha radicato nel team la filosofia dello «User-Centered Design». Si è compreso come l'aderenza agli standard WCAG non sia un mero adempimento formale, ma un requisito etico necessario per garantire il diritto all'accesso alle informazioni a ogni categoria di utente.

9.2. Prospettive Future

L'analisi dei risultati ottenuti evidenzia come EasyGuitar, pur essendo un prodotto completo nelle sue funzionalità core, presenti ampi margini di evoluzione. Guardando al futuro, l'obiettivo non risiede tanto nell'espansione indiscriminata delle funzionalità, quanto nel consolidamento della qualità del software esistente.

Sarebbe auspicabile dare continuità al progetto focalizzandosi su tre direttrici principali:

- **Evoluzione del Codice:** Un'attività costante di **refactoring** e ottimizzazione della codebase per garantire performance sempre più elevate e una maggiore manutenibilità nel tempo.
- **Accessibilità Incrementale:** L'accessibilità non è un traguardo statico ma un processo in divenire. L'intento è quello di affinare ulteriormente l'interfaccia utente, recependo i feedback reali per abbattere ogni residua barriera tecnologica.
- **Filosofia del Progetto:** Mantenere inalterata la visione originale di uno strumento essenziale e privo di distrazioni, continuando a sviluppare soluzioni che mettano la musica e l'esecutore al centro dell'esperienza digitale.