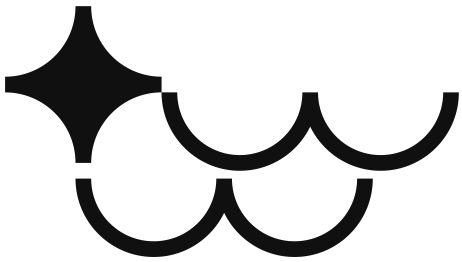


TECNOLOGIE WEB
A.A. 2022/2023

soundstage

Relazione sulla progettazione e realizzazione del sito

Bonavigo Riccardo - 1225420 - riccardo.bonavigo@studenti.unipd.it - **referente**
Gardin Giovanni - 2010003 - giovanni.gardin.1@studenti.unipd.it
Filippini Giovanni - 2052784 - giovanni.filippini@studenti.unipd.it
Marchioro Elena - 2009099 - elena.marchioro.4@studenti.unipd.it

Indirizzo web
http://tecweb.studenti.math.unipd.it/rbonavig

Utenti
Username: *user*, Password: *user*
Username: *admin*, Password: *admin*

Indice

1. Introduzione

Il progetto per il corso di Tecnologie Web per l'Anno Accademico 2022 - 2023 ha come scopo la realizzazione di un sito web che permetta all'utente di cercare, recensire e salvare in liste i film presenti nel database.
Il sito permette a un utente non registrato di visualizzare la descrizione e le recensioni dei film oltre alle liste (dette *collezioni*) create dagli amministratori del sito. Un utente può creare il proprio account personale, ottenendo così la facoltà di creare liste personali, lasciare recensioni sulla pagina di un film e vedere le proprie statistiche sulle sue attività di visione dei film.

2. Analisi dei requisiti

2.1 Mockup

Prima di partire con lo sviluppo vero e proprio sono stati sviluppati dei mockup, ovvero delle rappresentazioni grafiche dimostrative, per capire come organizzare al meglio i contenuti e la gerarchia del sito. In questo modo si è potuta costruire e studiare l'interfaccia grafica prima di passare alla scrittura del codice. Si sono adoperate tutte le misure possibili per rendere il sito semplice, intuitivo ed accessibile a tutti. I mockup ci hanno permesso di testare e correggere eventuali problematiche prima di incontrarle durante lo sviluppo, in modo tale da ottimizzare il lavoro. Importante durante questa fase è stata il mantenere una coerenza tra i vari elementi grafici, imponendo delle regole di stile interne per agevolare la navigazione dell'utente.

2.2 Analisi utenti

soundstage vuole essere un sito utilizzabile da tutti gli appassionati di film. Trattandosi di un pubblico ampio, abbiamo adottato un linguaggio informale, non tecnico e facilmente comprensibile. Struttura e layout sono semplici. Ci aspettiamo che la maggioranza degli utenti acceda al sito da mobile e con browser recenti e aggiornati. L'età degli utenti spazia tra i 18 e i 50 anni.

3. SEO

Muovendo dalla premessa che il codice HTML sia valido, vengono indicate per ordine di importanza le ricerche alle quali il sito vuole rispondere:

- il nome del sito.
- tutte le ricerche che contengono il titolo di un film, un attore, un regista o una casa produttrice.
- il genere di un film.
- ricerche più generali come "Liste film", "Salva film", "Film guardati" o "Film da guardare".

Le parole chiave sono state selezionate pensando sia ad utenti che sanno già cosa stanno cercando, sia ai nuovi utenti che stanno cercando un sito che risponda alle loro esigenze. Si tratta di chiavi di ricerca competitive monopolizzate da grandi siti di settore. Sarà quindi d'obbligo, una volta lanciato il servizio, lavorare sul miglioramento dei fattori off-site, come ad esempio la raccolta di numerosi link in ingresso di qualità.

4. Progettazione

4.1 Schema organizzativo e struttura

Abbiamo adottato uno schema organizzativo esatto per i diversi contenuti ospitati nel sito (film, collezioni, persone, liste). Sono presenti collegamenti contestuali tra film, persone e collezioni. Inoltre, il sito offre la funzione di ricerca e, solo per i film, dei filtri per raffinare i risultati. Gli utenti registrati possono aggiungere qualsiasi film in una o più liste personali.

4.2 Supporto a IE11

No.

4.3 Tipi di utente

I tipi di utenti definiti in fase di progettazione sono:

- Ospite: l'utente ospite può accedere a tutte le pagine pubbliche del sito. Non è consentita la creazione e/o modifica dei dati, la gestione delle liste, e la pubblicazione delle recensioni.
- Registrato: quando un utente si registra sul sito, può aggiungere un film alle proprie liste (di default sono presenti *Visti* e *Da vedere*, ma può crearne delle altre), modificare i propri dati e rilasciare recensioni. Può inoltre accedere alle proprie statistiche.
- Amministratore: in aggiunta alle funzioni rese disponibili per gli utenti registrati, l'amministratore può creare, modificare ed eliminare film, persone e collezioni.

4.4 Scelte e convenzioni interne

- Dove dovrebbe esserci un link alla pagina corrente, questo viene rimpiazzato da uno span con classe `active`. Questo ha uno specifico background che serve ad indicare all'utente che il link non è disponibile. Lo scopo è evitare link circolari.
- In `#page-header` (header principale) è presente un link ridondante alla home nel rispetto della convenzione esterna che vuole che il logo di un sito rimandi alla homepage. Abbiamo deciso di mantenere il duplice link, nascondendo il link nel logo agli screen reader e rimuovendolo dal tab flow.

```
1 <a href="index.php" id="logo" aria-label="Vai alla homepage"
    aria-hidden="true" tabindex="-1">
```

5. Realizzazione

5.1 HTML

Il sito è sviluppato in HTML5 con sintassi XML.

5.1.1 Ancore e sezioni

Abbiamo tentato di mantenere più struttura HTML possibile nei file statici, per poi sostituire con PHP le parti necessarie. Per fare ciò abbiamo usato due tipi di marcatori: ancore e sezioni. I primi sono indicati con `@@ancora@@`, le sezioni sono invece racchiuse da commenti di inizio (`<!-- nome_start -->`) e fine (`<!-- nome_end -->`). Le ancore rappresentano quindi singole stringhe inserite, mentre le sezioni parti che possono essere duplicate o rimosse in base ai casi.

Ad esempio, questo è un pezzo di codice del file `html/film.html`:

```
1  <!-- generi_start -->
2  <section>
3      <header>
4          <h2>Generi</h2>
5      </header>
6      <ul class="pills">
7          <!-- genere_start -->
8          <li><a href="cerca_film.php?fn=genere&fvg=@@id@@">@@nome@@</a></
               li>
9          <!-- genere_end -->
10     </ul>
11 </section>
12 <!-- generi_end -->
```

Un film può avere da 0 a 17 generi associati, scelti tra quelli predefiniti. Se almeno un genere è definito, viene duplicata il giusto numero di volte la sezione `genere`. Altrimenti viene rimossa per intero la sezione `generi`. Questo approccio porta dei vantaggi:

- fissare la struttura, le classi e gli id in html ha permesso di sviluppare in modo indipendente su PHP, CSS e JavaScript
- permette una prima validazione sui file html statici.

Riteniamo pertanto che valga il piccolo costo in termini di performance.

5.1.2 shared.html

Il file `shared.html` è un file HTML valido contenente le sezioni che sono presenti in (quasi) tutte le pagine. Sono:

- head: meta author, meta viewport, link ai fogli di stile e alle utilities JS.
- header: sono presenti le principali pagine del sito: *Home*, *Film*, *Collezioni* e *Persone*. Per gli utenti registrati è inoltre visibile la voce *Liste*, mentre per gli Admin la voce *Aggiungi* per l'aggiunta di nuovi contenuti. Sono inoltre presenti i pulsanti per l'accesso all'account ed il cambio del tema.
- footer: il piede della pagina contiene il link alla sezione contenente alcune informazioni sul progetto, la possibilità di mandarci una mail (con `mailto:`, attualmente un placeholder) e l'attribuzione per la provenienza dei dati.

- server_messages: struttura usata per mostrare messaggi di errore/successo dal server, se presenti.

Queste vengono inserite dinamicamente con PHP durante la costruzione della pagina.

5.2 Errori di navigazione o del server

Se l'utente visita un link errato o inesistente, ad esempio l'url di un film rimosso, viene mostrata una pagina 404 personalizzata. Allo stesso modo, per errori lato server (collegamento assente a DB ecc.) viene mostrata una pagina di errore 500. I messaggi di errore sono informativi e offrono all'utente una soluzione al problema.

5.3 CSS

Per la presentazione abbiamo utilizzato CSS3, cercando di mantenerlo il più semplice possibile e utilizzando un unico file per le varie pagine così da non avere richieste multiple per pagine differenti. Il layout principale è realizzato con grid evitando troppi livelli di nesting. Abbiamo utilizzato anche layout flex-box. Il sito è responsive e si adatta ai diversi dispositivi. Sono presenti tre fogli di stile, dedicati alla versione desktop, mobile e per la stampa. Nessuna regola utilizza `!important`. L'unico uso di una working draft (quindi non nello standard ufficiale e di conseguenza segnalata come errore) riguarda la pseudo-classe `:has()`, usata per arricchire graficamente alcuni elementi. Non si tratta di una caratteristica critica, e pertanto abbiamo deciso di mantenerla.

5.4 Immagini e icone

Le immagini sono salvate in duplice formato: PNG (per garantire retrocompatibilità) e WEBP. Per servirle ai browser utilizziamo il tag HTML5 `<picture>`. Non è stato possibile servire le immagini di background della homepage con duplice standard, perché ad oggi CSS3 non lo prevede (la regola `image-set()` è ancora in draft).
Sono state definite due risoluzioni per le copertine dei film e le immagini delle persone. La risoluzione maggiore è utilizzata nelle pagine che mostrano i dettagli del film, collezione e persona; quelli a risoluzione minore sono invece usati come anteprima nelle altre pagine: ricerca, collezione (film presenti), persona (film in cui ha partecipato) e lista (film presenti). Quando l'immagine di un contenuto non è presente, viene mostrato un segnaposto. Il server PHP si occupa di generare automaticamente tutti i formati richiesti, anche per gli upload degli utenti (ai quali viene anche applicato un crop, se necessario).
Tutte le icone usate nel sito sono SVG inline, compressi ed ottimizzati con `svgo`. Così facendo possiamo aggiornare il colore dei path a seconda del tema selezionato (usando `currentcolor`), senza dover richiedere al server icone dedicate. Questo, a nostro parere, compensa le dimensioni delle pagine HTML leggermente maggiorate.
Nelle pagine di film, collezioni e persone le immagini hanno `alt` vuoto perché riteniamo che non sarebbe stata un'aggiunta utile a contestualizzare l'informazione.

5.5 Validazione dell'input

Il sito fa ampio uso di form (ricerca, recensioni, aggiunta ad una lista, aggiornamento dati). Applichiamo gli stessi controlli sia tramite JavaScript che tramite PHP per assicurare consistenza. Le segnalazioni degli errori sono amichevoli e cercano di guidare l'utente verso una soluzione.
Per la maggior parte dei campi dati abbiamo scelto di segnalare immediatamente gli errori con una validazione `onInput`. Questo perché il nostro sito accetta la maggior parte dei simboli UTF, eccetto le parentesi angolari `< >` (per una questione di sicurezza). Ovunque sia stato usato `onInput`, ci siamo assicurati che la segnalazione dell'errore non avvenisse prematuramente. Altri campi, come quello della e-mail, validano il valore `onChange` per non disturbare l'utente durante l'inserimento o la modifica

dei dati.
Alcuni campi richiedono una data. Se il browser non supporta input HTML5 di tipo `date`, allora il campo degrada in tipo `text` e viene validato con il formato ISO 8601 (YYYY-MM-DD).

5.6 JavaScript

JavaScript è impiegato anche per:

- cambio tema, con priorità al tema utente in `localStorage`, se definito, e alla preferenza utente `prefers-color-scheme`, se espressa.
- apertura e chiusura del menù ad hamburger.
- comparsa dinamica del tasto per saltare all'inizio della pagina.
- durante la modifica o l'aggiunta di un film, per mostrare gli input per la gestione dei membri della crew e dei Paesi partecipanti.

Sono presenti più file JS: `utils.js` contiene funzioni di utilità comuni, mentre i file `validate-*.js` riguardano la validazione di specifiche entità.

Nella pagina di modifica/aggiunta di film (`gest_film`) si verifica un episodio di rottura della separazione tra contenuto e comportamento. I campi relativi ai membri (per i Paesi vale un ragionamento analogo) contengono chiamate `onClick` con parametro `this` per l'aggiunta/rimozione degli elementi. `this` è un riferimento richiesto per inserire o rimuovere nodi del DOM.
Questa scelta è stata obbligata dall'uso di `datalist` anziché campi `select` per la selezione della Persona o del Paese. Si tratta di una precisa scelta di progettazione, volta a ridurre il carico sul client: ad esempio, ad ogni aggiunta di un Paese, JS avrebbe dovuto introdurre 250+ nodi. Così facendo, le prestazioni sarebbero presto degradate. Per l'associazione delle persone, la situazione sarebbe stata ancora più grave. Riteniamo questa scelta il giusto compromesso tra complessità e prestazioni. Tuttavia, se per qualsiasi motivo JS non fosse disponibile, non sarebbe possibile aggiungere/rimuovere elementi.

5.7 Database

5.7.1 Schema

Descrizione delle tabelle che non sono di immediata comprensione:

- `stato`: possibili stati di produzione di un film (produzione, rilasciato, ...).
- `ruolo`: ruoli che una persona può assumere in un film (direttore, produttore, ...). Non è presente il ruolo "attore", poiché non utilizzabile all'interno del sito.
- `film_paese`: Paesi in cui è stato prodotto un film.
- `crew`: rappresenta la troupe cinematografica. In particolare non prevede gli attori.
- `lista`: lista personale creata da un utente.
- `lista_film`: film presenti in una specifica lista.

La rappresentazione grafica dello schema è presente in ultima pagina per aumentare la leggibilità.

5.7.2 Progettazione e popolamento

Nella cartella `/database` sono presenti due file:

- `schema.sql`: struttura pura senza dati.
- `dump.sql`: dump con dati esportato da PhpMyAdmin.

Abbiamo progettato il database per rispettare i vincoli di consegna (leggi terza forma normale). Poi, invece di popolarlo a mano, abbiamo utilizzato i dati forniti dalle API di TMDB: permettono libero uso purché ci siano le dovute attribuzioni. Più precisamente, abbiamo scritto un programmino in Python per scaricare i dati associati ai (circa) 160 film più votati su TMDB. Abbiamo usato qualche libreria:

- `tmdbsimple`: wrapper Python per le API.
- `pandas`: per maneggiare dati.
- `sqlalchemy`: per eseguire query sul database.

5.7.3 Marcatura

I dati forniti da TMDB non avevano marcature di lingua per le parole internazionali, tanto meno per le abbreviazioni. Abbiamo dovuto aggiungerle manualmente per tutti i dati che ne avevano bisogno: film (descrizione), collezioni (descrizioni), persona (nomi). Abbiamo deciso di mantenere i dati più puri possibili: invece di inserire direttamente elementi HTML, abbiamo utilizzato dei marcatori che vengono convertiti dinamicamente da PHP. I marcatori sono quindi di due tipi:

- lingua (lang): sono indicati con `[en]...[/en]`. Dentro le quadre possono esserci 2 o 3 caratteri, come previsto per l'attributo lang secondo lo *Iana language subtag registry*. Ad esempio `[en]stuff[/en]` viene convertito a `<span lang="en">stuff</span>`.
- abbreviazioni (`<abbr>`): possono essere indicati in due modi:
  - con title: `{abbr}Marvel Cinematic Universe;MCU{/abbr}`
  - senza title: `{abbr}FBI{/abbr}`

Si noti che i due esempi sopracitati di abbreviazioni sono in lingua inglese. In questi casi, nel database il marcatore di lingua è esterno a quello di abbreviazione. Quindi: `[en]{abbr}FBI{/abbr}[/en]`.

5.8 PHP

5.8.1 Informazioni generali

Per semplicità di sviluppo e gestione, tutte le pagine sono generate dinamicamente con PHP. L'impatto sulle performance è impercettibile dall'utente finale.
Per l'invio dei dati tramite form con `method="post"` (quindi non per la ricerca) abbiamo usato il modello Post/Redirect/Get. Questo consiste nell'inviare i dati al server su una pagina dedicata (Post), essere reindirizzati ad un'altra pagina (Redirect) che viene poi richiesta al server (Get). Questo modello consente la separazione dei file di visualizzazione dati da quelli che invece li manipolano, oltre che a bloccare il fastidioso problema del doppio invio (ad esempio ricaricando la pagina). L'unico svantaggio è che non permette un semplice ripopolamento dei vari campi di input in caso di errore. Per risolvere, si potrebbe usare AJAX oppure salvare i file in una variabile temporanea. Purtroppo non abbiamo avuto il tempo di gestire la cosa, ma facciamo affidamento sulla validazione dell'input client-side (con JS) per bloccare il submit se i campi non sono validi. Nella radice del progetto sono presenti i file visibili dagli utenti e i file `post_*.php` (come descritti subito sopra). I file dell'area riservata vengono visualizzati solo se l'utente registrato o amministratore ha fatto l'accesso correttamente.
Nella sottocartella `/post` del progetto sono presenti tre file:

- `database.php`: gestisce la connessione al database. Usa una funzione per ogni query necessaria.
- `tools.php`: contiene tutte le funzioni di supporto utilizzate nelle altre pagine. Ad esempio:
  - `buildPage`: aggiunge ad una pagina HTML le sezioni presenti in `shared.html`.
  - `replaceAnchor`: rimpiazza un'ancora con una stringa.

- `uploadImg`: gestisce il crop e salvataggio delle immagini.
- `toHtml`: converte i marcatori del database in elementi ed eventuali caratteri speciali (quote, ampersand, ...) in entità.

- `ini.php`: imposta il charset a UTF-8 ed il livello di log degli errori (nella consegna lasciati visibili).

5.8.2 Sicurezza

- Tutte le query eseguite con `mysqli` sono "prepared statements". Questo evita a tutti gli effetti il rischio di SQL Injection.
- Prima degli inserimenti di dati nel database, i dati vengono "puliti" con un adattamento della funzione vista a lezione. La funzione è `pulisciInputHelper` nel file `php/database.php`.
- Il costruttore in `php/database.php` cattura eventuale eccezione durante tentata connessione al database e rilancia con messaggio generico per evitare leak delle credenziali di accesso nel messaggio di errore dell'eccezione. Questo di fatto non è necessario nel progetto perché in caso di errori con il database viene mostrata la pagina 500, ma è buona pratica.
- Possibilità di nascondere gli eventuali errori di PHP cambiando il flag nel file `php/ini.php`.

5.9 Elementi della UI

5.9.1 Font

La scelta dei caratteri non è stata casuale. Sora, di tipo sans-serif, e Frank Ruhl Libre per la stampa, un font di tipo serif, sono stati scelti per due motivi:

- la vasta quantità di caratteri disponibili permette di visualizzare senza problemi anche i nomi delle persone contenenti caratteri non latini.
- sono font a dimensione variabile, ovvero permettono di definire il peso con valori numerici, senza bisogno di versioni diverse. Questo riduce la quantità di dati trasmessi e il numero di richieste HTTP al server.

I font scelti non supportano completamente i caratteri CJK (Chinese/Japanese/Korean), per cui verranno usati quelli offerti dal sistema operativo. Per avere un supporto completo ai caratteri CJK avremmo avuto bisogno di typeface dedicati, col rischio di appesantire il sito.

5.9.2 Colori

Abbiamo utilizzato quattro colori principali:

- testo.
- primario: per definire lo sfondo sul sito.
- secondario: utilizzato per mettere in risalto l'elemento dallo sfondo.
- terziario: per i pulsanti primari di azione (invio, cerca, cambio tema, ecc).

Abbiamo rispettato il contrasto **4.5:1** per:

- testo - background
- link non visitato - background
- link visitato - background

e **3:1** per:

- testo - link non visitato
- testo - link visitato

Non siamo riusciti a individuare una coppia di colori che permettesse un contrasto 3:1 tra il link non visitato ed il link visitato.

6. Accessibilità

6.1 Scelte fatte per rendere il sito accessibile

Oltre alle caratteristiche di cui abbiamo parlato in precedenza, per migliorare l'accessibilità sono state effettuate altre scelte, elencate di seguito, alcune delle quali sulla base dei consigli rilasciati dall'associazione British Dyslexia.

- Implementazione tema scuro per agevolare la lettura.
- Uso ridotto di verde, rosa e rosso, ritenuti colori difficili per chi ha carenze nella visione dei colori.
- Navigazione da tastiera coerente con l'ordine grafico e completa.
- Tabelle accessibili e facilmente comprensibili.
- Background monocolore per evitare distrazioni.
- Linguaggio semplice e comprensibile.
- Tag `abbr` per aiuto agli screen reader.
- Con l'evento `onSubmit` viene evidenziato il primo campo con errore (se presente).
- Ordine di tabulazione corretto.

Non è stata messa una sitemap in quanto riteniamo che la gerarchia ampia e poco profonda del sito sia facilmente navigabile.

6.2 Aiuti allo screen reader

All'interno del sito sono stati inseriti alcuni link, nascosti in CSS, per migliorare l'accessibilità per gli utenti di screen reader o tastiere. Tutte le pagine includono link *Torna su* e *Salta al contenuto*. Sono poi presenti aiuti per le pagine di ricerca di film, collezioni e persone. Le classi di riferimento sono `.sr-only` e `.skip-to-content`.
Inoltre abbiamo sfruttato in maniera oculata ARIA live-regions e roles (`status` e `alert`) per aiutare l'utente di SR nell'interazione con il sito, in particolare per quanto riguarda messaggi di errore e di successo.

7. Installazione del progetto

Passaggi già effettuati da noi, sono segnati solo per riproducibilità:

1. copiare la cartella del progetto in `public_html` del server
2. con phpMyAdmin, eliminare le tabelle esistenti (se presenti) e ripristinare il dump (`dump.php`). L'eliminazione serve perché il dump forza la creazione.
3. inserire le credenziali di accesso al database nel file PHP che lo gestisce: (`php/database.php`) da riga 8.
4. Configurare in `.htaccess` la gestione dell'errore 404 per le pagine non presenti. Tenere attiva la preferita delle due presenti nel file:
   - `RewriteRule ".*$" "404.php"` (con relative `RewriteCond` immediatamente sopra)
   - `ErrorDocument 404 http://tecweb.studenti.math.unipd.it/rbonavig/404.php`

Delle due versioni per la gestione 404, la seconda è quella standard utilizzata in tutti i server Apache, però per funzionare ha bisogno di un URL completo. La prima invece usa path relativi, quindi funziona correttamente sia dalla rete di dipartimento che utilizzando proxy. Non è ideale perché mostra la pagina 404 correttamente solo per file non esistenti dalla radice (`/nonexistent`); con le subdirectory (`/nonex/nonexistent`) non carica il CSS per via del path relativo. Considerato lo scopo didattico del progetto, abbiamo tenuto attiva la seconda per l'agevolazione al testing.

8. Test

8.1 CI e test automatici

Abbiamo predisposto un container remoto per la CI, basato su Docker, ospitato su Azure e impostato per replicare il più fedelmente possibile lo stack LAMP del dipartimento. La pipeline CI è basata su GitHub Actions e prevede il deploy automatico dei branch sul server di test e una serie di controlli automatici su accessibilità, link non funzionanti e prestazioni (con Google PageSpeed Insights e Lighthouse).

8.2 Peso delle pagine

La pagina più pesante è la homepage con 711 KB complessivi (443 KB al termine del caricamento e fino a quando l'utente non cambia tema). Tutte le altre pagine pensano attorno ai 200 KB. Opportunamente minificati, i tre fogli di stile CSS pesano 18 KB complessivi.

8.3 Test manuali

Tutti i test sono stati eseguiti sulle pagine PHP servite dal server. I test sono stati i seguenti:

- Toptal: test per protanopia, deuteranopia, tritanopia e achromatopsia.
- WAVE e Axe: per controlli generali, non sono stati rilevati errori.
- NVDA, Talkback e VoiceOver: le pagine sono navigabili attraverso screen reader.
- TotalValidator, W3C Validator (per HTML e CSS) e AChecker.
- Compatibilità con browser testata sui browser principali basati su tecnologie Blink, Gecko e WebKit: Opera, Edge, Chrome, Firefox, Samsung Internet, Safari e Safari Mobile.

8.4 Avvisi di validazione

La validazione con Total Validator ha segnalato due falsi positivi:

1. *P871 (Probable error): Link text is missing.*
   Segnalata nel #page-header, dove il logo è un link alla homepage: `<a href="index.php" id="logo" aria-label="Vai alla homepage" aria-hidden="true" tabindex="-1">`
   Il testo del link non è assente, perché riportato nella `aria-label`. In ogni caso, l'elemento è nascosto.
2. *W872 (Warning): Using 'onchange' with selections may prevent keyboard operation.*
   Segnalato nella pagina di ricerca dei film. Abbiamo verificato che non avvenisse cambio di focus o di contesto durante l'interazione con i filtri di ricerca.

9. Organizzazione del lavoro

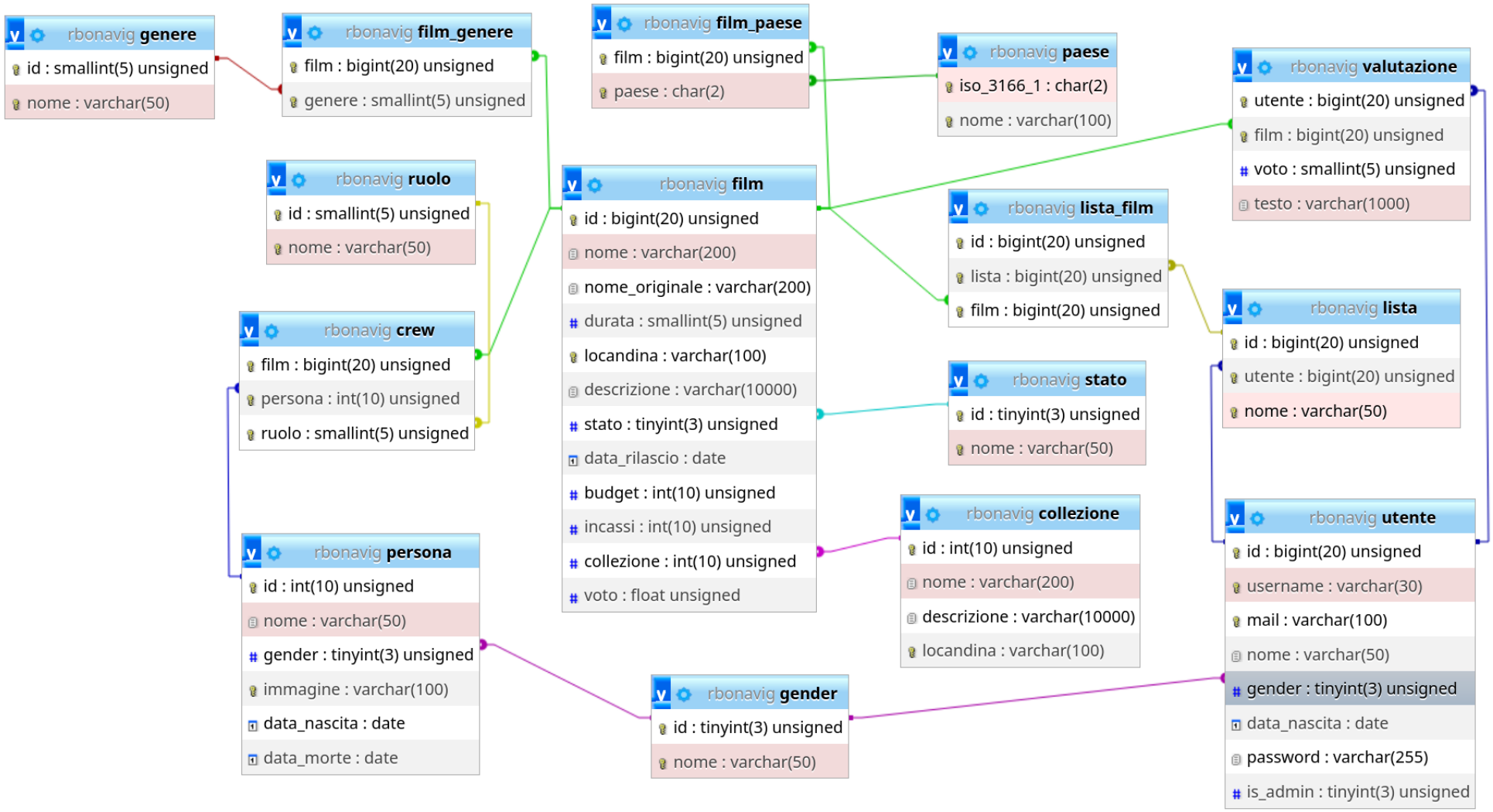
9.1 Modalità di incontro, lavoro, strumenti utilizzati

Il lavoro è iniziato non appena ricevute le specifiche di progetto. Una volta individuato il tema del sito, abbiamo realizzato dei mockup per aiutarci durante le fasi di analisi dei requisiti e progettazione.

9.2 Suddivisione dei ruoli

La suddivisione dei ruoli è stata la seguente:

- Bonavigo Riccardo
  - HTML
  - CSS
  - PHP
  - DB: creazione, popolamento, marcatura
  - Relazione
- Filippini Giovanni
  - HTML
  - CSS
  - DB: marcatura
  - Grafica
  - Testing
  - Relazione
- Gardin Giovanni
  - HTML
  - CSS
  - JS
  - DB: marcatura
  - Testing
  - Relazione
- Marchioro Elena
  - HTML
  - CSS
  - DB: marcatura
  - Testing
  - Relazione


rbonavig genere
id : smallint(5) unsigned
nome : varchar(50)
rbonavig film_genere
film : bigint(20) unsigned
genere : smallint(5) unsigned
rbonavig film_paese
film : bigint(20) unsigned
paese : char(2)
rbonavig paese
iso_3166_1 : char(2)
nome : varchar(100)
rbonavig valutazione
utente : bigint(20) unsigned
film : bigint(20) unsigned
voto : smallint(5) unsigned
testo : varchar(1000)
rbonavig ruolo
id : smallint(5) unsigned
nome : varchar(50)
rbonavig film
id : bigint(20) unsigned
nome : varchar(200)
nome_originale : varchar(200)
durata : smallint(5) unsigned
locandina : varchar(100)
descrizione : varchar(10000)
stato : tinyint(3) unsigned
data_rilascio : date
budget : int(10) unsigned
incassi : int(10) unsigned
collezione : int(10) unsigned
voto : float unsigned
rbonavig lista_film
id : bigint(20) unsigned
lista : bigint(20) unsigned
film : bigint(20) unsigned
rbonavig lista
id : bigint(20) unsigned
utente : bigint(20) unsigned
nome : varchar(50)
rbonavig crew
film : bigint(20) unsigned
persona : int(10) unsigned
ruolo : smallint(5) unsigned
rbonavig stato
id : tinyint(3) unsigned
nome : varchar(50)
rbonavig collezione
id : int(10) unsigned
nome : varchar(200)
descrizione : varchar(10000)
locandina : varchar(100)
rbonavig utente
id : bigint(20) unsigned
username : varchar(30)
mail : varchar(100)
nome : varchar(50)
gender : tinyint(3) unsigned
data_nascita : date
password : varchar(255)
is_admin : tinyint(3) unsigned
rbonavig persona
id : int(10) unsigned
nome : varchar(50)
gender : tinyint(3) unsigned
immagine : varchar(100)
data_nascita : date
data_morte : date
rbonavig gender
id : tinyint(3) unsigned
nome : varchar(50)