Ingegneria dei Sistemi Web

Michela Minichini, Giulia Vanni

November 2023

GalaxyCineVerse Link al repository su GitHub del progetto

1 Introduzione

In questo progetto è stata realizzata un'applicazione web che consente agli utenti di visionare una lista di film, selezionare quello che più interessa analizzando i relativi dettagli come data e orario, per poi proseguire con la scelta di un posto e l'acquisto di un biglietto.

Una volta selezionato il film che si desidera vedere, cliccando sull'apposito bottone "Scheda Film" sottostante, l'utente può procedere ad effettuare il login al suo account personale e proseguire con la prenotazione di un posto a sedere e l'acquisto di un biglietto.

Se invece ad effettuare il login, cliccando sul bottone "Login", è l'amministratore, allora sarà disponibile una seconda versione dell'applicazione in cui modificare i campi presenti, come per esempio i film disponibili, gli orari e le date.

2 Obiettivo del Progetto

L'obiettivo principale di questo progetto è offrire una piattaforma incentrata sull'utente che semplifica il processo di scoperta, selezione e godimento dei film. Attraverso un'interfaccia web intuitiva, gli utenti possono non solo esplorare un catalogo di film, ma anche procedere agevolmente alla prenotazione dei biglietti e alla scelta dei posti preferiti, tutto dal comfort dei propri dispositivi. Agevolazioni anche per gli amministratori della pagina web, per i quali la realizzazione delle modifiche, come cancellazione o un semplice cambiamento, non è mai stato così semplice. Grazie ad una seconda versione della web app interamente pensata per il compimento di correzioni o variazioni, il lavoro degli amministratori viene così reso efficace e veloce.

Mentre approfondiamo il report, esploreremo le varie caratteristiche del progetto che contribuiscono collettivamente al successo di GalaxyCineVerse.

3 Tecnologie Utilizzate

- Node.js
- npm
- Express.js
- MySQL
- Vue.js
- Vite

- TypeScript
- Vue Router
- Axios
- XAMPP control panel per poter avviare MySQL e accedere al database e Apache per avviare il web server
- Bootstrap per la realizzazione dello stile dell'applicazione web insieme a Strass (SCSS)

4 Struttura del progetto

Il progetto è stato diviso in 2 cartelle principali: frontend e backend. La cartella frontend contiene tutte le dipendenze relative alla parte client mentre quella backend al suo interno ha tutte le dipendenze relative alla parte server. Vediamo nel dettaglio ciò che è stato installato ed utilizzato nell'intero progetto. Prima osserviamo le dipendenze comuni in entrambe le cartelle, poi in ciascuna cartella nello specifico.

4.1 Dipendenze installate e comandi utilizzati

In entrambe le cartelle è stato installato Node.js e di conseguenza npm. Di fatto, sia nella cartella frontend che backend possiamo osservare la presenza dei seguenti elementi:

- package.json: elenco che contiene le dipendenze man mano che vengono installate
- package-lock.json: file che contiene un elenco dettagliato di TUTTE le dipendenze.
- node-modules: cartella che contiene il codice dei moduli installati

4.1.1 Node.js

Sito utilizzato per scaricare Node.js

4.1.2 npm

Node Package Manager (npm) è la più grande libreria (library o registry) di software al mondo. Esso è anche un Package Manager software e Installer.
Per installarlo è necessario installare Node.js (4.1.1). Tutti i pacchetti disponibili sono scaricabili gratuitamente grazie alla Command Line Client (CLI) che npm mette a disposizione. La struttura generale del comando da utilizzare per scaricare i pacchetti è

```
npm install <package>
```

Una volta scaricati tutti i pacchetti necessari a realizzare un determinato progetto, essi vengono riportati in dei files detti package.json, il cui contenuto è obbligatoriamente scritto in formato JSON.

Nel progetto, in entrambe le cartelle, è stato eseguito il seguente comando

```
npm init -y
```

il quale genera un file package.json base che poi in futuro può essere modificato. Si usa l'opzione "-y" per poter generare questo file JSON senza dover rispondere a tutte le domande che vengono proposte al momento della sua generazione.

4.1.3 XAMPP

XAMPP è un web server solution stack package cross-platform (X), gratis e open-source. XAMPP è una distribuzione di Apache completamente gratuita e semplice da installare, contenente MySQL, PHP e Perl. Il pacchetto open source è stato creato per essere estremamente facile da installare e utilizzare. Viene principalmente usato per testare applicazioni web su un host web server locale.

In questo progetto è stato utilizzato avviando Apache e MySQL dal control panel per poter avviare il web server e poter accedere al database.

4.1.4 TypeScript

TypeScript nasce come superset di JavaScript, ossia un'estensione di Javascript che introduce nuove features ed ne espande le capacità. TypeScript dunque aggiunge tipi opzionali a JavaScript che supportano strumenti per applicazioni JavaScript su larga scala per qualsiasi browser, per qualsiasi host, su qualsiasi sistema operativo. TypeScript si compila in JavaScript leggibile e basato su standard.

Nel progetto, dunque in entrambe le cartelle, è stato utilizzato TypeScript.

Sono state installate tutte le dipendeze per TypeScript utilizzate con il seguente comando (unico)

```
npm i -D typescript ts-node-dev @types/node
```

Notiamo, dal flag -D, che sono tutte dipendenze di sviluppo poichè non sono necessarie in fase di esecuzione, ma solo per compilare il programma.
Tra queste dipendenze, tralasciando "typescript" che rappresenta l'installazione di TypeScript, sono presenti "ts-node-dev", utilizzata per compilare TypeScript in tempo reale in fase di sviluppo e "@types/node" che indica i tipi di TypeScript per tutte le API di Node.js (fs, path, ecc.). Permette dunque a TypeScript di funzionare con le API di Node.js

Successivamente, TypeScript è stato inizializzato eseguendo il comando

```
tsc --init
```

4.2 Backend

In questa specifica cartella sono stati installati e utilizzati i seguenti:

4.2.1 Express.js

Express è un web framework per Node.js ed è usato per costruire applicazioni web singole, multi-page o ibride. Inoltre, aiuta a gestire i server e le rotte. Per installare Express è stato eseguito il comando

```
npm install express
```

4.2.2 mysql2

La libreria mysql2 mette a disposizione delle API con le funzionalità più comuni:

- createConnection: istanzia la connessione al database, ricevendo in input un oggetto contenente le informazioni per la connessione come host, user, password e database.
- query: per eseguire una qualsiasi query sql.
- execute: per eseguire query utilizzando i prepared statements, che sono query parametrizzate e che generalmente aiutano a proteggere da attacchi basati su SQL Injection.

Per installare questa libreria è stato utilizzato il comando

```
npm install mysql2
```

4.2.3 connect-history-api-fallback

Middleware per delegare le richieste attraverso una pagina indice specificata, utile per le applicazioni a pagina singola (SPA) che utilizzano l'API HTML5 History.

È un type di TypeScript, come si può notare dalla presenza dell'icona DT accanto al nome del package (su NPM è possibile cercare i pacchetti e controllare il loro tipo grazie alle icone). Dunque, è stato necessario installarlo separatamente eseguendo 2 comandi:

```
npm i connect-history-api-fallback
```

e in seguito

```
npm install --save-dev @types/connect-history-api-fallback
npm i -D @types/PACKAGE_NAME (struttura generale del comando)
```

4.3 Frontend

La cartella "frontend" è stata realizzata con Vite. Di fatto, possiamo osservare la presenza di files generati con la creazione della cartella come "vite.config.ts", ".gitignore", "index.html", "README.md" e "tsconfig.node.json".

4.3.1 Vite

Vite (parola francese per "veloce", pronunciata /vit/) è uno strumento di compilazione che mira a fornire un'esperienza di sviluppo più rapida e snella per i moderni progetti web. È composto da due parti principali:

- Un server di sviluppo che fornisce funzionalità avanzate rispetto ai moduli ES nativi, come ad esempio l'Hot Module Replacement (HMR) estremamente veloce.
- Un comando di compilazione che raggruppa il codice con Rollup, preconfigurato per produrre risorse statiche altamente ottimizzate per la produzione.

Per installare Vite, è stato eseguito il seguente comando

```
npm create vite@latest
```

In questo modo creiamo la cartella progetto, da noi denominata "frontend". Seguendo le istruzioni, si imposterà:

- nome progetto (frontend)
- framework: Vue, in questo caso
- variante: TypeScript, nel nostro caso

Nella cartella "frontend", inoltre, sono stati installati e utilizzati i seguenti:

4.3.2 Volar - estensione per Visual Studio Code

Vue Language Features (Volar) è un'estensione di supporto linguistico costruita per Vue, Vitepress e petite-vue. Si basa su @vue/reactivity per calcolare tutto on-demand, per implementare le prestazioni native del linguaggio TypeScript a livello di servizio.
Per aggiungere questa estensione a Visual Studio Code è necessario installarla dal seguente link Vue Language Features (Volar). In questo modo, si avvierà automaticamente VS Code e sarà possibile scaricare l'estensione.

4.3.3 Vue

Installato grazie a Vite nella scelta del Framework, Vue Vue è un framework JavaScript per la creazione di interfacce utente. Si basa su HTML, CSS e JavaScript standard e fornisce un modello di programmazione dichiarativo e basato su componenti che aiuta a sviluppare in modo efficiente le interfacce utente, siano esse semplici o complesse.

Inoltre, è stato installato Vue Router come una libreria NPM con il seguente comando

```
npm i vue-router
```

4.3.4 Axios

Axios è un client HTTP basato su promesse per node.js e il browser. È isomorfo (= può essere eseguito nel browser e in nodejs con la stessa base di codice). Sul lato server utilizza il modulo http nativo di node.js, mentre sul client (browser) utilizza XMLHttpRequests.

In questo progetto, Axios è stato installato nella cartella "frontend" con il seguente comando

```
npm install axios
```

4.3.5 Strass e Bootstrap - la gestione dello stile

Per quanto riguarda lo stile del progetto, sono stati utilizzati SASS e Bottstrap.

SASS (Syntactically Awesome Style Sheets) è una estensione di CSS che aggiunge potenza ed eleganza al linguaggio di base. Permette agli sviluppatori di utilizzare variabili, regole annidate, mixins, importazioni in linea e altro ancora. Inoltre, SASS aiuta a mantenere ben organizzati grandi fogli di stile e a far funzionare rapidamente i piccoli fogli di stile. Sass è un preprocessore CSS ed è completamente compatibile con tutte le versioni di CSS. In più, riduce la ripetizione di CSS e quindi fa risparmiare tempo.

In questo progetto, SASS è stato installato con il seguente comando

```
npm i sass
```

Bootstrap è un framework CSS gratuito e open-source indirizzato allo sviluppo web front-end reattivo e mobile-first. Contiene modelli di design basati su HTML, CSS e JavaScript per la tipografia, i moduli, i pulsanti, la navigazione e altri componenti dell'interfaccia.
Per l'installazione di Bootstrap è stato eseguito il seguente comando

```
npm i --save bootstrap @popperjs/core
```

4.4 La struttura del progetto - i dettagli

Osserviamo più nel dettaglio il contenuto di ciascuna cartella dopo aver visionato le dipendenze installate in ciascuna di esse.

4.4.1 Backend

Più nello specifico, la cartella "backend" al suo interno racchiude un insieme di ulteriori cartelle

- una cartella "public" che raccoglie tutti i file statici (come le immagini)
- una cartella "sql" che contiene i files per la creazione del database e l'inserimento dei dati al suo interno
- una cartella "src" che al suo interno raccoglie altre cartelle:
  - una cartella "utils", in cui troviamo il file "db.ts" che si occupa di instaurare la connessione al database tramite mysql2. Questo file, dunque, esporterà la connessione al DB con il comando module.exports in modo che possa poi essere importato utilizzando il comando require.

  - una cartella "routes", dove sono presenti un file per ogni router che gestisce le rotte di una determinata entità
  - una cartella "controllers", in cui sono presenti un file per ogni controller che definiscono le funzioni chiamate in corrispondenza delle varie rotte.

- infine, oltre ai files di configurazione della cartella, è presente un file denominato "app.ts", il quale, scritto in TypeScript, presenta diverse informazioni. Esso, infatti, include express, istanzia l'oggetto "app", racchiude tutti i middleware come quello per la gestione dei file statici e mette il server in ascolto su una determinata porta.

4.4.2 Frontend

La cartella "frontend", invece, al suo interno presenta solo una cartella denominata "src", la quale a sua volta comprende altre cartelle:

- una cartella "style", che include tutti i files di stile, scritti con SASS (estensione ".scss")
- una cartella "pages", che al suo interno racchiude tutti i files con estensione ".vue" che rappresentano le varie pagine che compongono complessivamente la pagina principale presente nel file "App.vue", che però è situata al di fuori di questa cartella
- un file denominato "main.ts", il quale, scritto con TypeScript, racchiude varie informazioni. In questo file vengono di fatto importate tutte le pagine presenti nella cartella "pages", varie dipendenze utilizzate come bootstrap e alcuni elementi presi da Vue ("createApp") e Vue Router ("createRouter", "createWebHistory", "Router"). Questi elementi, infatti, vengono usati per instaurare il Router e la WebHistory, che consente di istanziare i diversi percorsi ("path") che compongono le varie parti dell'applicazione
- un file detto "types.ts", che creato per mettere in comunicazione il sito web con il server Express, in modo da ottenere i dati da mostrare nelle pagine presenti nella cartella "pages". Questo file, scritto con TypeScript, contiene tutte le interfacce TypeScript dei dati restituiti dal server.

5 Avviare il progetto

Una volta clonato il progetto da GitHub, seguendo le istruzioni riportate nel "README.md" del repository, sarà sufficiente aprire un editor di codice come Visual Studio Code e aprire due terminali. In un terminale dovrà essere eseguito il comando

```
cd frontend
```

per poter entrare nella cartella "frontend" ed eseguire in seguito il comando

```
npm run dev
```

Allo stesso modo, andranno eseguiti i medesimi comandi nella cartella "backend" dal secondo terminale, dunque

```
cd backend
```

e di conseguenza

```
npm run dev
```

In questo modo, il progetto verrà definitivamente avviato e sarà sufficiente cliccare sul link visualizzabile nel terminale in cui si sono eseguiti i comandi nella cartella "frontend" per poter visionare l'applicazione web.

6 Conclusioni

Per riassumere, abbiamo realizzato un progetto costituito da un frontend che sfrutta Vite, Vue e TypeScript e un backend realizzato con Express e TypeScript.

7 Bibliografia

References

[1] Author One. Title of the article. *Journal Name*, 2023.

[1]