

# Progetto Vacanze

## Web Playlist: Admin & Client

Sviluppo Web Multi-Pagina

### Introduzione

L'obiettivo di questo progetto è creare una Piattaforma di Playlist Musicale personale completa. Il progetto chiede di simulare un vero ambiente di sviluppo web separando la parte pubblica (quella che vedono gli utenti) dalla parte di amministrazione (dove si gestiscono i contenuti).

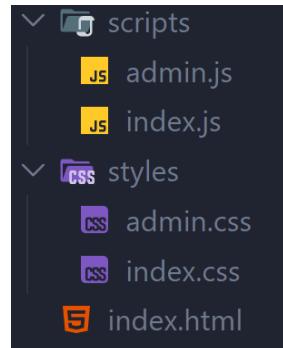
**Non viene fornito alcun HTML/CSS di base, avete quindi libertà totale sull'interfaccia grafica.**

L'applicazione sarà divisa in due interfacce distinte:

1. **Dashboard Admin:** Per inserire e rimuovere le canzoni.
2. **Playlist Pubblica:** Per visualizzare le canzoni e ascoltarle.

### Fase 1: Organizzazione del Progetto

Creare le cartelle e i file per il progetto seguendo **tassativamente** questo schema:



Importante notare che *index.html* è nella cartella root.

### Perché questa struttura?

- **Ordine:** I file CSS e JS sono raggruppati per tipo.
- **URL puliti:** Creando un file “index.html” dentro la cartella admin, nel browser basterà scrivere “[www.example.com/admin](http://www.example.com/admin)” per visualizzare la pagina, anziché “[www.example.com/admin.html](http://www.example.com/admin.html)”. I link del sito risultano più puliti e professionali.
- **Percorsi:** Attenzione ai percorsi: all’interno del file `admin/index.html`, per collegare il CSS che si trova nella cartella `styles`, è necessario "tornare indietro" di un livello usando `../styles/admin.css`.

## Fase 2: Salvataggio dei dati con LocalStorage

Siccome ci sono due pagine HTML separate, le variabili di una pagina non sono visibili nell’altra. Per farle comunicare e permettere che lavorino sul medesimo set di dati, bisogna utilizzare il **localStorage** del browser come se fosse un database condiviso. Come visto durante l’esercitazione, recuperare i dati dal localStorage tramite questa riga di codice:

```
const example = JSON.parse(localStorage.getItem("example"))
|| [];
```

Quando dobbiamo invece salvare nuovi dati, utilizzare la seguente istruzione:

```
localStorage.setItem("example", JSON.stringify(example));
```

Ogni canzone deve ora avere un **ID univoco** per poterla distinguere da tutte le altre (utile per la cancellazione).

Per generare un ID sicuramente univoco ogni volta che viene aggiunta una nuova canzone, è possibile utilizzare `Date.now()` al momento della creazione e salvarne il valore come **ID** della canzone. `Date.now()` è una funzione JavaScript utilizzata per la gestione delle date e ritorna il numero di millisecondi passati dalla data 1 gennaio 1970.

Fatta questa premessa, ciascuna canzone deve essere salvata in formato JSON seguendo questa struttura:

```
{
  "id": 1703254412399, // Date.now(),
  "titolo": "Black Or White",
  "artista": "Michael Jackson",
  "genere": "Pop Rock",
  "link": "https://..."
```

## Fase 3: Il pannello admin (Backend)

Lavorare sui file `admin/index.html`, `styles/admin.css` e `scripts/admin.js` e implementare le seguenti funzionalità:

- Bloccare l'accesso alla pagina admin tramite un `prompt()`: il pannello deve essere inizialmente nascosto (per esempio tramite CSS); se la password inserita dall'utente corrisponde a quella salvata nel codice, il pannello viene mostrato.

*Nota: in un progetto reale non si salvano mai password in chiaro nel codice, ma ai fini di questo progetto va bene utilizzare una variabile come segue:*

```
const password = "sonoLaPassword123";
```

- Creare un Form per l'inserimento, con i seguenti input:
  - **titolo** (tag input di tipo “text”)
  - **artista** (tag input di tipo “text”)
  - **genere** (tags select&option)
  - **link** (tag input di tipo “url”)
  - **Aggiungi** (tag button di tipo “submit”)
- Quando viene premuto il bottone “Aggiungi”:
  1. Validare i dati.
  2. Creare l'oggetto canzone aggiungendo l'**id** univoco generato tramite `Date.now()` agli altri valori presi dal form.
  3. Aggiungere il nuovo oggetto al localStorage.
- Sotto il form, mostrare una *Lista sintetica* delle canzoni già inserite. Accanto a ogni canzone, aggiungere un pulsante “Elimina” con **ID** pari all'identificativo generato tramite `Date.now()`. In questo modo, è possibile associare ciascun bottone “Elimina” alla rispettiva canzone.
- Implementare una funzione “`deleteSong(id)`” che, dato l'**ID** della canzone preso dai bottoni appena creati, la rimuova dall'array nel localStorage.

## Fase 4: La pagina pubblica (Frontend)

Lavorare sui file `index.html` (nella root), `styles/index.css` e `scripts/index.js` e implementare le seguenti funzionalità:

- Al caricamento della pagina, lo script deve prelevare i dati dal localStorage.
- Per ogni canzone trovata nel localStorage, mostrarla appropriatamente nella pagina con stili definiti nel file `styles/index.css`. In particolare, ciascuna card deve mostrare:
  1. titolo della canzone
  2. autore
  3. tag `<a>` con href pari al link specificato nel form admin

Non mostrare il genere della canzone.

- Implementare, a scelta, una tra le seguenti opzioni:
  1. applicare un background color in base al genere delle canzoni mostrate
  2. mostrare un'icona in base al genere della canzone. Potete utilizzare icone prese dal seguente sito: <https://icons8.com/>
  3. mostrare le varie canzoni raggruppandole per colonne in base ai vari stili

## Fase 5: Funzionalità “Random”

Nel file `index.html`, aggiungere un bottone “Random”. Una volta cliccato, prendere casualmente dalla lista di canzoni salvate un link e aprirlo automaticamente in una nuova pagina.

Per aprire il link tramite JavaScript, utilizzare la seguente funzione:

```
window.open("https://www.example.com");
```

## Fase 6: Pubblicazione

Creare il proprio account gratuito su <https://it.altervista.org/> e pubblicare il proprio sito, caricando tutti i file sul pannello di controllo di Altervista.