Counter App — Presentazione Progetto

Applicazione web sviluppata in JavaScript puro con SCSS e Bootstrap

Autore: Cristiano D’Alterio

Data: 02/09/2025

# Link utili

**Repository GitHub:** https://github.com/CristianoDAlterio/counter-app  
**Deploy (Firebase Hosting):** https://counter-app-cristiano-f0d0a.web.app/

# Descrizione

Questo documento presenta il progetto **Counter App**, una semplice applicazione web che simula un contatore con pulsanti “+” e “−“. L’obiettivo didattico è dimostrare la manipolazione del DOM con JavaScript puro (senza framework), l’uso di SCSS per lo stile e una struttura di progetto professionale, completa di repository GitHub e deploy su Firebase Hosting.

# Funzionalità principali

* Valore iniziale impostato a 0 all’apertura della pagina.
* Pulsanti “+” e “−“ per incrementare/decrementare il valore.
* Blocco del contatore a 0 (non può andare in negativo).
* Interfaccia generata dinamicamente via JavaScript (manipolazione del DOM).
* Stile con **Bootstrap 5** e **SCSS** modulare (parziali, variabili, mixin).
* Struttura del codice modulare con `script.js` + `modules/counter.js`.
* Immagini e screenshot organizzati in `assets/img/`.

# Struttura del progetto

counter-app/  
│── index.html  
│── assets/  
│ ├── css/  
│ │ └── style.css  
│ ├── scss/  
│ │ ├── style.scss  
│ │ ├── \_variables.scss  
│ │ ├── \_mixins.scss  
│ │ ├── \_base.scss  
│ │ ├── \_layout.scss  
│ │ └── \_components.scss  
│ ├── js/  
│ │ ├── script.js  
│ │ └── modules/  
│ │ └── counter.js  
│ └── img/  
│ ├── app\_0.jpg  
│ └── app\_count.jpg  
│── README.md  
│── firebase.json  
│── .firebaserc

# Scelte tecniche e architettura

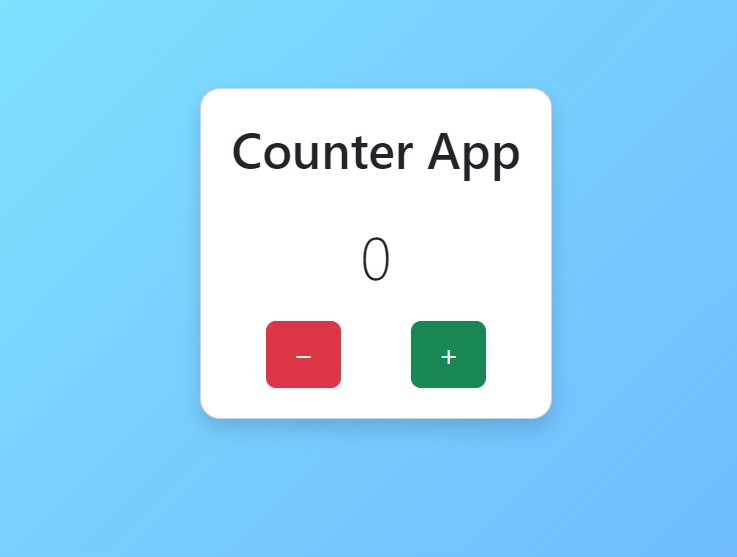
* **JavaScript puro** per tutta la logica: niente jQuery o framework (React/Angular/Vue).
* UI del counter creata dinamicamente con `document.createElement`, `appendChild`, `addEventListener`.
* Separazione logica: `script.js` (bootstrap dell’app) importa `createCounter` da `modules/counter.js`.
* Stile tramite **Bootstrap 5** (classi `btn`, `btn-success`, `btn-danger`, `d-flex`, `justify-content-around`, ecc.).
* Background stellato impostato via SCSS, mantenendo la card del counter bianca per massima leggibilità.
* Deploy con Firebase Hosting, link incluso in README e in questo documento.

# Spiegazione del codice

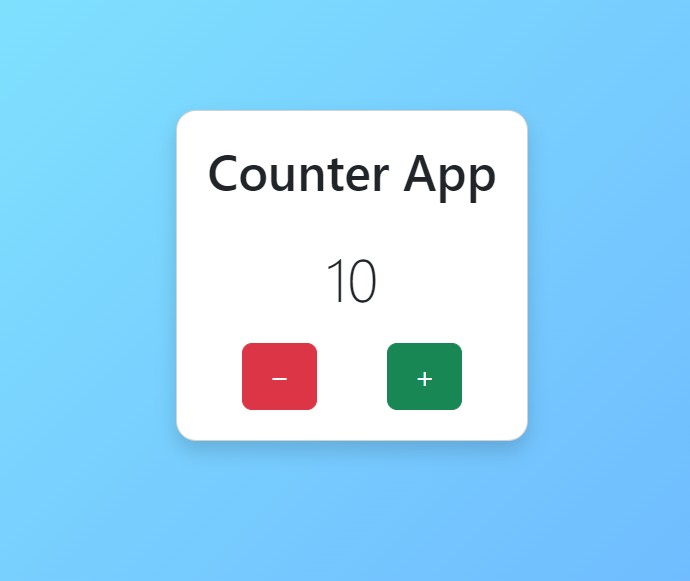
Il codice è suddiviso in due parti principali: **script.js** e **counter.js**  
  
- In `script.js` si trova il punto di ingresso dell’applicazione: al caricamento del DOM, viene chiamata la funzione `createCounter` passando l’ID del contenitore (`app`). Questo file funge da bootstrap e mantiene la logica separata dalla UI.  
  
- In `counter.js` è definita tutta la logica del contatore. La variabile `counter` mantiene lo stato corrente del contatore, inizializzato a 0. La funzione `createCounter` crea dinamicamente tutti gli elementi dell’interfaccia: titolo, valore del contatore e pulsanti + e −.  
  
 • Con `document.createElement` vengono generati gli elementi del DOM (div, h1, h2, pulsanti).  
 • Le classi Bootstrap (es. `btn btn-success btn-lg`) vengono assegnate per garantire lo stile.  
 • I pulsanti hanno gestori di eventi (`addEventListener`) che modificano il valore della variabile `counter` e aggiornano la visualizzazione chiamando `updateCounter`.  
 • Il pulsante − è protetto da una condizione (`if (counter > 0)`) che impedisce di scendere sotto lo zero.  
  
- Infine, gli elementi creati vengono assemblati e inseriti nel contenitore della pagina (`app`). Questo approccio assicura che l’interfaccia sia completamente dinamica e gestita dal codice JavaScript, rispettando il requisito di non avere markup statico per i pulsanti o il valore del contatore.  
  
In sintesi, il funzionamento generale è:  
1. All’avvio della pagina, il contatore parte da 0.  
2. L’interfaccia (card con titolo, valore e pulsanti) viene costruita via JavaScript.  
3. Con un click su “+” il valore aumenta, con un click su “−“ diminuisce ma non va sotto 0.  
4. La UI è aggiornata in tempo reale con il nuovo valore.

# Screenshot

Valore iniziale (0)



Valore incrementato



# Mappatura ai criteri di valutazione

✔ Funzionamento pulsanti +/-

* Il contatore si incrementa/decrementa correttamente. Il decremento è bloccato a 0 (no valori negativi).

✔ Implementazione via JavaScript della UI e del valore

* L’interfaccia è creata dinamicamente via DOM APIs; il valore è mostrato/aggiornato in tempo reale.

✔ Qualità del codice e suddivisione logica

* Codice modulare: `script.js` (bootstrap) e `modules/counter.js` (logica UI e stato). Funzioni piccole e leggibili.

✔ Struttura del progetto (HTML/CSS/JS/img)

* File organizzati per cartelle (`assets/css`, `assets/scss`, `assets/js`, `assets/img`). `index.html` minimale in root.

✔ Repository GitHub e README

* Repo pubblica con README dettagliato (funzionalità, struttura, screenshot, istruzioni, link al deploy).

# Conclusione

Il progetto rispetta pienamente i requisiti richiesti: uso di JavaScript puro, interfaccia generata via DOM, struttura modulare e professionale, repository GitHub completo e deploy su Firebase Hosting. Gli screenshot e i link forniti consentono una verifica immediata del funzionamento.