

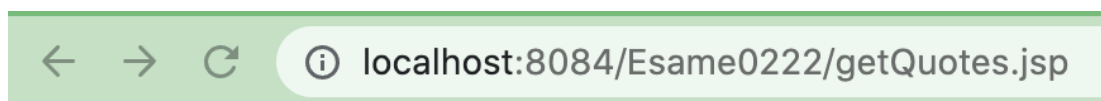
Introduzione alla Programmazione Web – Appello di Febbraio '22

Un sito di e-banking permette agli utenti di acquistare e vendere azioni.

- 1) Si crei una pagina di controllo della Borsa, che ogni secondo chiede al server web di aggiornare il valore delle azioni. Sono presenti due titoli (ENEL e WEBUILD) che al lancio iniziale della webApp valgono rispettivamente 100.00 e 200.00. Ogni aggiornamento modifica il valore di ciascuna azione (in modo indipendente) di una percentuale casuale compresa tra -1% e +1%, arrotondando a due cifre decimali il risultato. La pagina mostra con continuità (ma **senza ricaricarsi**) il valore corrente delle azioni.



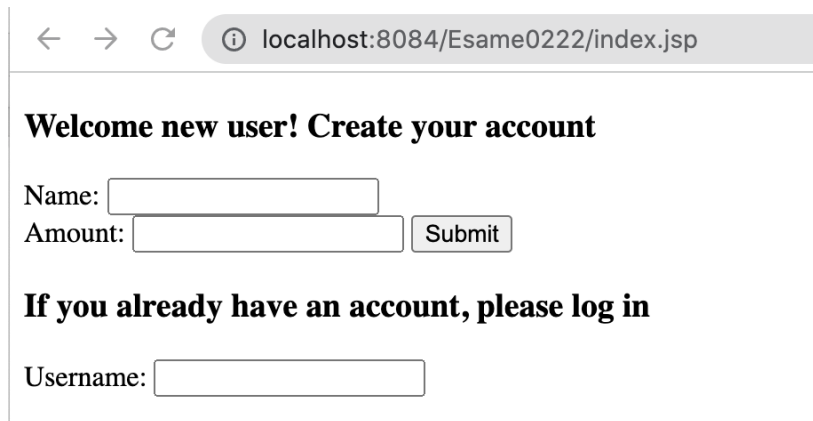
- 2) Nella pagina sono presenti due bottoni, start e stop. Solo uno dei due è abilitato. Inizialmente non si ha aggiornamento dei dati, che viene avviato solo quando si preme il bottone di Start ('unico abilitato inizialmente).
- 3) Contestualmente il bottone di start viene disabilitato, e si abilita quello di stop,
- 4) Ora premendo il bottone di stop, l'aggiornamento dei dati cessa (la Borsa chiude e non vi sono più variazioni di valore dei titoli): si disabilita il bottone di stop e si riabilita quello di start.
- 5) Premendo il bottone di start la Borsa riprende le contrattazioni: l'aggiornamento continuo riprende, ripartendo dai valori di chiusura (gli ultimi valori calcolati).
- 6) Una seconda pagina presenta due bottoni: uno per ENEL ed uno per WEBUILD. Premendo il bottone si ottiene il valore dell'azione nell'istante corrente. Se il valore è maggiore o uguale al valore di partenza (punto 1), questo viene mostrato in italico verde, altrimenti in grassetto rosso. **Si gestiscano gli stili con classi CSS ad hoc.**



- 7) Se la seconda pagina viene richiamata prima del primo avvio della Borsa, non vi sono ancora quotazioni, e quindi viene dato il messaggio "Sistema non ancora attivo"
- 8) Se la seconda pagina viene richiamata dopo il primo avvio della Borsa, ma a Borsa chiusa, viene comunque dato il valore corrente dell'azione (ovvero l'ultimo valore assunto mentre le contrattazioni erano attive).

Vi è poi la pagine utente. Un utente può:

- 9) Accedere tramite il proprio username, se già registrato (non serve utilizzare passwords), oppure registrarsi dando il proprio username (che non deve essere già esistente) e contestualmente depositare una somma di denaro (non serve preoccuparsi di omonimie, o utilizzare passwords). **I campi devono avere le seguenti restrizioni: username non nullo, amount non nullo e numerico.** Deve essere anche possibile che gli utenti siano automaticamente riconosciuti (tramite cookies o sessions), e passino quindi automaticamente alla visualizzazione successiva.



← → ↻ ⓘ localhost:8084/Esame0222/index.jsp

Welcome new user! Create your account

Name:

Amount:

If you already have an account, please log in

Username:

- 10) Passata l'identificazione, l'utente collegato vede lo stato del proprio conto (soldi disponibili, quantità e valore corrente delle azioni possedute) e può acquistare o vendere azioni (scegliendone il tipo da un popup menu, e specificandone il numero). La pagina viene poi ricaricata con il nuovo stato del conto.
- 11) I dati dell'utente (ammontare del conto, tipo e numero di azioni possedute) sono mantenuti server-side.

Suggerimenti per la soluzione

Per disabilitare l'aggiornamento continuo di una pagina, si veda:

<https://stackoverflow.com/questions/109086/stop-setinterval-call-in-javascript>

(Se non avete un vostro account su stackoverflow, potete accedere con le seguenti credenziali: Email scalaenne@gmail.com, pw esame123)

Per aggiungere e togliere dinamicamente classi css:

https://www.w3schools.com/howto/howto_js_add_class.asp

La sufficienza si ottiene anche senza completare il progetto, ma quanto consegnato **deve girare** per essere valutato.