Relazione progetto "UmbriaSagre"

PROGRAMMAZIONE 3

ANNO ACCADEMICO 2021/2022

Docente: Marco Baioletti

A cura di:

ANDREA IMPARATO, 323840

1. Requisiti funzionali dell'applicazione

UmbriaSagre è un'applicazione Web il cui scopo è promuovere l'interazione tra gli organizzatori di eventi regionali (in particolare sagre) e il pubblico locale. Si basa sul paradigma Client-Server e prevede due tipologie di utilizzatori: gli **Organizzatori** e gli **Utenti semplici**. L'applicazione si basa innanzitutto su tre concetti fondamentali:

- Tutti gli utilizzatori dell'App devono registrarsi e autenticarsi
- Gli Organizzatori pubblicano le informazioni relative a eventi che avranno luogo nell'immediato futuro in modo che gli Utenti semplici possano visualizzarle, manifestando il loro eventuale interesse.
- Organizzatori e Utenti semplici possono avvalersi di un sistema di posta personale molto elementare per scambiarsi brevi messaggi.

Intorno a questi obiettivi fondamentali ruotano tutte le funzionalità specifiche che l'applicazione mette a disposizione degli utilizzatori. Esse sono:

- **Registrati** , offerta ai visitatori generici affinché possano creare un nuovo profilo.
- **Accedi**, offerta ai visitatori generici affinché possano autenticarsi.
- **Ricerca**, offerta agli utenti semplici affinché possano visualizzare l'elenco degli eventi a cui possono iscriversi. L'elenco include anche dei filtri che permettono di restringere la lista sulla base di informazioni quali la data e il comune.
- **Iscrizione**, offerta agli utenti semplici affinché possano iscriversi a un evento a cui sono interessati. L'iscrizione determina l'inserimento dell'evento in una lista a parte (i "Preferiti").
- **Preferiti**, offerta agli utenti semplici affinché possano visualizzare l'elenco degli eventi a cui si sono iscritti. Anche su questo elenco possono essere applicati dei filtri.

- **Disiscrivi**, offerta agli utenti semplici affinché possano annullare l'iscrizione a un evento (l'evento viene anche rimosso dalla lista preferiti).
- **Scrivi all'organizzatore**, offerta agli utenti semplici affinché possano inviare un messaggio privato all'organizzatore di un evento a cui si sono iscritti.
- **Messaggi**, offerta sia agli utenti semplici che agli organizzatori affinché possano accedere alla loro casella di posta personale e visualizzare i messaggi ricevuti.
- **Visualizza messaggio**, offerta sia agli utenti semplici che agli organizzatori affinché possano accedere al contenuto di un messaggio specifico.
- **Cancella messaggio**, offerta sia agli utenti semplici che agli organizzatori affinché possano eliminare un messaggio dalla loro casella di posta.
- **Crea risposta**, offerta sia agli utenti semplici che agli organizzatori affinché possano creare un messaggio di risposta a seguito della visualizzazione di un messaggio ricevuto.
- **Nuovo evento**, offerta agli organizzatori affinché possano inserire nel sistema i dati relativi a un nuovo evento appena organizzato.
- **Eventi organizzati**, offerta agli organizzatori affinché possano visualizzare la lista degli eventi da loro creati. Anche questa lista include filtri.
- **Elimina evento**, offerta agli organizzatori affinché possano eliminare dal sistema i dati relativi a un evento da loro creato.
- **Scrivi agli iscritti**, offerta agli organizzatori affinché possano creare un messaggio che il sistema provvederà a recapitare a tutti gli utenti che risultano iscritti a un evento da loro creato.
- **Visualizza dettagli evento**, offerta sia agli organizzatori che agli utenti semplici affinché possano visualizzare alcuni dettagli aggiuntivi di un evento specifico (che normalmente non vengono mostrati in tabella).
- **Visualizza locandina**, offerta sia agli organizzatori che agli utenti semplici affinché possano visualizzare la locandina associata a un evento specifico.
- **Esci**, offerta a tutti gli utilizzatori autenticati affinché possano effettuare il log-out e terminare la sessione.

2. Linguaggi, tecnologie e librerie

Il Front-End dell'applicazione è scritto usando una semplice combinazione di HTML, CSS e JavaScript. Il Back-End si compone invece di due livelli: un Server basato su Node.js e una base di dati (a cui accede il Server) implementata usando PostgreSql. Lo Script eseguito con Node.js fa uso dei seguenti moduli:

- **express** (visto anche a lezione), per la realizzazione del Server
- **pg** , per interagire con il database postgresql
- **crypto**, per eseguire il sorteggio di un numero casuale
- **bcrypt**, per effettuare l'hashing delle password
- **cookieParser**, per effetturare il parsing dei cookie dei visitatori
- **rateLimit**, per impostare un tetto massimo alle chiamate dallo stesso IP
- **multer**, per salvare su disco eventuali immagini allegate agli eventi
- **fs**, per eliminare dal disco eventuali immagini allegate a eventi cancellati
- **methodOverride**, per gestire richieste PUT, PATCH e DELETE che vengono generate tramite invio di form html (che nativamente supporta solo i metodi GET e POST).

3. Descrizione ad alto livello del Server

Essendo l'App di dimensioni relativamente contenute, il Front End fa uso di un singolo documento html renderizzato preventivamente tramite Ejs affinché ne vengano fornite versioni diverse a seconda del visitatore (che a seconda dei casi può essere Organizzatore, Utente semplice o Sconosciuto). Una delle route di express (**GET "/"**) è riservata allo scaricamento di questo documento dal Server. Tutte le altre servono invece a implementare API che i Client possono sfruttare per interagire con le informazioni conservate nel database dell'Applicazione: a seconda dei casi queste API possono o produrre una modifica nel database oppure interrogarlo e restituire al Client il risultato dell'interrogazione in formato JSON. L'uso del formato JSON e la semantica scelta per le route di express rientrano anche nell'ottica di uniformare il più possibile le API allo standard REST. Ognuna delle funzionalità offerte ai Client precedentemente illustrate è legata ad (almeno) una di queste API, nello specifico:

- **GET "/eventi"** viene usata da **Ricerca** per interrogare il database e ottenere la lista di eventi a cui l'utente può iscriversi. Viene anche usata da **Eventi organizzati** per ottenere la lista di eventi creati dall'organizzatore che ha invocato la funzionalità.
- **GET "/evento"** viene usata da **Visualizza dettagli evento** per interrogare il database e ottenere tutte le informazioni relative a un singolo evento di cui il visitatore vuole conoscere i dettagli aggiuntivi.
- **GET "/comuni"** viene usata da **Nuovo evento** per interrogare il database e ottenere la lista di tutti i comuni possibili: con questa lista viene poi creato nel Client il menù a tendina che l'organizzatore usa per selezionare il comune in cui sta organizzando il nuovo evento. Più generalmente, viene anche usata da tutte le funzionalità che producono una tabella in cui è incluso un filtro che guarda al comune in cui ha luogo ogni evento (basato sempre su un menù a tendina in cui devono figurare tutti i comuni).
- **GET "/iscrizioni"** viene usata da **Preferiti** per interrogare il database e

ottenere la lista di eventi a cui l'utente è iscritto.

- **GET "/messaggi"** viene usata da **Messaggi** per interrogare il database e ottenere la lista di messaggi il cui destinatario risulta essere il visitatore.
- **GET "/messaggio"** viene usata da **Visualizza messaggio** per interrogare il database e ottenere le informazioni relative al messaggio che il visitatore vuole visualizzare. Produce anche una modifica nel database etichettando il messaggio in questione come "già letto almeno una volta".
- **GET "/isUsernamePreso"** viene usata da **Registrati** per interrogare il database e scoprire se lo username scelto per un nuovo profilo è già in uso.
- **GET "/isNuoviMessaggi"** viene usata in fase di inizializzazione della Home page per interrogare il database e scoprire se esistono dei messaggi non letti il cui destinatario risulta essere il visitatore.
- **POST "/utente"** viene usata da **Registrati** per inserire nel database una nuova tupla contenente le informazioni relative al nuovo profilo appena creato.
- **POST "/evento"** viene usata da **Nuovo evento** per inserire nel database una nuova tupla contenente le informazioni relative al nuovo evento appena creato.
- **POST "/iscrizione"** viene usata da **Iscrizione** per inserire nel database una nuova tupla rappresentante l'iscrizione all'evento.
- **POST "/messaggio"** viene usata da **Scrivi all'organizzatore** per inserire nel database una nuova tupla rappresentante il nuovo messaggio appena scritto. Viene anche usata da **Crea risposta** per lo stesso fine.
- **POST "/broadcast"** viene usata da **Scrivi agli iscritti** per inserire nel database una sequenza di nuove tuple rappresentante il nuovo messaggio appena creato (per ogni tupla il destinatario sarà diverso, fino a coprire tutti gli iscritti all'evento).
- **PUT "/utente"** viene usata da **Accedi** per modificare la tupla relativa al profilo che ha eseguito il log-in (inserendo i dati relativi alla sessione appena creata).
- **DELETE "/evento"** viene usata da **Elimina evento** per cancellare dal database la tupla relativa all'evento che l'organizzatore vuole eliminare.
- **DELETE "/iscrizione"** viene usata da **Disiscrivi** per eliminare dal database la

tupla rappresentante l'iscrizione all'evento da cui l'utente vuole disiscriversi.

- **DELETE "/messaggio"** viene usata da **Cancella messaggio** per rimuovere dal database la tupla rappresentante il messaggio da eliminare.
- **DELETE "/sessione"** viene usata da **Esci** per modificare la tupla del database associata al profilo che sta eseguendo il log-out, azzerando le informazioni relative alla sessione in corso.

Il Server fa inoltre uso di quattro funzioni di supporto che non vengono esposte all'esterno.

- La prima di queste è la funzione **sessionCheck**. Questa funzione ha lo scopo di identificare un Client che ha chiamato un'API permettendo la sua classificazione in Utente semplice, Organizzatore, Sconosciuto o Non valido. L'identificazione avviene confrontando le informazioni contenute nei cookie del Client con le informazioni relative alle sessioni attive che risiedono nel database. Sulla base del risultato l'accesso all'API può essere concesso o negato.
- La seconda funzione è **gestioneErrori**, ed entra in gioco quando si verificano errori o situazioni anomale. Ad ogni messaggio personalizzato di errore usato dal Server la funzione associa un codice HTTP che ne rifletta il più possibile la natura per dare un valore al campo status della risposta. La funzione procede poi a renderizzare tramite ejs un documento html speciale che viene usato per notificare il Client dell'errore. In determinate situazioni (dipendenti dalla natura dell'errore) la funzione procede anche a cancellare i cookie del Client (che dovrà quindi effettuare nuovamente il log-in).
- La terza funzione è **puliziaDatabase**. La sua invocazione ha luogo a intervalli regolari in virtù di una chiamata SetInterval che viene operata subito dopo la creazione della connessione al database. Essa si fa carico di eliminare periodicamente dal database le tuple associate a messaggi o eventi molto vecchi al fine di liberare spazio.
- La quarta funzione è **convertiData**, e si usa per effettuare un post-processing sulle date estratte dal database per riportarle alla forma YYYY-MM-DD.