Gestione Eventi

JAITA116 Project Work

Progettare un'applicazione web utilizzando le tecnologie trattate a lezione (Spring, Javascript, Bootstrap, MySql) che permetta la collaborazione di diversi utenti nel creare, aggiornare e gestire una piattaforma di ORGANIZZAZIONE EVENTI. I dati relativi agli utenti sono conservati in una base di dati di nome "*projectwork*", contenente una tabella di nome "utenti". Ciascun utente sarà rappresentato attraverso un record della suddetta tabella. Il file registrazione.sql allegato in appendice contiene le istruzioni SQL per la definizione della base di dati descritta.

Front-end
Sarà necessario inventare il cliente immaginario che richiede la piattaforma, creando una brand identity (nome, logo, palette di colori, font, ecc…) che detti poi lo stile del sito internet.
Per ogni pagina dell'applicazione sarà necessario creare, usando Bootstrap, un layout responsivo che contenga almeno gli elementi richiesti dalla consegna e che sia progettato con buon senso dal punto di vista della User Experience.

Back-end
I dati relativi agli utenti sono conservati in una base di dati di nome "*projectwork*", contenente una tabella di nome "*utenti*". Ciascun utente sarà rappresentato attraverso un record della suddetta tabella.
Il file *projectwork.sql* allegato in appendice contiene le istruzioni SQL per la definizione della base di dati descritta.

Utenti

Esistono 3 tipologie di utenti, identificabili attraverso il campo "*ruolo*" della tabella utente:

1. **Amministratore** (es. Ruolo="ROLE_ADMIN").
   Questi utenti possono caricare nuovi eventi (Concerto, film, ecc..) specificandone le caratteristiche (luogo, locandina, caratteristiche..) e le disponibilità.

2. **Utente Registrato** (es. Ruolo="ROLE_USER").
   Può registrarsi al portale per prenotare un evento disponibile in città scegliendo da una lista divisa in categorie.
   Può terminare una prenotazione.
   Può modificare una prenotazione (cambio evento, cambio data) (facoltativo).

3. **Utente pubblico**, non registrato (non presente nella tabella utente). Questo utente può solo vedere gli eventi disponibili: lista totale di tutti gli eventi e funzione ricerca per categoria disponibilità, tipologia, luogo o data.

Non è richiesto implementare procedure di inserimento di nuovi utenti nell'archivio, ma soltanto le necessarie procedure di autenticazione attraverso un form di login

Eventi disponibili

L’evento è costituito da una serie di informazioni:

1. evento_id
2. tipologia (Concerto, film, ecc..)
3. caratteristiche
4. descrizione
5. luogo evento
6. coordinate
7. disponibilita
8. posti
9. data evento
10. locandina (immagine)
11. prezzo listino

Esempio piatto

1. 3
2. Concerto
3. Prenotazione Obbligatoria
4. Rassegna Hard Rock, il meglio degli anni ‘70
5. via Carlo Alberto 22, Torino
6. 41.9002061,12.4900739,17 (latitudine,longitudine)
7. 50
8. true
9. 2023-07-23
10. hardRock.jpg
11. 33.5

Pagina Home

Tale pagina deve contenere:

1. Uno slider
2. L'elenco degli eventi disponibili
3. un link al pannello di lavoro (visibile/raggiungibile solo da un utente amministratore)
4. un link per il logout o per il login a seconda che l'utente abbia già superato o meno le procedure di autenticazione
5. Testo che racconta di cosa tratta il sito

Carousel scorrevole

Si tratta di un campo di testo e immagine scorrevole, aggiornato dinamicamente, in funzione del contenuto dell'archivio degli eventi disponibili. Tale campo di testo dovrà contenere l'elenco di tutti gli eventi disponibili.
Usare Slick o Swiper JS.

In tale elenco, ciascun evento dovrà essere rappresentato dalla sequenza:
***Tipologia – Descrizione – Luogo – link alla pagina evento.html con passaggio dell'id evento come parametro di una richiesta http di tipo GET *** .

Disponibilità evento (opzionale) (disponibilita.html)

Dovrà essere costituito da un elenco contenente eventi aggiornati dinamicamente (facoltativo: in funzione della data e dell'ora). Suddiviso in 2 sezioni, la prima conterrà gli eventi disponibili, la seconda conterrà l'elenco degli eventi attualmente terminati.

Ciascun evento dovrà essere rappresentato attraverso gli elementi:
Tipologia – Descrizione – Luogo,
visualizzabile tramite richiesta http di tipo GET.

Pagina del singolo evento(evento.html)

Tale pagina viene prodotta dinamicamente in funzione dell'ID dell'evento ricevuto come parametro di una richiesta di GET. Qualora la pagina venisse richiesta senza nessun parametro, l'utente deve essere rediretto verso la pagina home.

Tale pagina dovrà contenere tutte le informazioni disponibili sull'evento, opportunamente impaginate, e una mappa con il marker sulla posizione dell'evento e se disponibile alla prenotazione.

Pannello di lavoro (pannello.html)

Protetta da un controllo per l'autenticazione, accesso limitato ad utenti con ruolo di amministratore, deve mostrare i seguenti contenuti:

1. Se l'utente non ha già effettuato le operazioni di login, deve essere visualizzato un form di login in cui inserire UserID e password, per poi essere reindirizzati, una volta autenticati, alla stessa pagina protetta dalla quale è scaturita la richiesta di login.
2. Se l'utente è di tipo base "ROLE_USER" deve essere reindirizzato a una pagina di errore (errore.html) contenente un messaggio che specifichi che l'utente non dispone di diritti sufficienti alla modifica degli eventi e un link per il logout.
3. Se l'utente è di tipo admin ("ROLE_ADMIN") deve essere visualizzato un pannello di amministrazione contenente le 3 seguenti funzionalità:
    a. Inserimento di un nuovo evento– link a inserisci.html
    b. Elenco degli eventi amministrabili – link a modifica-evento.html
    c. Logout

Pagina di inserimento di un evento(inserisci-evento.html)

Protetta da un controllo per l'autenticazione. Deve prevedere un form per l'inserimento di un evento nell'archivio e un bottone di logout attraverso il quale tornare alla pagina home.
Tale form dovrà contenere i seguenti campi:
tipologia, caratteristiche, descrizione, luogo evento, coordinate, disponibilita, data evento, prezzo, locandina immagine (OPZIONALE: upload di un immagine) ,URL di una pagina web esterna all'applicazione per cercare indirizzo e coordinate gps (es. OpenStreetMap).
(facoltativo - implementare inserimento dinamico di indirizzo e coordinate tramite javascript)

Attraverso questa pagina l'amministratore deve poter inserire le informazioni relative ad un nuovo evento, che pertanto dovranno essere visibili nella home e attraverso la pagina del singolo evento.

Elenco eventi amministrabili (view-eventi.html)

Protetta da un controllo per l'autenticazione. Deve prevedere un bottone di logout attraverso il quale tornare alla pagina home. Deve permettere la visualizzazione di una lista con una riga per ciascun evento amministrabile e un corrispondente checkbox che, se spuntato, rende visibile l'evento all'utente.

Se sono presenti più di 10 eventi, la scansione dell'elenco deve essere ripartita su più pagine generate dinamicamente (eventi da 1 a 10, da 11 a 20 ecc.).
A fianco di ogni evento dell'elenco devono essere presenti due pulsanti con testo "modifica" e "cancella", attraverso i quali sarà possibile attivare le procedure di modifica e cancellazione dell'evento.

1. La procedura di modifica, dovrà prevedere la visualizzazione di un form, precompilato con i dati relativi all'evento selezionato, ma editabile in tutte le sue componenti tranne evento_id, un pulsante di reset che ripristinerà i dati, e uno di submit per i nuovi dati.

2. La procedura di cancellazione prevederà l'eliminazione del record relativo all'evento dall'archivio di quelli disponibili, e conseguentemente da tutte le componenti dinamiche che ne permettevano la visualizzazione.

3. L'aggiunta e modifica di un evento devono prevedere la possibilità di inserire un'immagine da caricare in una cartella del server (opzionale), e il salvataggio del nome dell'immagine nel relativo campo del database, che verrà successivamente utilizzata nell'src del tag dell'immagine

Opzionale:

1. Prenotazioni
    a. view-prenotazioni.html
        i. CRUD (create-read-update-delete)

Logout

Deve prevedere la cancellazione di tutte le informazioni che permettono il riconoscimento dell'utente dell'applicativo e il ritorno alla pagina home

Pagina utente (utente.html)

Protetta da un controllo per l'autenticazione.
Deve contenere l'elenco ordini di un utente con la possibilità modifica e/o cancellare la prenotazione, ed un eventuale elenco prenotazioni chiuse/terminate.

Appendice A

Si dispone di una base di dati creata attraverso questo file di definizione:
(modificare a piacere)

--- *projectwork*.sql ---

```
CREATE DATABASE  IF NOT EXISTS `projectwork`;
USE `projectwork`;

#CREATE USER 'app_generation'@'localhost' IDENTIFIED BY 'generation_2022';
GRANT ALL ON projectwork.* TO 'app_generation'@'localhost';
FLUSH PRIVILEGES;

CREATE TABLE IF NOT EXISTS `utenti` (
  `utente_id` int NOT NULL AUTO_INCREMENT,
  `nome` varchar(75) DEFAULT NULL,
  `cognome` varchar(75) DEFAULT NULL,
  `data_nascita` date DEFAULT NULL,
  `email` varchar(50) NOT NULL,
  `password` varchar(20) NOT NULL,
  `ruolo` enum('RUOLO_ADMIN','RUOLO_UTENTE') NOT NULL,
  PRIMARY KEY (`utente_id`),
  KEY `k_email` (`email`)
);

INSERT INTO `utente`(nome,cognome,dataNascita,email,password,ruolo)
VALUES ('Paolo','Rossi','1994-06-07','admin@email.com','admin','RUOLO_ADMIN'),
('Carlo','Verdi','2001-03-19','utente@email.com','utente','RUOLO_UTENTE');
```

Appendice B

- Trello, Figma, Canva, Unsplash
    - Organizzazione e assegnazione task https://trello.com/
    - https://www.figma.com/
    - https://www.canva.com/
    - https://unsplash.com/

- Github per condivisione codice
- Carousel
    - https://kenwheeler.github.io/slick/
    - https://swiperjs.com/
- Leaflet
    - mappa con markers (tipo google maps)
    - https://leafletjs.com/examples.html
    - registrarsi su mapbox e ottenere token
- Mapbox Geocoding - facoltativo
    - ricerca indirizzi e coordinate gps
    - https://docs.mapbox.com/api/search/geocoding/
- Chart.js
    - grafici opzionali
    - https://www.chartjs.org/docs/latest/getting-started/

- **Tutti i nomi delle pagine html e attributi database possono essere modificati a piacere, l'importante è mantenere le funzionalità richieste**

- **Realizzare il progetto usando un design pattern MVC o REST o un misto delle 2.**