**Relazione Progetto del corso “Introduzione alla Programmazione Web”**

**ID del gruppo: 01**

**1.** **Introduzione**

Il progetto consiste nello sviluppo di una web app per la gestione di un servizio di ticketing online per eventi nella provincia di Trento. L'applicazione permette agli utenti di consultare e acquistare biglietti per eventi culturali e ricreativi, come concerti, spettacoli teatrali, attività sportive e visite guidate.  
Ogni evento mostra informazioni dettagliate come nome, data, orario, luogo, prezzo, immagine e descrizione breve. Gli utenti possono scegliere tra posti a sedere o in piedi e beneficiare di sconti, come un biglietto gratuito ogni cinque acquistati. L'acquisto richiede autenticazione tramite registrazione o login. La piattaforma include anche una sezione amministrativa per la creazione e gestione degli eventi, il monitoraggio delle vendite e la gestione del catalogo.

Il lavoro del progetto è stato organizzato in modo strutturato e collaborativo. Si è iniziato con la costruzione del database, scritto in linguaggio Java, a cui è seguita la realizzazione dell'interfaccia utente. Per rendere l'interfaccia più chiara e intuitiva, sono state utilizzate le classi di Bootstrap.  
Il front end è stato sviluppato principalmente in HTML, arricchito con JavaScript nei casi in cui erano necessarie funzioni più complesse. Per garantire un allineamento costante, sono state organizzate chiamate di gruppo quotidiane, durante le quali si discutevano i progressi e si lavorava insieme. Quando gli obiettivi giornalieri non venivano raggiunti durante gli incontri, i singoli membri continuavano a lavorare autonomamente per completare i propri incarichi. Questo approccio ha facilitato la reperibilità di tutti i componenti, colmando eventuali lacune e mantenendo una visione chiara e condivisa dell'intero progetto.

**2.** **Implementazione MVC**

Nel nostro progetto il pattern MVC è stato implementato in questo modo:

* I Model:

rappresentano i dati e la logica dell’applicazione. Per garantire un’architettura riutilizzabile e chiara, è stata creata una classe astratta chiamata `AbstractModel`, da cui tutte le altre classi model devono ereditare. Ogni model è dedicato a uno scopo specifico: `AdminModel`, `DiscountModel`, `EventModel`, `TicketModel`, e `UserModel`.

I model nel nostro progetto implementano tutte le funzioni necessarie per attuare la logica dell'applicazione, oltre a gestire l’accesso ai dati nel database. Per gestire le connessioni, la classe `AbstractModel` e tutti gli altri model dispongono di due costruttori: uno che crea una nuova connessione al database e uno che sfrutta la connessione ricevuta come parametro, consentendo la gestione condivisa delle connessioni nelle servlet (nel progetto si utilizza una connessione per servlet).

È fornita anche una funzione statica che permette di ottenere una connessione al database. Inoltre, tutte le query che richiedono parametri sono implementate utilizzando `PreparedStatement` anziché `Statement`, assicurando così la sicurezza contro gli attacchi SQL injection.

* Le View:

rappresentano l'interfaccia utente. Per garantire la sicurezza e una gestione ordinata delle pagine, tutte le componenti, eccetto `index.jsp`, sono contenute nella cartella `WEB-INF/view`, in modo che non siano direttamente accessibili dagli utenti.

Sono state create tutte le pagine necessarie e i componenti riutilizzabili da includere in queste pagine, come `topNavAdmin.jsp`, `topNavUser.jsp`, `footer.jsp`, e `returnTopPage.jsp`, che è il pulsante che consente di tornare in cima alla pagina.

Ogni pagina non accede direttamente ai Model per ottenere i dati necessari; questi vengono invece forniti tramite i Controller, assicurando una chiara separazione tra logica di business e presentazione, migliorando così la sicurezza, la manutenibilità dell'applicazione e facilitando la gestione di errori.

* I Controller:

gestiscono l’interazione tra i Model e le View. Nel progetto, i controller sono implementati come Servlet e sono responsabili della generazione delle risposte ai client, ottenendo i dati dai Model (dove le righe delle tabelle sono rappresentate da oggetti entità come `Category`, `Discount`, `Event`, `Location`, `Ticket`, `User`) e passando questi dati alle View.

Le Servlet sono progettate per rispondere a un insieme di path, ad esempio `/admin/\*`, gestiscono le richieste in base al percorso specifico. I path vengono poi differenziati all'interno delle funzioni della Servlet per eseguire operazioni specifiche. Ad esempio, tutte le richieste sotto `/admin/\*` sono gestite dalla `AdminServlet`, che esegue solo operazioni relative agli amministratori.

Alla creazione di una Servlet, viene stabilita una connessione al database e vengono istanziati gli oggetti Model necessari. Al termine della richiesta, la connessione viene chiusa per garantire l'efficienza delle risorse. Inoltre, esiste una classe astratta `AbstractController`, da cui tutti i controller devono estendere, per garantire un’architettura riutilizzabile e chiara.

**3.** **Eventuali funzionalità non implementate**

È stato deciso di non implementare un metodo di pagamento che richiedesse l'inserimento di carte di credito o debito, poiché la web app è concepita come una simulazione di una piattaforma di ticketing e non come un vero e proprio servizio di vendita di biglietti. Si è preferito, infatti, di concentrarsi sull’implementazione delle richieste presenti nella consegna. Ogni funzionalità è stata implementata con successo, garantendo un'esperienza utente completa e coerente con gli obiettivi del progetto. Nonostante l'assenza del sistema di pagamento, l'app risponde pienamente ai requisiti stabiliti.

**4.** **Contributo di ogni componente del gruppo allo sviluppo del progetto**

Sottostante sono riportati i principali contributi al progetto dei singoli membri al progetto.

* Daniele Ye: Si è occupato dell’organizzazione e della creazione del Database della web app, inclusa la gestione delle funzioni per il recupero e la manipolazione dei dati. Ha realizzato le funzioni che implementano la logica dell’applicazione e le classi entità del progetto (Ticket, User, Event, Discount, Location, Category) e si è occupato delle funzioni e delle pagine relative all’amministratore, inclusa la rispettiva servlet. Inoltre, ha implementato i filtri per l’autenticazione degli utenti e dell’amministratore, la pagina principale, le pagine di visualizzazione dei dettagli degli eventi e il pagamento dei biglietti, comprese le rispettive servlet per login, logout e sign up. Infine ha contribuito alla redazione di questo documento.
* Giovanni Panighel: Ha contribuito allo sviluppo della pagina principale della web app, dell’area privata degli utenti, delle pagine di login, logout e sign up, nonché delle servlet relative. Ha lavorato anche ai filtri di autenticazione degli utenti e dell’amministratore, e ha contribuito allo sviluppo della pagina di creazione eventi e a una piccola parte della servlet riguardante le pagine dell’amministratore. Ha redatto questo documento e revisionato il progetto in fase di consegna.
* Elisa Beltrame: Ha lavorato sulla pagina principale, sulle pagine per l’acquisto dei biglietti e su quelle informative del singolo evento, curando l’estetica generale della web app, ideando il logo e il nome. Ha anche contribuito alla redazione di questo documento.

**5.** **Eventuali criticità della vostra soluzione e sviluppi futuri**

Nel progetto da noi sviluppato è presente una criticità riguardante le tabelle presenti nelle pagine “Gestione Eventi” e “Visualizzazione Utenti”: quando si ordinano con l’apposito bottone non si ordinano effettivamente le righe della pagina, ma si manda direttamente una nuova pagina rendendo scomodo il tasto per “tornare indietro” dei browser. Per quanto riguarda le prospettive di sviluppo futuro, crediamo che la webapp da noi programmata possa essere, dopo essere stata migliorata, una ottima base per un sito di e-commerce adibito alla vendita di biglietti semplice e immediatamente comprensibile agli utenti clienti.