

Giuseppe Catillo
(Mat. 165722)

-

28/10/2024

Indice

Introduzione.....	2
Idea.....	2
Funzionalità.....	2
Progettazione.....	3
Permessi.....	3
Database.....	3
Suddivisione in app.....	5
Tecnologie utilizzate.....	6
Backend.....	6
Frontend.....	7
Recommendation System.....	8
Descrizione.....	8
Implementazione.....	8
Preparazione dei dati.....	8
Costruzione della matrice Utente-Evento.....	9
Calcolo della similarità tra utenti.....	9
Generazione delle raccomandazioni.....	9
Statistiche.....	10
Notifiche.....	11
Test.....	12
Risultati.....	13
Conclusioni.....	16

Introduzione

Idea

L'obiettivo del progetto Tikez è quello di creare una **piattaforma di e-commerce dedicata alla vendita e gestione di biglietti per eventi musicali**, come concerti, festival e spettacoli teatrali.

L'applicazione si distingue per un'interfaccia intuitiva e funzionale che facilita l'acquisto dei biglietti per i consumatori ed allo stesso tempo la gestione degli eventi per gli organizzatori.

Funzionalità

Di seguito si elencano le funzionalità dell'applicazione:

- **Visualizzazione eventi:** gli utenti possono visualizzare e filtrare una lista di eventi musicali, inizialmente ordinata per data;
- **Ricerca eventi:** gli utenti possono cercare specifici eventi, organizzatori o luoghi;
- **Acquisto e gestione biglietti:** gli utenti registrati possono acquistare e gestire i biglietti per gli eventi, inclusa la possibilità di cambiare il nominativo entro una finestra temporale;
- **Recommendation system user-based:** gli utenti registrati, che abbiano effettuato almeno un ordine, possono visualizzare una serie di eventi consigliati basati sulla similarità con altri utenti che hanno effettuato acquisti simili;
- **Profilo utente:** il sito prevede un meccanismo di login e registrazione, utilizzato anche per differenziare le operazioni consentite;
- **Salvataggio elementi “preferiti” e ricezione notifiche:** gli utenti registrati possono seguire eventi, organizzatori o luoghi e ricevere aggiornamenti a riguardo;
- **Gestione eventi:** gli organizzatori possono creare e gestire eventi, inclusi dettagli come biglietti, location e locandina;
- **Statistiche eventi:** gli organizzatori possono visualizzare statistiche dettagliate sugli eventi da loro gestiti;
- **Gestione globale:** l'amministratore del sistema ha pieno controllo sulle operazioni della piattaforma tramite una sezione dedicata;
- **Messaggi di log:** l'applicazione mostra messaggi o pagine informative sulle operazioni svolte, favorendo la sicurezza e l'usabilità.

Progettazione

Permessi

Il sito è navigabile da differenti tipologie di utenti, il che richiede un meccanismo di **differenziazione dei permessi** accennato precedentemente:

- **Utente anonimo:**
 - può visualizzare la lista di eventi, inizialmente ordinata per data, e filtrarla sulla base di categoria, località e periodo;
 - può ricercare eventi specifici, organizzatori e luoghi evento;
 - nel momento in cui desidera effettuare un'operazione che non sia tra quelle elencate precedentemente deve registrarsi alla piattaforma.
- **Utente registrato (cliente):**
 - può gestire un proprio profilo personale (dati personali, storico acquisti, preferiti);
 - può visionare una sezione di eventi consigliati;
 - può ricevere aggiornamenti su organizzatori, luoghi evento o eventi in particolare che segue (ticket alert);
 - può acquistare biglietti e gestirli (cambio nominativo).
- **Organizzatore:**
 - può creare eventi e gestirne i dettagli (biglietti, location, locandina ecc...);
 - può visionare statistiche sugli eventi e sui partecipanti attesi.
- **Admin:**
 - registra gli organizzatori;
 - ha pieno controllo su tutto.

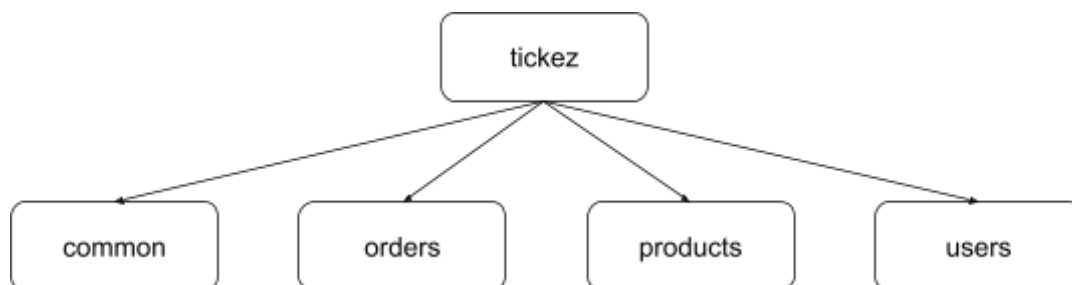
Database

Il database del progetto è composto dalle seguenti entità:

- **Luogo:** località fisica dove possono avvenire gli eventi, registrata dall'admin;
- **Notifica:** notifica del sistema;
- **Ordine:** acquisto effettuato da un utente registrato;
- **BigliettoAcquistato:** istanza di una tipologia di biglietto, compone l'ordine;
- **Evento:** evento musicale;
- **Biglietto:** tipologia di biglietto associata ad un evento;
- **Utente:** cliente registrato della piattaforma;
- **Organizzatore:** organizzatore dell'evento (artista).

Suddivisione in app

Il progetto, sviluppato tramite framework Django, è stato suddiviso logicamente nelle seguenti “app”, con l'intento di garantire una chiara organizzazione del progetto stesso, a favore di una migliore manutenibilità e scalabilità del sistema:



- **Common:** app dedicata alle funzionalità ed elementi condivisi tra tutti gli utenti.
 - **Modelli:** Luogo, Notifica;
 - **Template:** Homepage, lista e dettagli dei luoghi, funzionalità di ricerca.
- **Orders:** app dedicata alla gestione degli ordini dei clienti, in particolare l'acquisto e la modifica dei biglietti acquistati.
 - **Modelli:** Ordine, BigliettoAcquistato
 - **Template:** Processo d'acquisto, modifica di un biglietto acquistato
- **Products:** app dedicata alla creazione, gestione e visualizzazione degli eventi e le tipologie di biglietti annesse.
 - **Modelli:** Evento, Biglietto
 - **Template:** Creazione, modifica ed eliminazione di evento/biglietto, lista e dettagli degli eventi, statistiche evento
- **Users:** app dedicata alle operazioni di registrazione, autenticazione e gestione dei profili degli utenti e organizzatori, compresa la funzionalità del salvataggio di elementi “preferiti”
 - **Modelli:** Utente, Organizzatore (entrambi estendono User);
 - **Template:** Moduli di login e registrazione, profilo utente, lista e dettagli degli organizzatori (artisti).

Nella top-level directory ‘tickez’ si trovano i template ‘base.html’, ‘footer.html’ e ‘navbar.html’ utilizzati per definire la struttura di tutte le pagine che verranno renderizzate, oltre al template ‘404.html’ utilizzato per la gestione degli errori.

Tecnologie utilizzate

Backend

Lo sviluppo dell'applicazione si basa sul design pattern Model-Template-View, tipico del **framework Django**, caratterizzato quindi da una chiara separazione delle responsabilità:

- **Model:** struttura logica dei dati;
- **Template:** layer di presentazione dei dati;
- **View:** logica dell'applicazione e formattazione dei dati.

Tra le tecnologie utilizzate possiamo evidenziare:

- **SQLite:** Database Management System predefinito di Django, ideale per progetti di dimensioni ridotte o durante le fasi iniziali di sviluppo data la sua leggerezza e facilità di configurazione;
- **django-filter:** implementazione di un filtraggio avanzato per gli eventi;

```
# form di filtraggio degli eventi
class EventoFilter(django_filters.FilterSet): ...
```

products/filters.py

- **crispy-forms:** miglioramento dell'aspetto e gestione dei form;
- **scikit-learn:** libreria che fornisce strumenti per l'analisi predittiva dei dati, in questo caso utilizzata per definire la cosine similarity (vedi [Recommendation System](#));
- **pandas:** libreria utilizzata per la manipolazione e analisi dei dati tramite strutture dati flessibili come DataFrame (vedi [Recommendation System](#));
- **django-braces:** libreria che semplifica l'implementazione di controlli di accesso basati sui ruoli ed autorizzazioni degli utenti;

```
# funzionalità di modifica nominativo, accessibile dai soli Clienti
class UpdatePurchaseView(GroupRequiredMixin, ...
    group_required = ['Clienti']
```

orders/views.py

- **pillow:** libreria utilizzata per la manipolazione e gestione delle immagini all'interno dei modelli e tabelle del database;
- **django-messages:** “flash messages” utilizzati per comunicare in modo istantaneo informazioni, al seguito di operazioni svolte dall'utente.

⚠ Si è verificato un errore. Riprova



Login

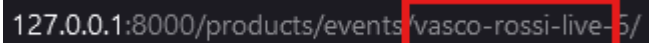
Sei un nuovo utente? **Registrati ora**

Messaggio di errore in caso di credenziali errate

Frontend

Di seguito si elencano le principali tecnologie e tecniche adottate per creare un'interfaccia utente responsiva e di facile utilizzo:

- **HTML:** linguaggio standard per la creazione di pagine web;
- **CSS:** utilizzato per la presentazione e lo stile delle pagine web, permette di applicare stili visivi agli elementi HTML;
- **Font Awesome:** libreria di icone vettoriali, utilizzate per migliorare l'estetica del sito;
- **Bootstrap:** framework CSS che fornisce una vasta gamma di componenti grafici predefiniti e stili personalizzabili;
 - **Datepicker:** plugin che consente agli utenti di selezionare facilmente una data tramite un'interfaccia grafica interattiva;
- **Slug:** tecnica per creare URL leggibili e ottimizzati per i motori di ricerca, trasforma i titoli delle pagine in stringhe di testo concise e descrittive;



```
127.0.0.1:8000/products/events/vasco-rossi-live-5/
```

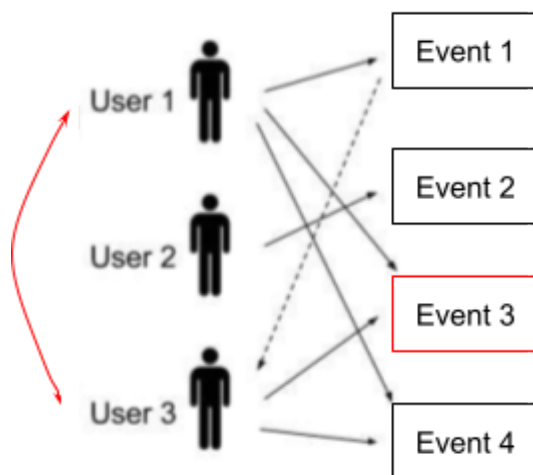
Esempio di slug per l'evento 'Vasco Rossi Live'

- **JavaScript:** linguaggio di scripting che aggiunge interattività e dinamismo alle pagine web, migliorando l'esperienza utente in tempo reale;
 - **jQuery:** libreria JavaScript che semplifica la manipolazione del DOM e la gestione degli eventi;
 - **Chart.js:** libreria JavaScript utilizzata per la visualizzazione delle statistiche degli eventi tramite grafici interattivi e personalizzabili (vedi [Statistiche](#)).

Recommendation System

Descrizione

Il sistema di raccomandazione implementato utilizza un approccio basato sulla **collaborazione tra utenti** (user-based collaborative filtering):



Il collegamento degli utenti avviene tramite una **matrice utente-evento** ed un'altra matrice che invece esprime la **similitudine tra utenti**.

Basandosi, poi, sui pattern di acquisto degli utenti, si genera una lista di eventi raccomandati per l'utente attuale, escludendo quelli già acquistati.

Implementazione

In questo caso specifico il sistema è stato implementato nel seguente modo.

Preparazione dei dati

Il sistema parte dalla raccolta dei **dati degli acquisti dei biglietti**, che includono informazioni su quali utenti hanno acquistato quali biglietti per quali eventi.

```
biglietti_acquistati = BigliettoAcquistato.objects.all()
ordini_utente = Ordine.objects.filter(utente=request.user.utente)

# gli utenti devono aver effettuato almeno un acquisto
if ordini_utente.count() > 0:
    # dataframe con le informazioni rilevanti
    data = {
        'utente_id': [biglietto.ordine.utente.id for biglietto in biglietti_acquistati],
        'evento_id': [biglietto.biglietto.evento.id for biglietto in biglietti_acquistati]
    }
    df = pd.DataFrame(data)
```

tickez/views.py

Costruzione della matrice Utente-Evento

Una volta ottenuti i dati degli acquisti, si crea una matrice che rappresenta l'**interazione tra gli utenti** (righe) e **gli eventi** (colonne):

```
# matrice utente-elemento
# l'argomento "aggfunc='size'" conta il numero di occorrenze di ciascuna combinazione di utente ed evento.
user_event_matrix = df.pivot_table(index='utente_id', columns='evento_id', aggfunc='size', fill_value=0)
```

tickez/views.py

Ogni cella della matrice rappresenta, quindi, il conteggio tra un utente specifico e un evento specifico. Per esempio, se un utente ha acquistato più volte lo stesso evento, il valore nella cella sarà maggiore.

Calcolo della similarità tra utenti

Utilizzando la matrice definita precedentemente, si calcola la similarità tra gli utenti.

In particolare, si utilizza la **similarità del coseno** per misurare quanto due utenti siano simili in base ai loro pattern di acquisto di eventi, il che restituisce a sua volta una matrice di similarità.

```
# calcolo della somiglianza attraverso similarità del coseno tra gli utenti
user_similarity = cosine_similarity(user_event_matrix)
user_similarity_df = pd.DataFrame(user_similarity, index=user_event_matrix.index, columns=user_event_matrix.index)
```

tickez/views.py

Generazione delle raccomandazioni

Quando un utente registrato visita l'homepage, il sistema trova gli utenti più simili, sulla base degli acquisti effettuati, e raccomanda eventi che non ha ancora acquistato.

```
# restituisce i 5 elementi più affini
def get_user_recommendations(utente_id, user_event_matrix, user_similarity_df, top_n=5):
    # ordina per somiglianza decrescente gli utenti della colonna 'utente_id', escluso se stesso, ottenendo così i più simili
    similar_users = user_similarity_df[utente_id].sort_values(ascending=False).index[1:]

    # recupera le righe della matrice utente-elemento corrispondenti agli utenti nella lista 'similar_users', e quindi i loro eventi acquistati
    similar_users_events = user_event_matrix.loc[similar_users]

    # somma le interazioni degli utenti simili per ottenere un punteggio di raccomandazione
    # axis = 0 specifica che l'operazione avviene lungo le righe, producendo una somma per ciascuna colonna
    recommendations = similar_users_events.sum(axis=0).sort_values(ascending=False)

    # rimuove dagli elementi consigliati gli eventi che l'utente ha già acquistato (quindi in cui la cella è !=0)
    user_events = user_event_matrix.loc[utente_id]
    recommendations = recommendations[recommendations[user_events] == 0]

    return recommendations.head(top_n).index

# recupera gli eventi consigliati per l'utente
utente_id = request.user.utente_id
recommended_event_ids = get_user_recommendations(utente_id, user_event_matrix, user_similarity_df)
recommended_events = Evento.objects.filter(id__in=recommended_event_ids)

ctx['recommended_events'] = recommended_events
```

tickez/views.py

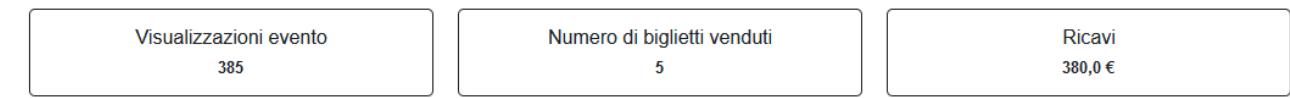
Statistiche

Il sistema di statistiche per gli eventi offre agli organizzatori una panoramica dettagliata dell'**andamento di ciascun evento** tramite una dashboard dedicata, nella quale vengono mostrati dati cruciali e analisi suddivisi in diverse categorie che consentono di ottimizzare le strategie di marketing e migliorare la pianificazione per eventi futuri:

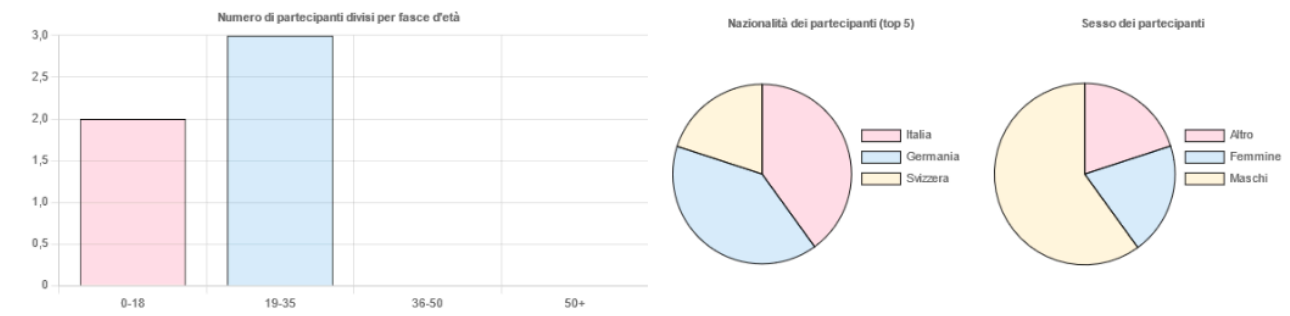
- **Numero di visualizzazioni dell'evento;**
- **Ricavo generato dalla vendita dei biglietti;**
- **Demografia dei partecipanti:** distribuzione per fasce di età dei partecipanti;
- **Nazionalità dei partecipanti:** panoramica delle cinque nazionalità più comuni tra i partecipanti all'evento;
- **Distribuzione di genere dei partecipanti;**
- **Vendite per tipologia di biglietto;**
- **Vendite giornaliere.**

Statistiche

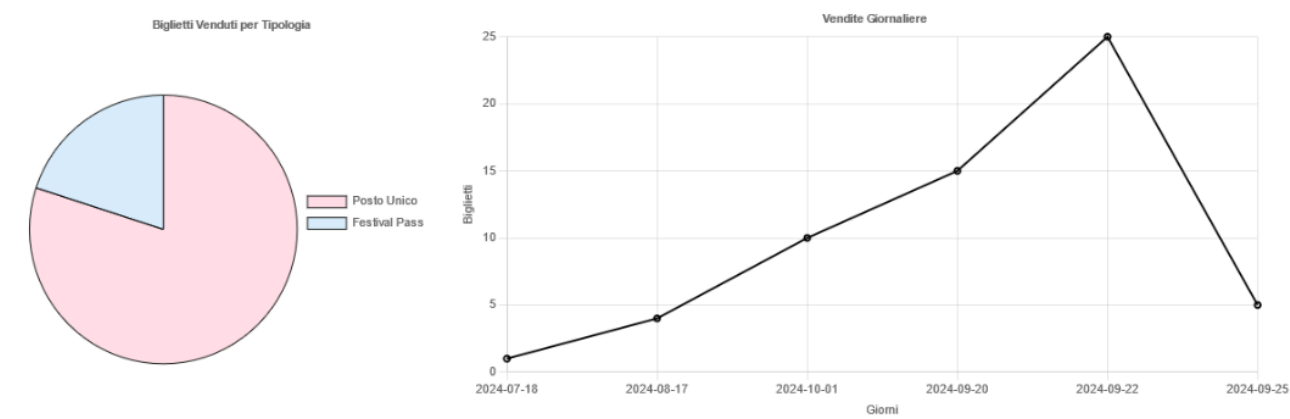
Festa della Musica 2024



Dettagli partecipanti:



Dettagli biglietti:



Statistiche per l'evento "Festa della Musica 2024 - Ligabue"

Notifiche

Le notifiche sono state progettate per informare un utente sui **nuovi eventi** creati dagli organizzatori che segue. Questa funzionalità è integrata direttamente nella barra di navigazione, accessibile da un menù a tendina.



Notifica per l'evento "Nome e Cognome Tour" creato da Ligabue

La disponibilità delle notifiche, assieme al loro stato, è garantita in tutti i template dell'applicazione tramite l'utilizzo di un **context processor**, che permette di evitare l'aggiunta di una logica di controllo per le notifiche in ogni singola view:

```
from .models import Notifica

# context processor utilizzato per fornire le notifiche dell'utente e il loro stato a tutti i template che richiedono questi dati.

# utilizzando @login_required, con un utente non autenticato otterrei un HttpResponseRedirect e non un dizionario vuoto
def notifications(request):
    notifications = []
    unread_count = 0

    if request.user.is_authenticated:
        # la funzionalità delle notifiche viene gestita solo per il model Utente
        try:
            utente = request.user.utente
            notifications = Notifica.objects.filter(
                organizzatore__in=utente.organizzatori_preferiti.all()
            ).order_by('-data_ora')
            unread_count = notifications.filter(letta=False).count()
        except:
            utente = None

    return {
        'notifications': notifications,
        'unread_count': unread_count,
    }
```

common/context_processor.py

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR, 'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
                'common.context_processors.notifications',
            ],
        },
    },
]
```

tickez/settings.py

Test

Il progetto implementa classi dedicate allo **unit testing** di alcune funzionalità del programma, che garantiscono qualità e manutenibilità del codice.

Nello specifico, sono state considerate:

- app *users* → **toggle_follow** (*users*): funzionalità di aggiunta o rimozione di elementi “preferiti” da parte dei soli utenti registrati (clienti);
- app *products* → **UpdateEvent**: “vista” (pagina) utente utilizzata dai soli organizzatori (e admin) per modificare i dettagli di un evento.

Ogni file *tests.py* presenta una fase di inizializzazione degli oggetti richiesti per i test, seguita poi dagli effettivi test che analizzano in ottica black-box la consistenza architetturale delle funzionalità, sulla base di codici di risposta HTTP, renderizzazione dei template e modifiche al database.

Ad esempio:

```
def setUp(self):
    # creazione di uno User per l'Utente di test
    self.user_utente = User.objects.create_user(username='testuser_utente', password='password123_utente')
    group_clienti, created = Group.objects.get_or_create(name='Clienti')
    self.user_utente.groups.add(group_clienti)
    # creazione di un Utente associato allo User appena creato
    self.utente = Utente.objects.create(
        user=self.user_utente,
        nome='Nome Utente',
        cognome='Cognome Utente',
        email='utente@example.com',
        data_nascita='2000-01-01'
    )
    ...
# consistenza architetturale
def test_toggle_follow_architectural_consistency(self):
    self.client.force_login(self.user_utente)

    entities = [
        ('evento', self.evento.pk),
        ('organizzatore', self.organizzatore.pk),
        ('luogo', self.luogo.pk),
    ]

    # azione 'follow' sulle varie entità
    for entity_type, entity_pk in entities:
        url = reverse('users:toggle-follow', args=[entity_type, entity_pk])
        response = self.client.post(url, {'action': 'follow'})

        entity = get_object_or_404({
            'evento': Evento,
            'organizzatore': Organizzatore,
            'luogo': Luogo,
        }[entity_type], pk=entity_pk)

        self.assertRedirects(response, entity.get_absolute_url())
        self.assertIn(self.user_utente.utente, entity.followers.all())

    # azione 'unfollow' sulle varie entità
    for entity_type, entity_pk in entities:
        url = reverse('users:toggle-follow', args=[entity_type, entity_pk])
        response = self.client.post(url, {'action': 'unfollow'})


        entity = get_object_or_404({
            'evento': Evento,
            'organizzatore': Organizzatore,
            'luogo': Luogo,
        }[entity_type], pk=entity_pk)

        self.assertRedirects(response, entity.get_absolute_url())
        self.assertNotIn(self.user_utente.utente, entity.followers.all())
```

users/tests.py

Risultati

L'homepage del sito si presenta in questo modo:



Eventi ▾

Evento, Artista o Luogo

Accedi


Categorie

Concerti

Festival

Teatro

Prossimi eventi




Eros Ramazzotti in Concerto

Eros Ramazzotti

Teatro alla Scala

07/11/2024

Dettagli



Elisa in Concerto

Elisa

Teatro degli Arcimboldi

15/01/2025

Dettagli

Billie Eilish in Concert

Billie Eilish

Mediolanum Forum

25/01/2025


Dettagli

Vedi tutti

Scopri

Artisti

Luoghi



Tickez

Chi siamo

Stampa

Lavora con noi

Assistenza Clienti

Termini e condizioni d'acquisto

Contattaci


FAQ

© 2024 Tickez. All Rights Reserved.

Informativa sulla privacy | Cookies | Gestione dei cookies

In caso di utente registrato, viene visualizzata anche una serie di eventi consigliati:

Consigliati per te




Concerto di Taylor Swift

Taylor Swift

Mediolanum Forum

28/02/2025

Dettagli



Ed Sheeran Live

Ed Sheeran

Mediolanum Forum

10/07/2026

Dettagli

Ligabue - Tour 2024

Ligabue

Teatro alla Scala

28/01/2026

Dettagli

Ariana Grande World Tour


Ariana Grande

Teatro alla Scala

21/07/2024

Dettagli

Prossimi eventi




Eros Ramazzotti in Concerto

Eros Ramazzotti

Teatro alla Scala

07/11/2024

Dettagli



Elisa in Concerto

Elisa

Teatro degli Arcimboldi

15/01/2025

Dettagli

Billie Eilish in Concert

Billie Eilish

Mediolanum Forum

25/01/2025

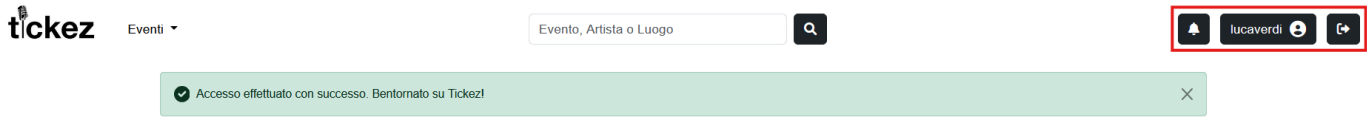
Dettagli

Vedi tutti

13

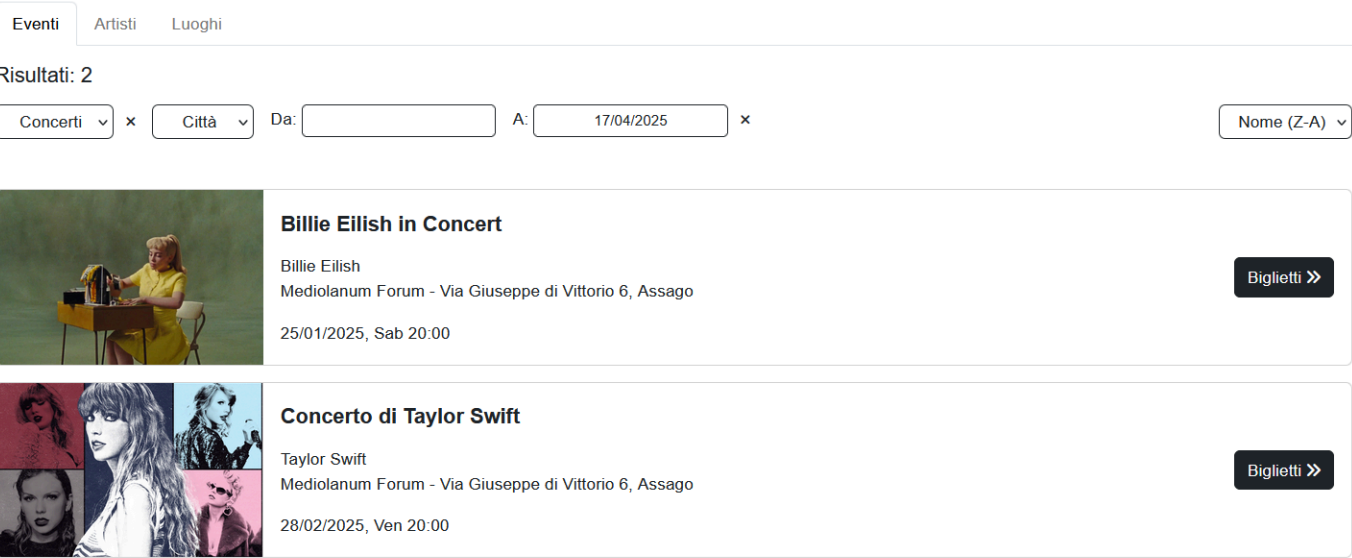
Ogni pagina visualizzata presenta la navbar ed il footer, oltre al contenuto specifico mostrato al centro.

Di seguito ulteriori sezioni interessanti dell'applicazione:



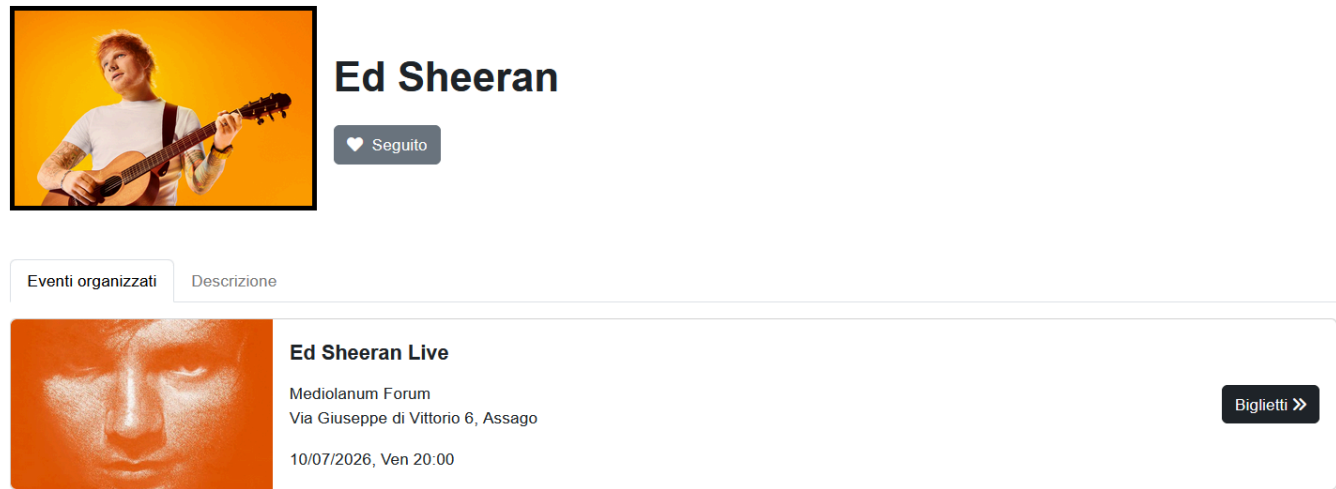
Visualizzazione navbar in caso di utente registrato

Ricerca: "concert" ✕



1

Funzionalità di ricerca e visualizzazione filtrata dei risultati



1

Dettagli organizzatore, visti da un utente registrato



Elisa in Concerto

Elisa

Teatro degli Arcimboldi - Viale dell'Innovazione 20, Milano

Mercoledì 15 Gennaio 2025 19:00



Biglietti

Descrizione

Per questo evento sarà consentito eseguire la procedura di cambio nominativo fino al 31/12/2024

Poltronissima

Posti in poltronissima, offrendo la migliore visibilità e il massimo comfort durante lo spettacolo.

100,0 €



Platea Numerata

Posti numerati nella platea del teatro, garantendo una visione ottimale e un comfort elevato.

80,0 €



Aggiungi biglietto

Dettagli evento, visti dall'organizzatore

Checkout

Riepilogo Ordine

- Poltronissima - 100,0 €

Totale: 100,0 €

Dati di Pagamento

Titolare carta*

Nome Cognome

Numero carta*

XXXX XXXX XXXX XXXX

Data di scadenza*

MM/YY

CVV*

XXX

Paga

Ripristina

Checkout utente registrato



Bentornato, **lucaverdi**

Dati personali

Ordini

Eventi preferiti

Artisti preferiti

Luoghi preferiti

Ordine N. 10 - 18/07/2024, Gio 04:48

Totale: 70,0 €

Festa della Musica 2024

Posto Unico - 70,0€

Nominativo: Luca Verdi



Sezione ordini del profilo utente

Conclusioni

Allo stato attuale, il progetto rappresenta una solida base per ulteriori estensioni delle funzionalità che potrebbero migliorare l'esperienza utente come, ad esempio:

- introduzione di **recensioni** da parte degli utenti;
- creazione di un **carrello** con l'integrazione di servizi di pagamento reali;
- implementazione di politiche di **reso/rimborso**.

Anche i modelli di Notifica e Luogo sono stati progettati con campi aggiuntivi per consentire futuri sviluppi dell'applicazione, senza richiedere significative modifiche alla struttura esistente.

Un problema riscontrato durante lo sviluppo, infatti, è risultato essere la progettazione del database, tra riferimenti circolari e modifiche durante lo sviluppo.

In sintesi, il progetto è ben strutturato e pronto per accogliere ulteriori funzionalità, mantenendo una solida base per garantire la qualità e la continuità nel tempo.