

Traccia d'Esame – Progetto di sistema informativo per la gestione di un Bed & Breakfast

Il progetto ha come obiettivo la realizzazione di un sistema informativo moderno e completo, destinato a digitalizzare la gestione operativa di un Bed & Breakfast, superando le attuali procedure manuali. Il sistema si concretizzerà in un'applicazione web intuitiva, progettata per migliorare l'efficienza gestionale e offrire un'esperienza utente avanzata.

Il cuore del sistema sarà costituito da un database relazionale SQL, progettato per organizzare in modo strutturato tutte le informazioni necessarie alla gestione della struttura. Al suo interno saranno presenti, tra le altre, una sezione dedicata al catalogo camere, con dati relativi a tipologia, numero, prezzo per notte, descrizione e stato di manutenzione, e un modulo per la gestione delle prenotazioni, che permetterà di verificare disponibilità, tracciare stato e durata del soggiorno, nonché includere eventuali servizi aggiuntivi.

Il sistema prevedrà tre principali tipologie di utenti:

- **Clienti**, per cui verranno archiviati dati anagrafici, recapiti e preferenze;
- **Staff**, coinvolto nella gestione operativa e assegnato ad attività o eventi;
- **Gestori**, con responsabilità amministrative e supervisione del personale.

L'accesso all'applicazione avverrà tramite credenziali personali, con interfacce differenziate a seconda del ruolo. I clienti avranno la possibilità di effettuare prenotazioni, aggiungere servizi extra e lasciare recensioni. Lo staff potrà consultare i propri incarichi, mentre i gestori avranno pieno controllo sul sistema e sul personale.

La gestione delle prenotazioni rappresenta una funzionalità centrale, permettendo di tracciare l'intero ciclo di soggiorno: dalla richiesta iniziale all'aggiunta di camere e servizi, fino alla conclusione. Ogni prenotazione conserverà lo storico dei servizi richiesti e delle camere assegnate, insieme al relativo prezzo complessivo e numero di ospiti.

Gli utenti avranno la possibilità di lasciare valutazioni e recensioni sui servizi offerti, tra cui camere, eventi locali ed esperienze proposte, contribuendo a garantire trasparenza e miglioramento continuo della qualità.

Il Bed & Breakfast organizza inoltre eventi locali ed esperienziali aperti agli ospiti, come degustazioni, escursioni guidate e attività culturali. Il sistema permetterà di pubblicare informazioni dettagliate sugli eventi, gestire le iscrizioni online, registrare le presenze e conservare uno storico degli eventi passati, con materiali associati.

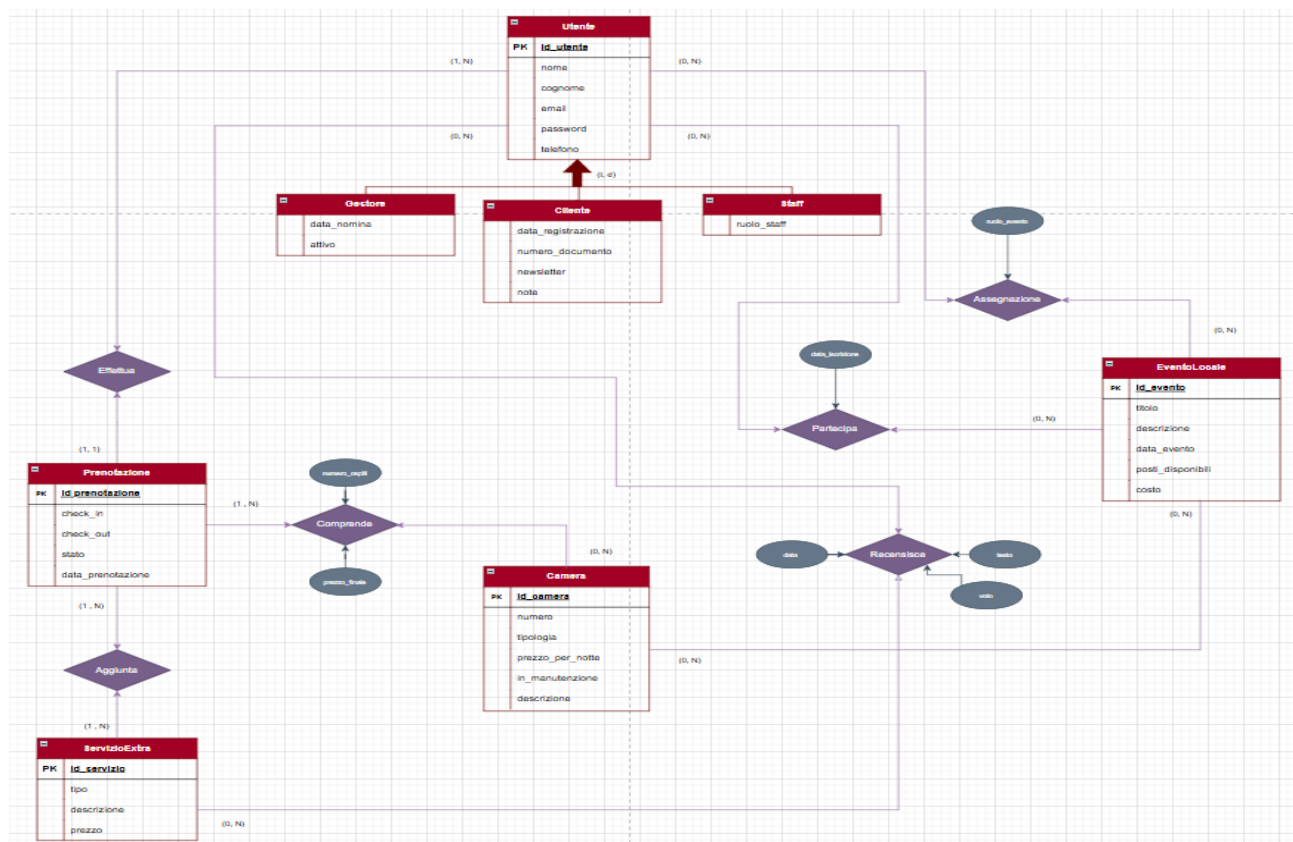
Modello E/R

Il sito è incentrato principalmente sulla gestione delle interazioni con i clienti del Bed & Breakfast. Questi ultimi potranno consultare un catalogo completo delle camere, contenente descrizioni dettagliate, immagini illustrative, prezzi per notte, tipologie disponibili e informazioni aggiornate sulla disponibilità o eventuale stato di manutenzione.

Una volta selezionata la camera desiderata, l'utente potrà effettuare una prenotazione, specificando le date di soggiorno, il numero di ospiti e, se desiderato, aggiungendo uno o più servizi extra messi a disposizione dalla struttura. Tutte le prenotazioni effettuate saranno consultabili tramite un'apposita area riservata personale, accessibile mediante credenziali inserite durante la fase di registrazione.

Il sistema prevede anche la gestione di eventi locali ed esperienziali organizzati dal B&B (ad esempio, escursioni, degustazioni, attività culturali), ai quali gli ospiti potranno iscriversi e partecipare. Gli eventi includeranno informazioni come luogo, data, descrizione e numero massimo di partecipanti.

I gestori della struttura avranno a disposizione un pannello di controllo dedicato, che consentirà loro di aggiornare lo stato delle prenotazioni, monitorare i servizi erogati, pubblicare eventi, consultare le recensioni inserite dagli utenti e accedere a tutte le informazioni relative ai clienti registrati e allo staff operativo.



Inizialmente è stato realizzato un modello E/R con un'entità padre e tre entità figlie, collegate da una relazione di specializzazione totale e disgiunta.

Relazione Utente → Cliente, Staff, Gestore

- Vincolo di copertura: Totale → Ogni utente registrato nel sistema ricopre sempre uno e un solo ruolo (Cliente, Staff o Gestore).
- Vincolo di disgiunzione: Disgiunta → Un utente può appartenere a una sola categoria per volta: o Cliente, o Staff, o Gestore.

La specializzazione dell'entità *Utente* è stata così definita:

- Attributi comuni (nella tabella padre Utente): id_utente, nome, cognome, email, password, telefono
- Cliente: data_registrazione, numero_documento, newsletter, note
- Staff: ruolo_staff
- Gestore: data_nomina, attivo

1. **Soluzione 1: Accorpamento in padre**

In questo approccio, la generalizzazione ingloba anche le specializzazioni, creando un'unica tabella Utente che contiene tutti gli attributi comuni e specifici delle entità figlie (Cliente, Staff e Gestore).

Gli attributi specifici vengono resi opzionali (accettano valori NULL se non pertinenti) e si aggiunge un attributo ruolo per distinguere il tipo di utente.

Utente(id_utente, nome, cognome, email, password, telefono, ruolo [cliente | staff | gestore], data_registrazione, numero_documento, newsletter, note, ruolo_staff, data_nomina, attivo)

Vantaggi:

- Una sola tabella da gestire.
- Maggiore semplicità nelle query generiche sugli utenti.

Svantaggi:

- Presenza di numerosi valori NULL per gli attributi non rilevanti in base al ruolo.
- Minore integrità semantica rispetto alle entità figlie.

2. **Soluzione 2: Accorpamento in figlie**

In questo approccio, le entità specializzate (Cliente, Staff, Gestore) ereditano gli attributi dall'entità padre Utente.

Si creano quindi tre tabelle distinte, ognuna con gli attributi comuni e quelli specifici del proprio ruolo.

Vantaggi:

- Nessun attributo con valore NULL.

- Maggiore chiarezza semantica: ogni tabella rappresenta in modo preciso un tipo di utente.

Svantaggi:

- Ridondanza nelle relazioni: tutte le entità figlie (Cliente, Staff, Gestore) devono essere collegate separatamente ad altre entità tramite relazioni come:
 - Effettua → verso la tabella Prenotazione
 - Partecipa → verso la tabella EventoLocale
 - Recensisce → verso le tabelle Camere, EventoLocale e ServiziExtra
- Aumento della complessità nella gestione delle chiavi esterne e delle query.

Modello E/R ottimizzato

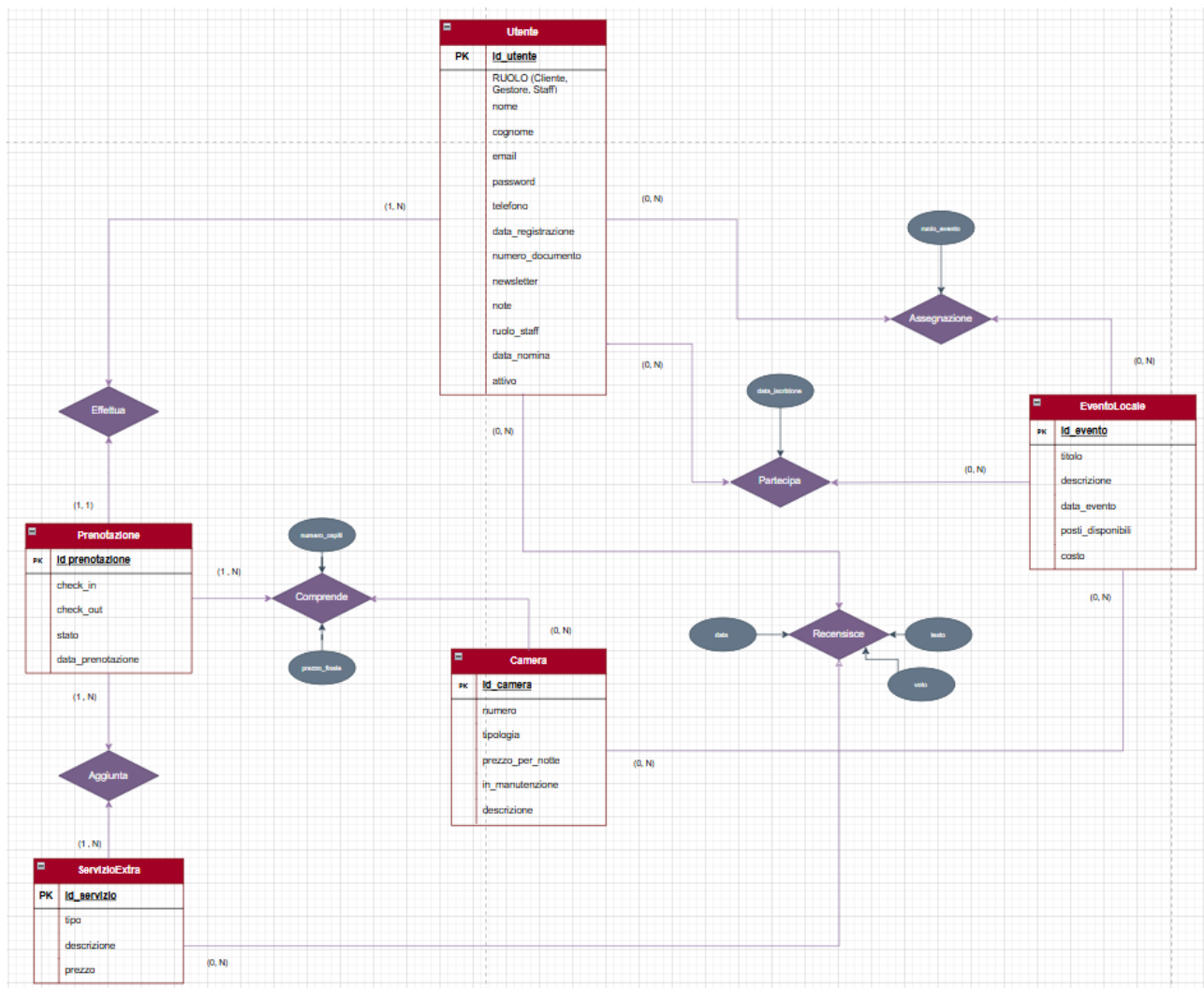
Si è scelto di risolvere la generalizzazione assorbendo le specializzazioni all'interno dell'entità padre Utente, inglobando al suo interno anche gli attributi specifici delle entità figlie Cliente, Staff e Gestore. Questa decisione è motivata dal fatto che le funzionalità previste dal sistema non richiedono una gestione operativa distinta per ogni tipologia di utente in fase di interrogazione del database.

L'approccio adottato prevede dunque la creazione di un'unica tabella Utente, nella quale confluiscono sia gli attributi comuni che quelli specifici dei ruoli. Gli attributi specifici vengono definiti come opzionali (ovvero possono assumere valore NULL se non pertinenti) e viene introdotto un attributo ruolo per distinguere il tipo di utente all'interno del sistema (cliente, staff, gestore).

Struttura risultante della tabella Utente:

- Utente: id_utente, nome, cognome, email, password, telefono, ruolo (cliente, staff, gestore)
 - *Cliente*: data_registrazione, numero_documento, newsletter, note
 - *Staff*: ruolo_staff
 - *Gestore*: data_nomina, attivo

Nel modello E/R, tutti questi attributi sono visualizzati all'interno dell'entità Utente, ma l'attributo ruolo permette di distinguere in modo chiaro la categoria funzionale di appartenenza.



Progettazione logica

Utente(id_utente, nome, cognome, email, password, telefono, ruolo (Cliente, Staff, Gestore), data_registrazione, numero_documento, newsletter, note, ruolo_staff, data_nomina, attivo)

Prenotazione(id_prenotazione, check_in, check_out, stato (in attesa, confermata, annullata, completata), data_prenotazione)

Camera(id_camera, numero, tipologia (singola, doppia, matrimoniale, suite), prezzo_per_notte, in_manutenzione (sì, no), descrizione)

ServizioExtra(id_servizio, tipo (colazione, navetta, spa, escursione), descrizione, prezzo)

EventoLocale(id_evento, titolo (degustazione, escursione, visita guidata, serata a tema), descrizione, data_evento, posti_disponibili, costo)

Effettua(id_utente:utente, id_prenotazione:prenotazione)

Comprende(id_prenotazione:prenotazione, id_camera:camera, numero_ospiti, prezzo_finale)

Aggiunta(id_prenotazione:prenotazione, id_servizio:servizioextra)

Partecipa(id_utente:utente, id_evento:eventocale, data_iscrizione)

Assegnazione(id_utente:utente, id_evento:eventocale, ruolo_evento)

Recensisce(id_utente:utente, id_camera:camera, id_evento:eventocale, id_servizio:servizioextra, data, testo, voto)

Entità

UTENTE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_utente	Identificativo univoco dell'utente	Integer	–	Primary Key, Auto Increment, Unique, Not null
email	Indirizzo email univoco dell'utente	Varchar	254	Unique, Not null
password	Password per l'accesso al sistema	Varchar	128	Not null
nome	Nome dell'utente (obbligatorio se ruolo = 'Cliente')	Varchar	50	Not null se ruolo = "Cliente"
cognome	Cognome dell'utente (obbligatorio se ruolo = 'Cliente')	Varchar	50	Not null se ruolo = "Cliente"
telefono	Recapito telefonico	Varchar	20	–
ruolo	Ruolo dell'utente (Cliente, Gestore, Staff)	Enum	–	Not null, valori ammessi: ['Cliente', 'Staff', 'Gestore']
data_registrazione	Data di registrazione (solo per clienti)	Date	–	Nullable, valorizzata se ruolo = "Cliente"
numero_documento	Numero identificativo del documento (solo per clienti)	Varchar	30	Nullable, valorizzato se ruolo = "Cliente"
newsletter	Flag per l'iscrizione alla newsletter	Boolean	–	Default = false
note	Note o informazioni aggiuntive	Text	–	Nullable

ruolo_staff	Specifica il ruolo operativo (solo se ruolo = "Staff")	Varchar	50	Nullable, obbligatorio se ruolo = "Staff"
data_nomina	Data di nomina (solo per gestori)	Date	–	Nullable, obbligatoria se ruolo = "Gestore"
attivo	Indica se il gestore è attivo (solo per gestori)	Boolean	–	Nullable, obbligatorio se ruolo = "Gestore"

Ruolo: può assumere come valori Cliente, Staff o Gestore.

- Se viene scelto Cliente, si potranno compilare solo i campi: nome, cognome, data_registrazione, numero_documento, newsletter, note. I campi ruolo_staff, data_nomina e attivo rimarranno di tipo NULL.
- Se viene scelto Staff, si compilerà esclusivamente il campo: ruolo_staff. Tutti gli altri campi specifici (data_registrazione, numero_documento, data_nomina, attivo, ecc.) rimarranno NULL.
- Se viene scelto Gestore, si compileranno solo i campi: data_nomina, attivo. I restanti campi specifici per altri ruoli resteranno NULL.

Vincolo di Tupla

- Email deve essere univoca all'interno del sistema: un utente non può registrarsi più volte con la stessa email.

PRENOTAZIONE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_prenotazione	Identificativo univoco della prenotazione	Integer	–	Primary Key, Auto Increment, Unique, Not null
check_in	Data prevista per l'arrivo	Date	–	Not null
check_out	Data prevista per la partenza	Date	–	Not null
stato	Stato attuale della prenotazione	Enum	–	Not null, valori ammessi: ['in attesa', 'confermata', 'annullata', 'completata']
data_prenotazione	Data in cui è stata effettuata la prenotazione	Date	–	Not null

CAMERA				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_camera	Identificativo univoco della camera	Integer	–	Primary Key, Auto Increment, Unique, Not null
numero	Numero identificativo della camera nella struttura	Varchar	10	Unique, Not null
tipologia	Tipologia della camera	Enum	–	Not null, valori ammessi: ['singola', 'doppia', 'matrimoniale', 'suite']
prezzo_per_notte	Prezzo della camera per notte	Decimal	6,2	Not null
in_manutenzione	Stato di disponibilità per manutenzione	Enum	–	Not null, valori ammessi: ['si', 'no']
descrizione	Descrizione testuale della camera	Text	–	Facoltativo (NULL)

SERVIZIO EXTRA				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_servizio	Identificativo univoco del servizio extra	Integer	–	Primary Key, Auto Increment, Unique, Not null
tipo	Tipologia del servizio offerto	Enum	–	Not null, valori ammessi: ['colazione', 'navetta', 'spa', 'escursione']
descrizione	Descrizione testuale del servizio	Text	–	Facoltativo (NULL)
prezzo	Prezzo del servizio extra	Decimal	6,2	Not null

EVENTO_LOCALE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_evento	Identificativo univoco dell'evento	Integer	—	Primary Key, Auto Increment, Unique, Not null
titolo	Tipologia o nome dell'evento	Enum	—	Not null, valori ammessi: ['degustazione', 'escursione', 'visita guidata', 'serata a tema']
descrizione	Descrizione testuale dell'evento	Text	—	Facoltativo (NULL)
data_evento	Data in cui si svolge l'evento	Date	—	Not null
posti_disponibili	Numero massimo di partecipanti ammessi	Integer	—	Not null, ≥ 0
costo	Costo di partecipazione all'evento	Decimal	6,2	Not null

Relazioni

EFFETTUA				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_utente	Utente che effettua la prenotazione	Foreign Key	—	Primary Key (composita), Not null, riferimento a Utente(id_utente), On delete cascade
id_prenotazione	Prenotazione associata all'utente	Foreign Key	—	Primary Key (composita), Not null, riferimento a Prenotazione(id_prenotazione), On delete cascade

Relazione 1:N tra Utente e Prenotazione

- Un utente può effettuare da 0 a N prenotazioni.
- Ogni prenotazione è effettuata da uno e un solo utente.
- La prenotazione non può esistere senza un utente associato.
- La relazione serve a gestire la proprietà della prenotazione e l'identità dell'utente che l'ha effettuata.

COMPRENDE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_prenotazione	Prenotazione a cui appartiene la camera	Foreign Key	—	Primary Key (composita), Not null, riferimento a Prenotazione(id_prenotazione), On delete cascade
id_camera	Camera assegnata alla prenotazione	Foreign Key	—	Primary Key (composita), Not null, riferimento a Camera(id_camera), On delete cascade
numero_ospiti	Numero di ospiti nella camera	Integer	—	Not null, ≥ 1
prezzo_finale	Prezzo totale della camera per il soggiorno	Decimal	—	Not null, ≥ 0

Relazione 1:N tra Prenotazione e Camera tramite Comprende

- Una prenotazione deve includere almeno una camera (1:N).
- Una camera può non essere associata a nessuna prenotazione (0:N).
- La relazione Comprende consente di associare più camere a una prenotazione, specificando numero di ospiti e costo per ciascuna.
- La relazione non può esistere senza una prenotazione e una camera associate.

AGGIUNTA				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_prenotazione	Prenotazione a cui è associato il servizio extra	Foreign Key	—	Primary Key (composita), Not null, riferimento a Prenotazione(id_prenotazione), On delete cascade
id_servizio	Servizio extra aggiunto alla prenotazione	Foreign Key	—	Primary Key (composita), Not null, riferimento a ServizioExtra(id_servizio), On delete cascade

Relazione N:M tra Prenotazione e ServizioExtra tramite Aggiunta

- Una prenotazione può includere uno o più servizi extra.
- Un servizio extra può essere collegato a una o più prenotazioni.
- La relazione Aggiunta rappresenta il legame tra una prenotazione e i servizi extra richiesti dall'utente.
- La relazione non può esistere senza una prenotazione e un servizio associato.

PARTECIPA				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_utente	Utente che partecipa all'evento	Foreign Key	—	Primary Key (composita), Not null, riferimento a Utente(id_utente), On delete cascade
id_evento	Evento a cui partecipa l'utente	Foreign Key	—	Primary Key (composita), Not null, riferimento a EventoLocale(id_evento), On delete cascade
data_iscrizione	Data in cui l'utente si è iscritto all'evento	Date	—	Not null

Relazione N:M tra Utente e EventoLocale tramite Partecipa

- Un utente può partecipare a nessun evento o a più eventi.

- Un evento può avere nessun partecipante o più partecipanti.
- La relazione Partecipa tiene traccia dell'iscrizione dell'utente a uno specifico evento, registrando anche la data di iscrizione.
- La relazione non può esistere senza un utente e un evento associati.

ASSEGNAZIONE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_utente	Staff assegnato all'evento	Foreign Key	—	Primary Key (composita), Not null, riferimento a Utente(id_utente), On delete cascade
id_evento	Evento al quale è assegnato lo staff	Foreign Key	—	Primary Key (composita), Not null, riferimento a EventoLocale(id_evento), On delete cascade
ruolo_evento	Ruolo svolto dallo staff nell'evento	Varchar	50	Not null

Relazione N:M tra Utente e EventoLocale tramite Assegnazione

- Un utente può essere assegnato a 0 o più eventi.
- Un evento può avere 0 o più utenti assegnati.
- La relazione Assegnazione gestisce il ruolo ricoperto dall'utente all'interno di ciascun evento.
- La relazione non può esistere senza un utente e un evento associati.

RECENSISCE				
Nome Campo	Descrizione	Tipo	Lunghezza	Vincoli
id_utente	Utente che scrive la recensione	Foreign Key	—	Primary Key (composita), Not null, riferimento a Utente(id_utente), On delete cascade
id_camera	Camera recensita (opzionale)	Foreign Key	—	Primary Key (composita), Nullable, riferimento a Camera(id_camera), On delete cascade

id_servizio	Servizio extra recensito (opzionale)	Foreign Key	—	Primary Key (composita), Nullable, riferimento a ServizioExtra(id_servizio), On delete cascade
id_evento	Evento recensito (opzionale)	Foreign Key	—	Primary Key (composita), Nullable, riferimento a EventoLocale(id_evento), On delete cascade
data	Data della recensione	Date	—	Not null
testo	Testo della recensione	Text	—	Not null
voto	Valutazione numerica (es. da 1 a 5)	Integer	—	Not null, range accettato [1–5]

Relazione N:M tra Utente e Camera / ServizioExtra / EventoLocale tramite Recensisce

- Un utente può scrivere 0 o più recensioni.
- Ogni camera, servizio o evento può ricevere 0 o più recensioni.
- La relazione Recensisce consente di collegare ogni recensione a uno e un solo elemento tra camera, servizio o evento.
- La recensione non può esistere senza un utente associato.
- Almeno uno tra id_camera, id_servizio o id_evento deve essere valorizzato per ogni recensione.

Vincoli interrelazionali

- Ruolo dell'utente e funzionalità consentite
Solo gli utenti con ruolo Cliente possono:
 - Effettuare prenotazioni
 - Aggiungere servizi extra a una prenotazione
 - Iscrivere a eventi locali
Gli utenti Staff e Gestore non possono svolgere queste azioni.
- Recensioni vincolate all'effettivo utilizzo
Le recensioni sono permesse solo se l'utente ha realmente usufruito del servizio:

- Una camera è recensibile solo se è stata inclusa in una prenotazione del cliente (Comprende)
- Un servizio extra può essere recensito solo se aggiunto alla prenotazione del cliente (Aggiunta)
- Un evento locale può essere recensito solo se il cliente ha effettivamente partecipato (Partecipa)
- Ogni elemento può essere recensito una sola volta per ciascuna prenotazione o partecipazione.
- Assegnazione eventi
Solo gli utenti con ruolo Staff possono essere assegnati a uno o più eventi locali tramite la relazione Assegnazione, con specifica del ruolo svolto (es. Chef, Guida, Barman).
- Composizione della prenotazione
Ogni prenotazione deve includere almeno una camera (Comprende).
È possibile aggiungere uno o più servizi extra alla prenotazione (Aggiunta).
- Iscrizione agli eventi locali
Un cliente può partecipare a un evento locale solo se ci sono ancora posti disponibili ($\text{posti_disponibili} > 0$).
Inoltre, non può iscriversi più di una volta allo stesso evento (vincolo di unicità sulla relazione Partecipa).
- Gestione temporale delle prenotazioni
 - Una camera non può essere prenotata contemporaneamente da più utenti per date sovrapposte (vincolo gestito lato applicativo).
 - La data di check-in non può essere successiva alla data di check-out (vincolo logico a livello di applicazione).

Implementazione del sistema informativo

L'applicazione web, sviluppata con Django e MySQL (tramite MariaDB su XAMPP), ha l'obiettivo di digitalizzare la gestione operativa del *Bed & Breakfast Magliulo*. Il sistema semplifica il lavoro del personale e migliora l'esperienza dell'utente, offrendo strumenti completi per la gestione di prenotazioni, eventi locali, servizi extra e recensioni.

Funzionalità attualmente implementate

- Registrazione e autenticazione per ruoli (Cliente, Staff, Gestore)
Il sistema supporta la registrazione e il login differenziato per ruolo, con interfacce personalizzate.
I clienti possono effettuare prenotazioni, iscriversi agli eventi e lasciare recensioni.
Gli utenti staff visualizzano gli incarichi assegnati.

I gestori accedono a una dashboard con strumenti di controllo e creazione contenuti (eventi e camere).

- Gestione delle prenotazioni

I clienti selezionano il periodo di soggiorno e le camere, con la possibilità di aggiungere servizi extra.

Il sistema verifica automaticamente il termine del soggiorno e aggiorna lo stato delle prenotazioni in: *in attesa, in corso, completata, annullata*.

- Gestione degli eventi locali

Gli eventi possono essere creati con titolo, descrizione, data e posti disponibili. I clienti possono iscriversi fino al limite massimo.

Gli utenti staff vengono assegnati ai singoli eventi con un ruolo specifico (es. Chef, Guida, Barman).

- Recensioni vincolate all'esperienza reale

I clienti possono recensire solo camere, eventi o servizi che hanno realmente utilizzato.

Il sistema consente un'unica recensione per ogni elemento associato a una prenotazione o partecipazione.

- Dashboard personale e storico prenotazioni

I clienti possono accedere a uno storico dettagliato delle prenotazioni effettuate, con possibilità di recensire servizi conclusi.

Gli utenti staff visualizzano l'elenco degli eventi a cui sono stati assegnati, con data e ruolo specifico.

Funzionalità previste per sviluppi futuri

- Catalogo camere e servizi extra

Una sezione dedicata alla visualizzazione di tutte le camere disponibili e dei servizi extra, con possibilità di filtri (tipologia, prezzo, descrizione).

- Notifiche e aggiornamenti automatici

Il sistema potrà inviare email o avvisi in dashboard per eventi imminenti, conferme, cancellazioni o variazioni, e aggiornare lo stato delle camere in tempo reale.

- Pianificazione automatica del personale

Verrà implementato un sistema intelligente per l'assegnazione automatica dello staff agli eventi, in base a disponibilità, specializzazione e carico di lavoro.

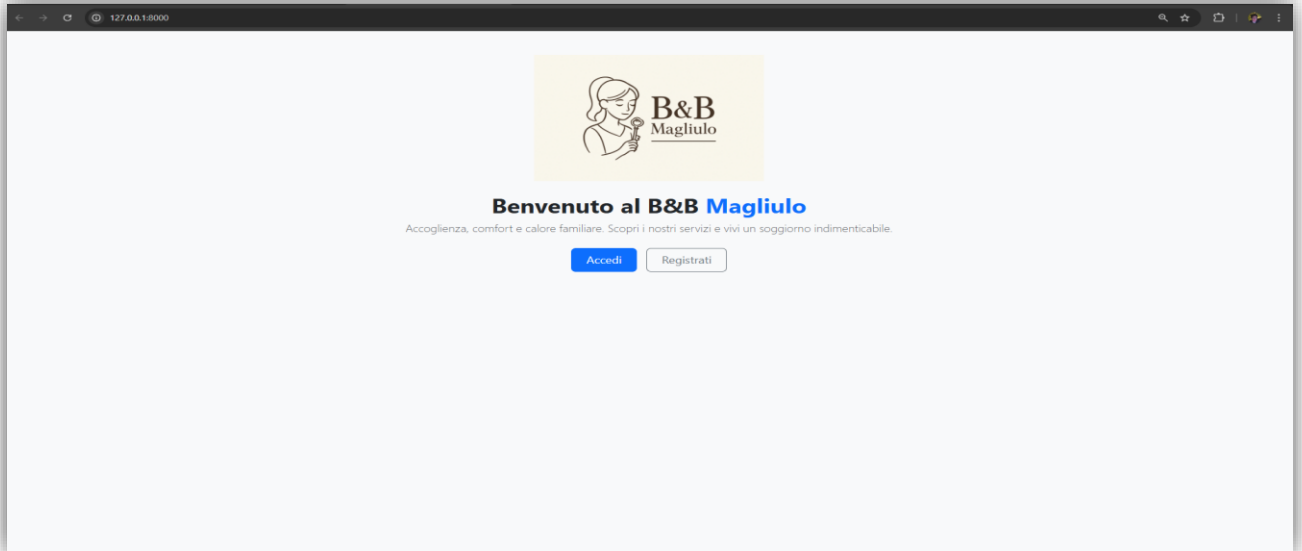
- Gestione e reportistica per i gestori

I gestori avranno accesso a funzionalità avanzate che permetteranno non solo di consultare statistiche su occupazione camere, partecipazione agli eventi, recensioni ricevute e andamento generale dell'attività, ma anche di creare e modificare camere ed eventi locali, centralizzando così la gestione operativa del B&B.

Descrizione della web app

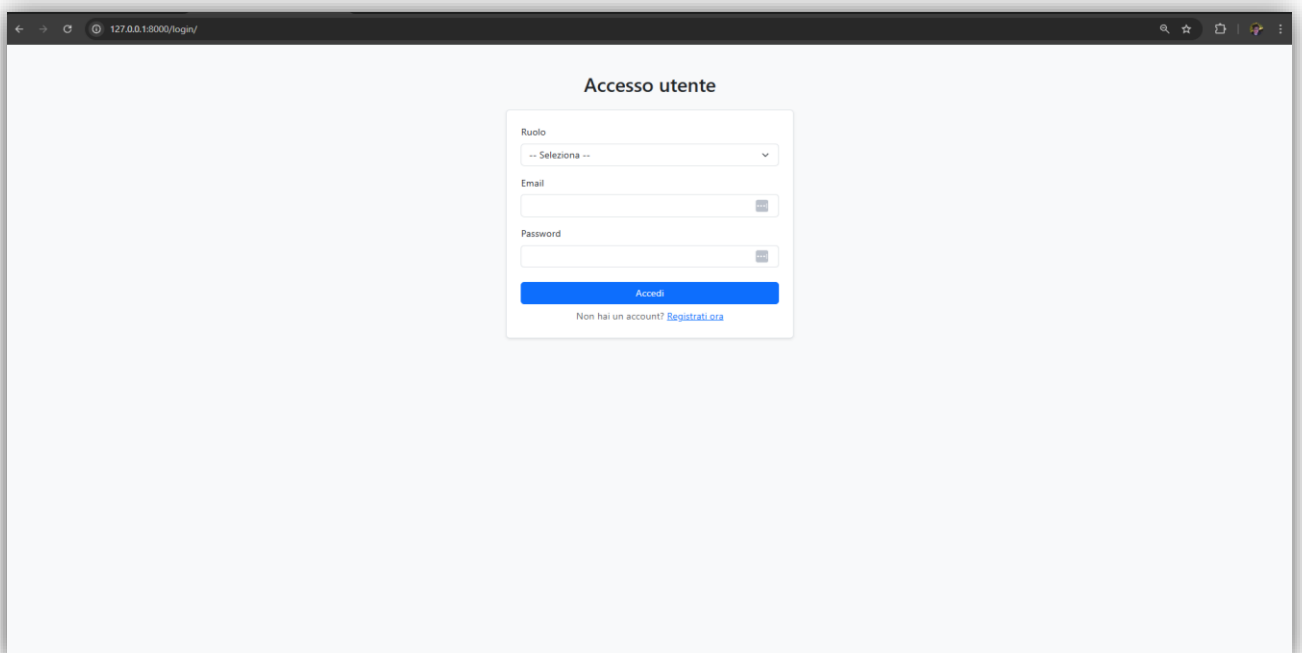
È stata implementata una web app responsive che consente agli utenti del B&B Magliulo di autenticarsi in base al proprio ruolo (Cliente, Staff, Gestore) e accedere a funzionalità specifiche, legate alla gestione delle prenotazioni, degli eventi locali e delle recensioni.

Home pubblica

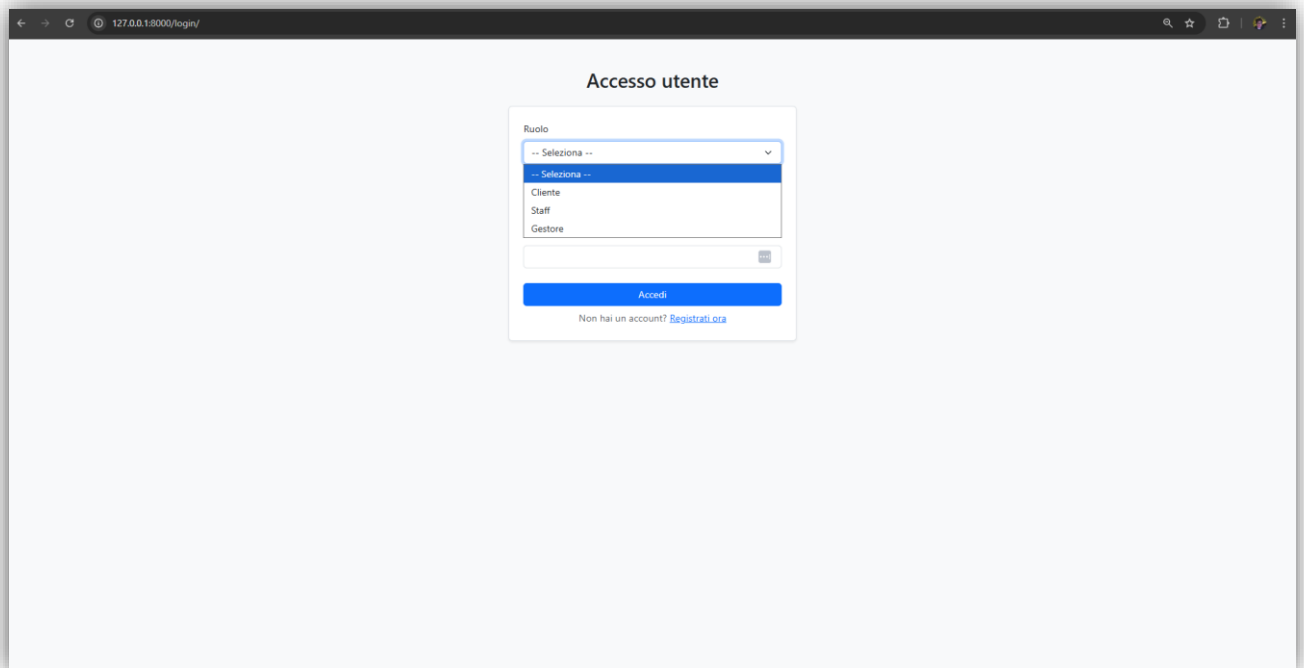


All'avvio della web app viene mostrata una schermata introduttiva con il logo e il nome della struttura, accompagnati da un messaggio di benvenuto. L'utente può scegliere se accedere oppure registrarsi, cliccando sugli appositi pulsanti.

Login utente



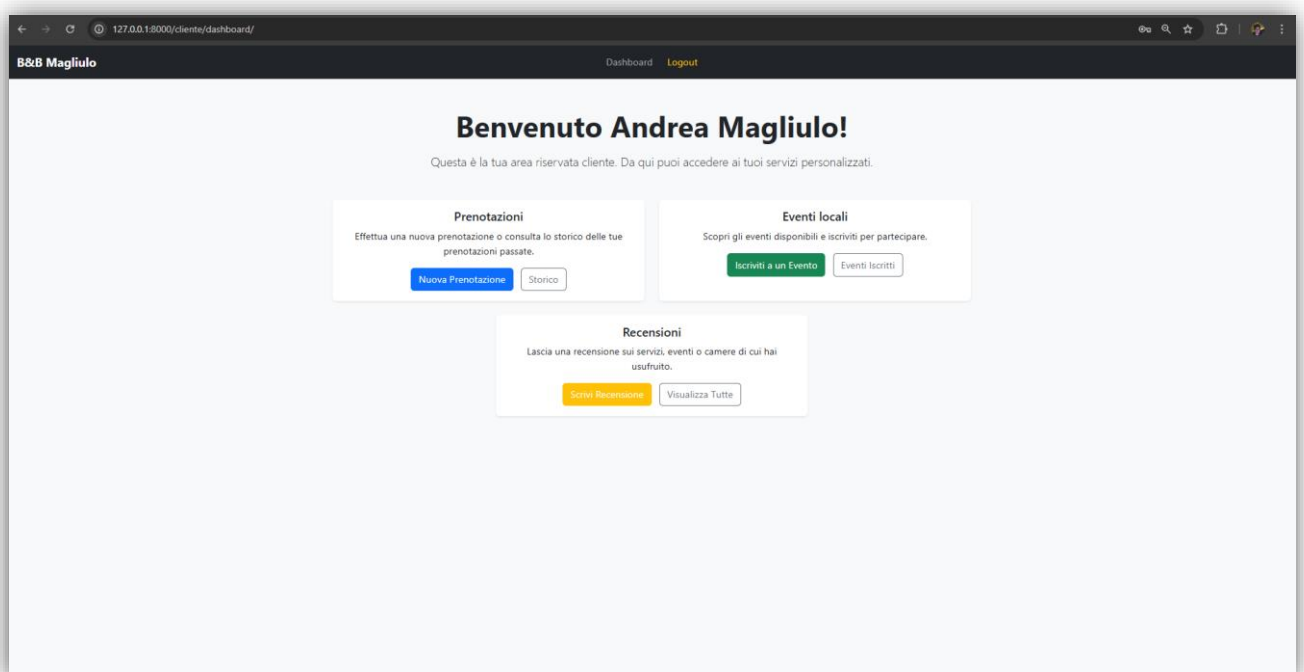
Dopo aver cliccato su “Accedi” dalla schermata iniziale, l’utente viene reindirizzato alla pagina di accesso. Qui deve selezionare il proprio ruolo dal menu a tendina (Cliente, Staff o Gestore) e inserire email e password.



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/login/'. The page title is 'Accesso utente'. It features a login form with a dropdown menu for 'Ruolo' (Role) with options: 'Seleziona --', 'Cliente', 'Staff', and 'Gestore'. Below the dropdown is an email input field and a password input field. A blue 'Accedi' button is at the bottom of the form. Below the button, there is a link: 'Non hai un account? [Registrati ora](#)'.

Il sistema verifica le credenziali e, se corrette, effettua il reindirizzamento alla dashboard personalizzata, diversa per ciascun ruolo. In caso di errore (ad esempio credenziali errate o ruolo sbagliato), viene mostrato un messaggio di errore.

Dashboard Cliente



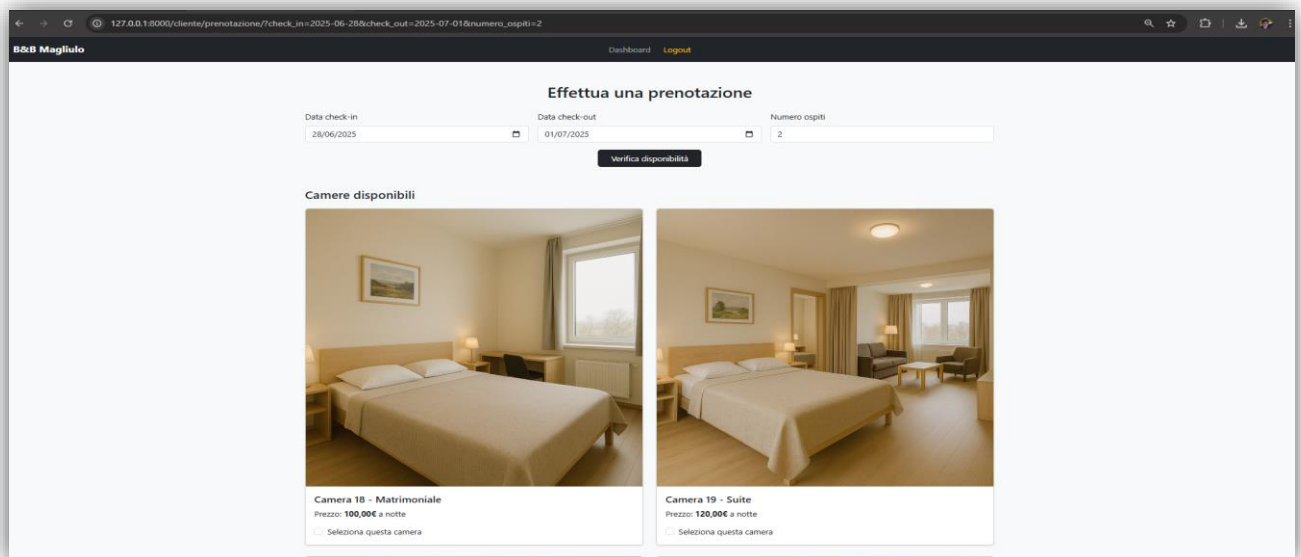
The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/cliente/dashboard/'. The page title is 'B&B Magliulo'. The dashboard is titled 'Benvenuto Andrea Magliulo!' and includes a subtitle: 'Questa è la tua area riservata cliente. Da qui puoi accedere ai tuoi servizi personalizzati.' There are three main sections: 'Prenotazioni' (Bookings) with buttons 'Nuova Prenotazione' and 'Storico'; 'Eventi locali' (Local events) with buttons 'Iscriviti a un Evento' and 'Eventi Iscriviti'; and 'Recensioni' (Reviews) with buttons 'Scrivi Recensione' and 'Visualizza Tutte'.

Dopo aver effettuato l'accesso con ruolo Cliente, l'utente viene reindirizzato alla propria area personale. In questa dashboard, chiara e suddivisa in sezioni, il cliente può gestire autonomamente le funzionalità offerte dal sistema.

Le opzioni disponibili sono:

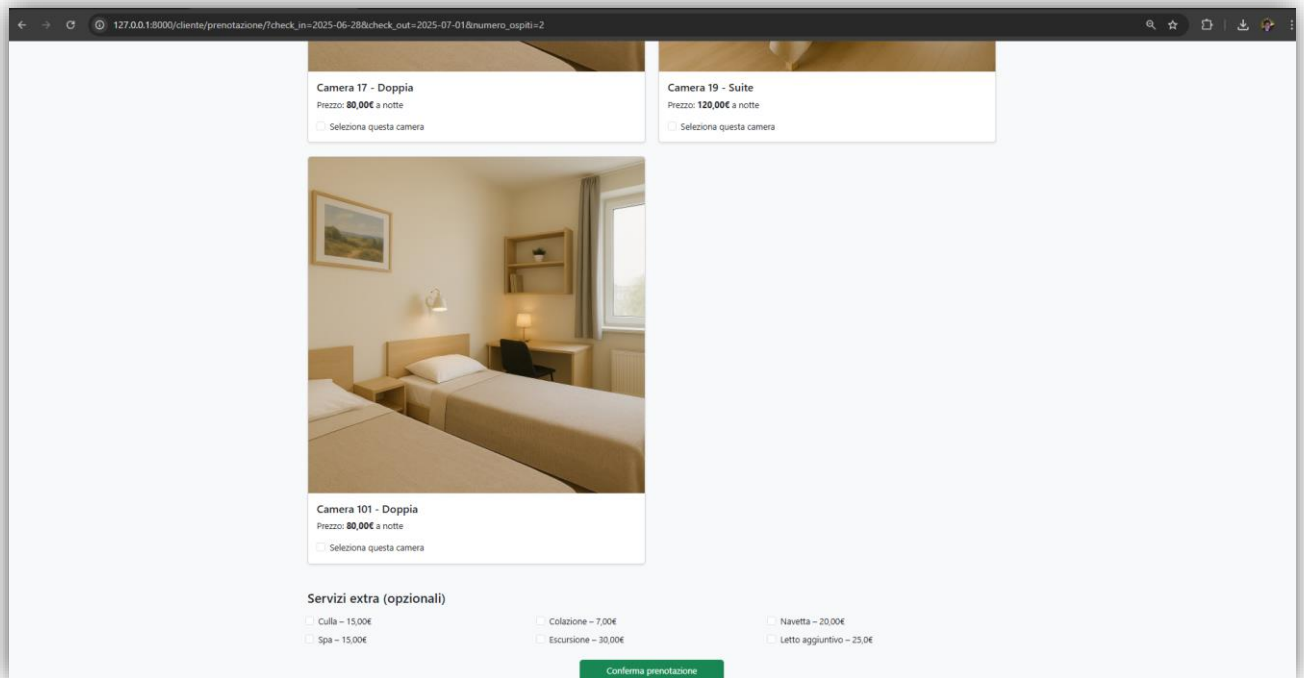
- **Prenotazioni**
Permette di effettuare una nuova prenotazione o visualizzare lo storico delle prenotazioni già effettuate.
- **Eventi locali**
Consente di esplorare gli eventi in programma (come degustazioni, escursioni o serate a tema) e iscriversi. È anche possibile consultare gli eventi a cui si è già iscritti.
- **Recensioni**
L'utente può lasciare una recensione per camere, eventi o servizi extra di cui ha usufruito. È anche possibile visualizzare l'elenco delle recensioni inviate.

Prenotazione



Dalla dashboard, il cliente può accedere alla sezione Effettua una prenotazione, dove potrà:

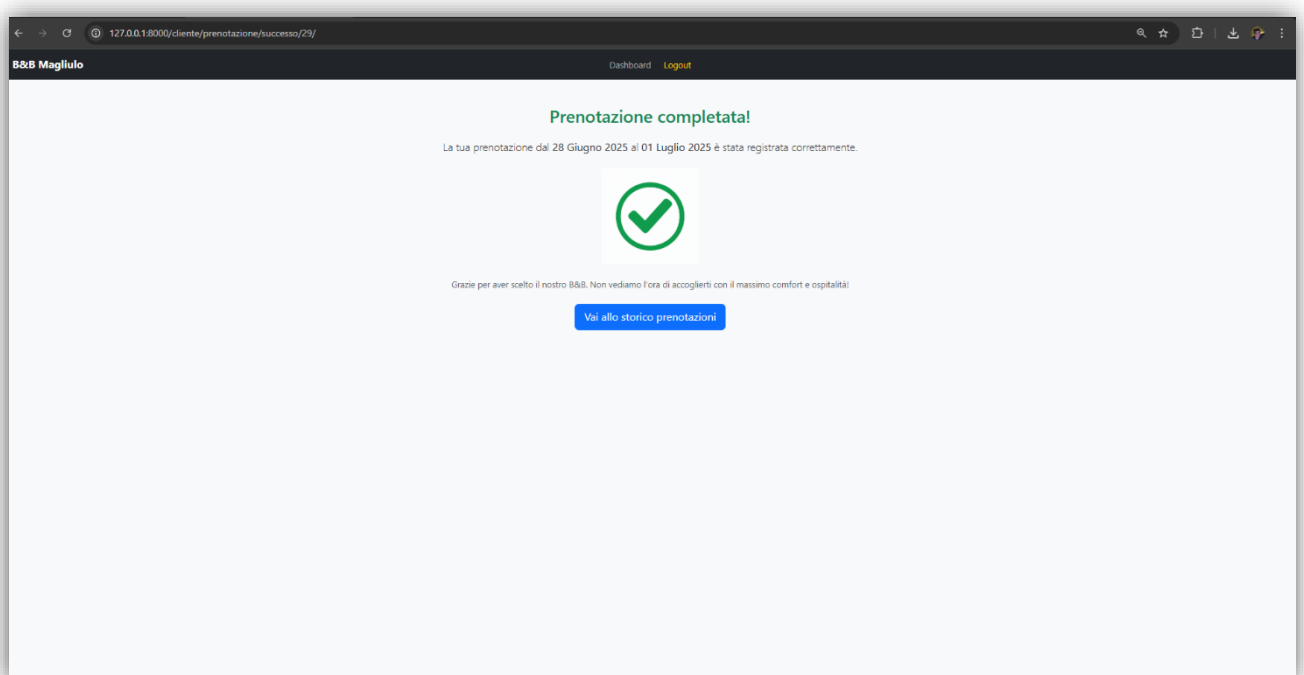
1. Inserire le date di check-in e check-out, oltre al numero di ospiti.
2. Cliccando su “Verifica disponibilità”, il sistema mostra tutte le camere disponibili per il periodo selezionato, specificando:
 - Numero e tipologia della camera (singola, doppia, matrimoniale, suite)
 - Prezzo per notte
 - Un'immagine rappresentativa della camera



Dopo aver selezionato una o più camere, il cliente può aggiungere servizi extra opzionali come: culla, letto aggiuntivo, navetta, colazione, spa, escursione.

Ogni servizio è mostrato con il relativo costo, permettendo una scelta flessibile.

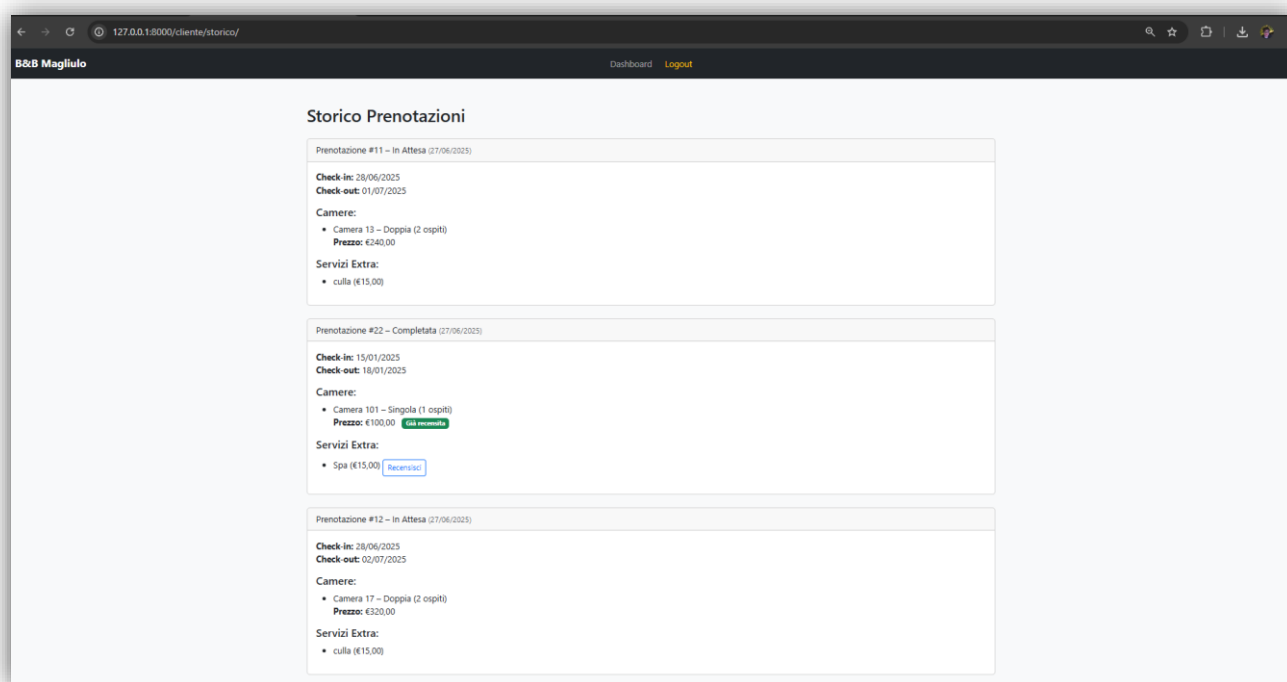
Prenotazione effettuata con successo



Dopo aver selezionato una o più camere disponibili, aggiunto eventuali servizi extra e confermato la prenotazione, l'utente viene reindirizzato a una schermata di conferma.

Qui viene mostrato un messaggio di successo con le date del soggiorno e un'icona visiva di completamento. L'utente ha anche la possibilità di accedere direttamente allo storico delle prenotazioni tramite un apposito pulsante, per rivedere tutte le prenotazioni effettuate.

Storiche prenotazioni

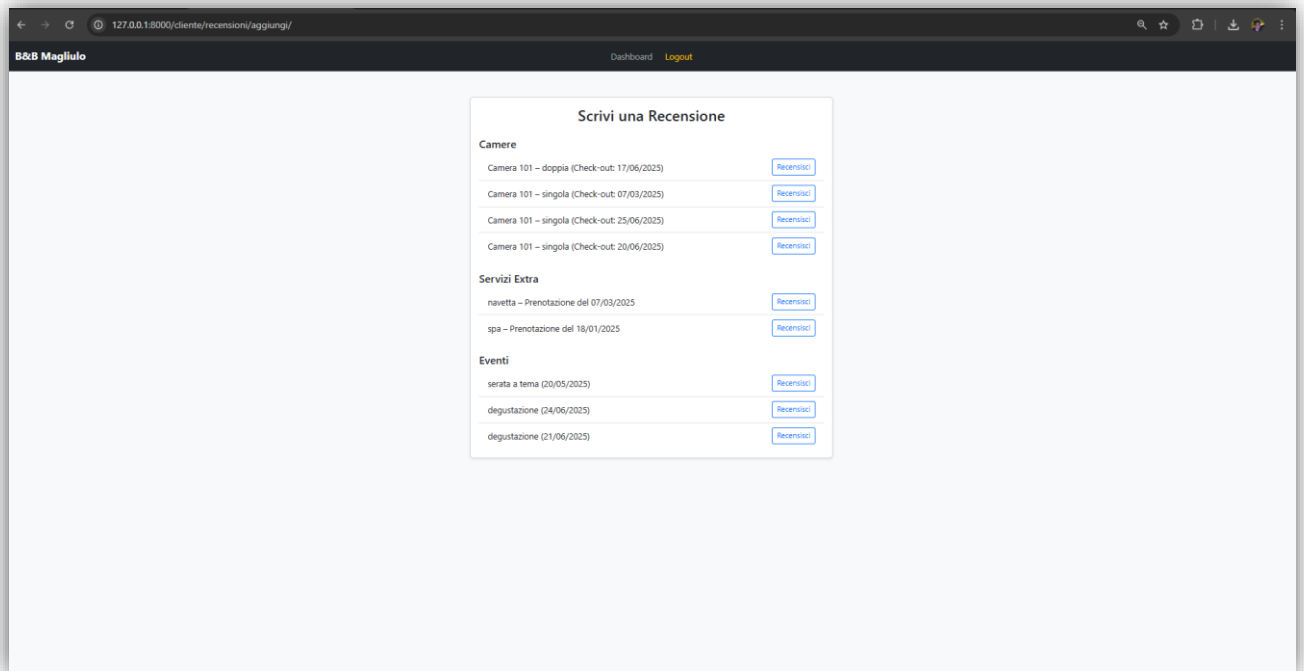


In questa sezione, l'utente può consultare tutte le prenotazioni effettuate, ordinate per data.

Per ogni prenotazione vengono mostrati:

- le date di check-in e check-out;
- il tipo di camera selezionata, con prezzo e numero di ospiti;
- l'elenco dei servizi extra aggiunti;
- lo stato della prenotazione (in attesa, in corso, completata);
- eventuali recensioni già inserite o da compilare (quest'ultimo solo in caso di prenotazione completata), con pulsante dedicato.

Recensioni



In questa sezione, il cliente può lasciare una recensione sui servizi di cui ha usufruito solo al termine del soggiorno.

Infatti, è possibile recensire camere, servizi extra ed eventi solo se la prenotazione è conclusa (check-out già avvenuto) o l’evento si è già svolto.

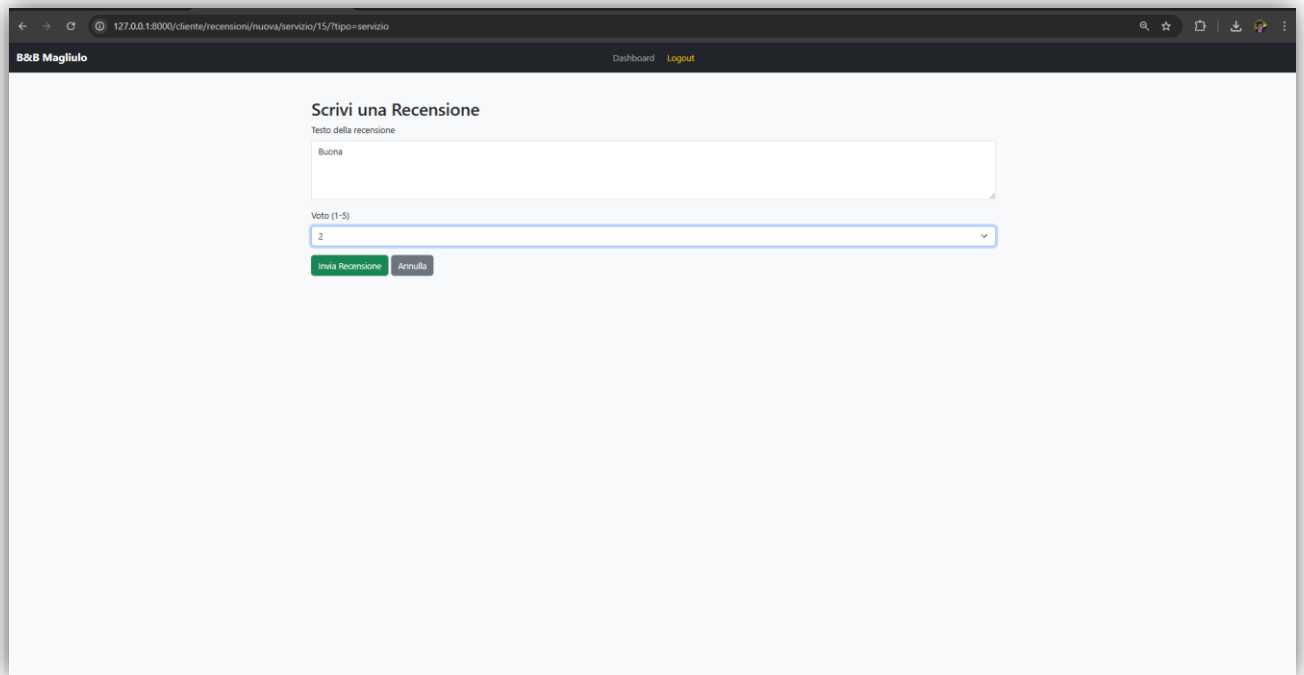
Gli elementi recensibili sono suddivisi in tre categorie:

- Camere: vengono elencate le camere utilizzate, con la relativa data di check-out.
- Servizi Extra: ogni servizio aggiuntivo selezionato nella prenotazione è elencato con la relativa data.
- Eventi: mostra gli eventi locali a cui il cliente ha partecipato.

Accanto a ciascun elemento è presente il pulsante “Recensisci”, che consente di accedere alla schermata di inserimento del voto e del commento.

In caso di recensione già effettuata, l’elemento non viene più mostrato in questo elenco, così da evitare duplicazioni.

Inserisci una recensione



The screenshot shows a web browser window with the address bar displaying '127.0.0.1:8000/cliente/recensioni/nuova/servizio/15/Tipologia-servizio'. The page title is 'B&B Magliulo'. The main content area is titled 'Scrivi una Recensione'. It contains a text input field labeled 'Testo della recensione' with the placeholder text 'Buona'. Below this is a dropdown menu labeled 'Voto (1-5)' with the value '2' selected. At the bottom of the form are two buttons: 'Invia Recensione' (green) and 'Annulla' (grey).

In questa schermata, l'utente può scrivere una recensione dettagliata per uno degli elementi selezionati (camera, servizio extra o evento).

Il modulo è composto da:

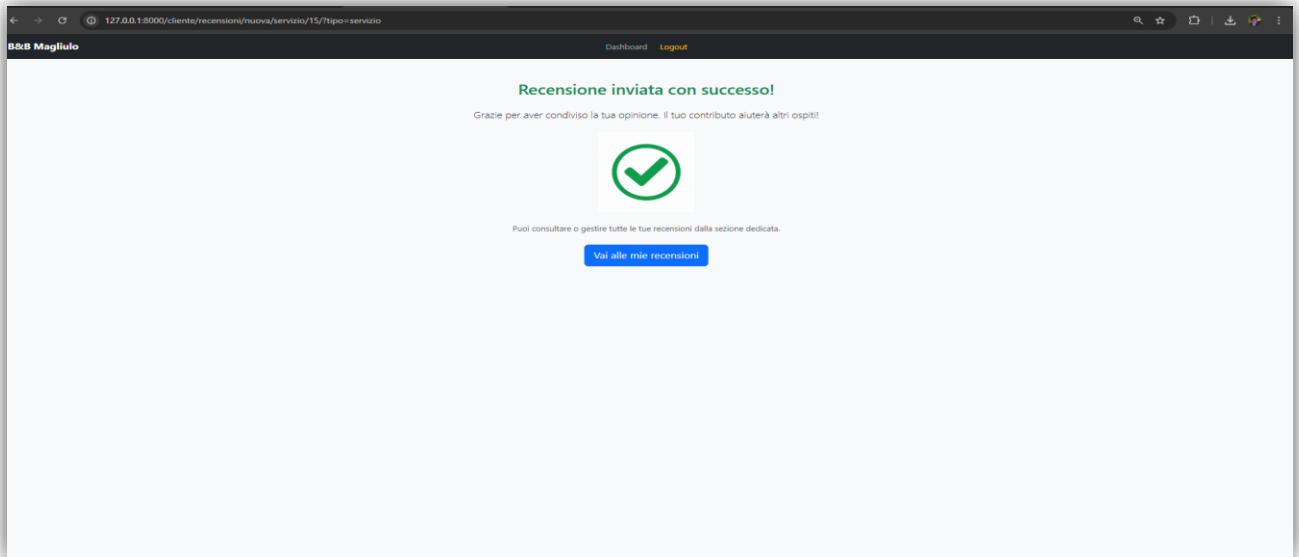
- Testo della recensione: campo libero dove il cliente può descrivere la propria esperienza.
- Voto (1–5): selezione a tendina che consente di assegnare un punteggio da 1 (scarso) a 5 (eccellente).

Sono disponibili due pulsanti:

- “Invia Recensione”: salva la recensione e la collega all'elemento specifico (servizio, in questo caso).
- “Annulla”: consente di tornare indietro senza salvare nulla.

Una volta inviata, la recensione non potrà essere modificata o duplicata.

Recensione aggiunta con successo

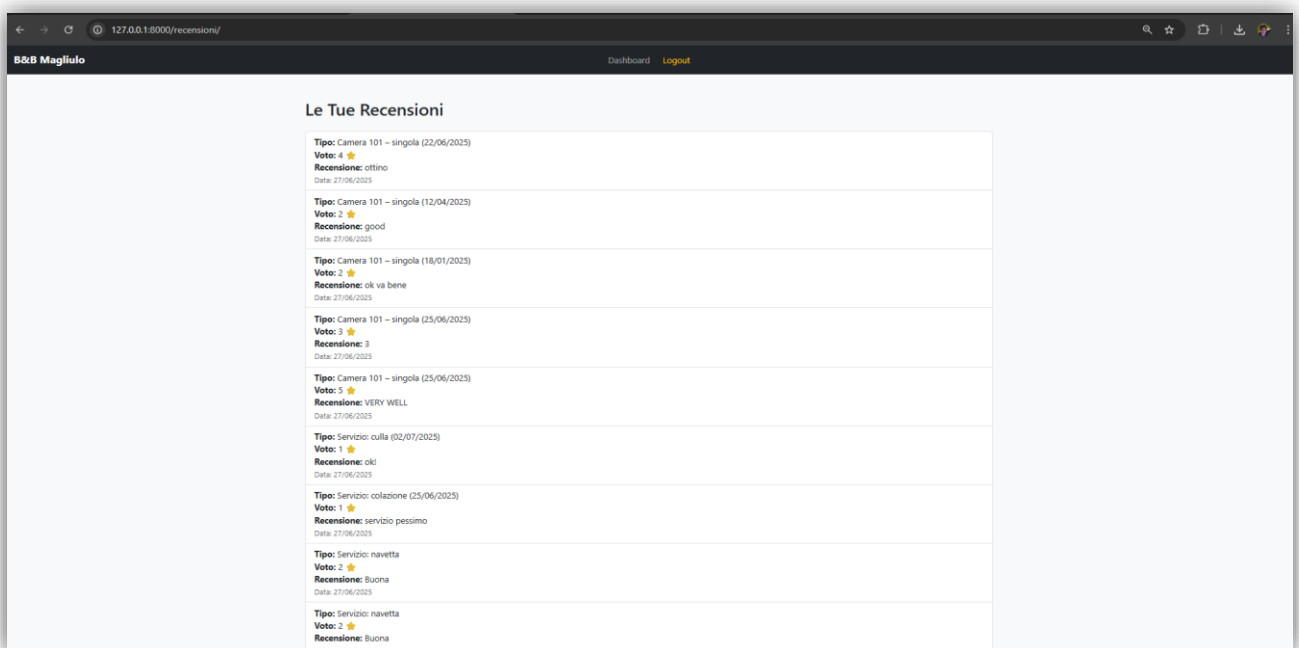


Dopo aver scritto una recensione per un servizio usufruito, come ad esempio un servizio extra legato a una prenotazione, l'utente viene reindirizzato a una schermata di conferma.

Questa pagina mostra un messaggio di successo accompagnato da un'icona visiva che conferma l'invio della recensione.

Viene inoltre fornito un pulsante che permette all'utente di consultare rapidamente lo storico di tutte le recensioni inviate, con possibilità di visualizzarle in dettaglio.

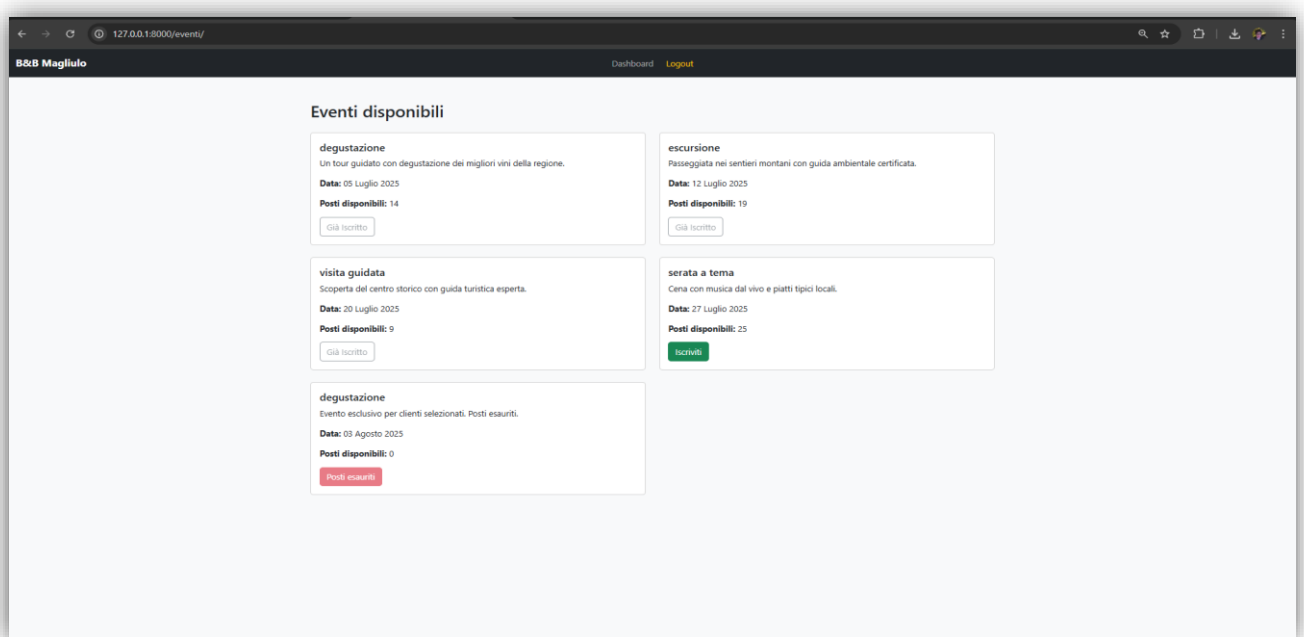
Storici recensioni



Questa schermata mostra all'utente lo storico completo di tutte le recensioni inviate, sia per camere che per servizi extra. Ogni recensione viene visualizzata con:

- il tipo e la data dell'elemento recensito (es. *Camera 101 – singola (22/06/2025)*),
- il voto espresso tramite stelle,
- il testo della recensione,
- la data in cui è stata inviata.

Evento locale



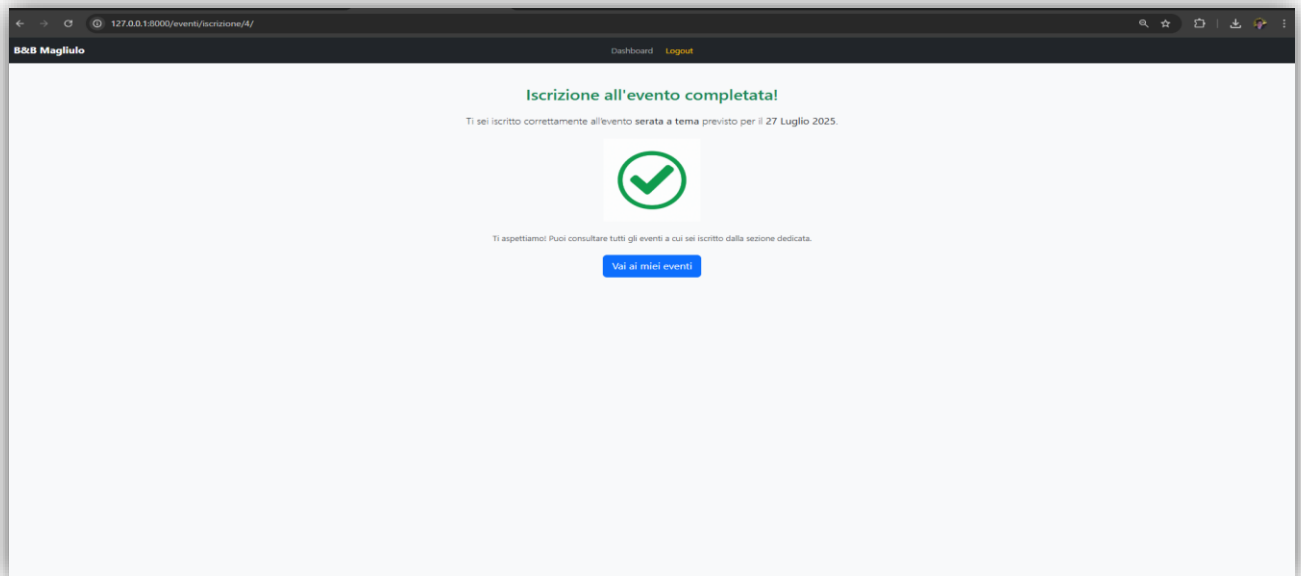
In questa schermata l'utente può visualizzare l'elenco completo degli eventi proposti dalla struttura, con tutte le informazioni utili per la partecipazione. Per ogni evento vengono mostrati:

- il titolo e una breve descrizione dell'attività,
- la data in cui si svolgerà l'evento,
- il numero di posti disponibili aggiornato in tempo reale,
- lo stato dell'iscrizione dell'utente (se già iscritto o meno).

In base alla disponibilità e alla situazione dell'utente, i pulsanti disponibili possono essere:

- "Iscriviti" (in verde), se l'utente può ancora partecipare,
- "Già iscritto" (disabilitato), se è già registrato all'evento,
- "Posti esauriti" (in rosso), se l'evento ha raggiunto la capienza massima.

Partecipazione ad evento avvenuta con successo

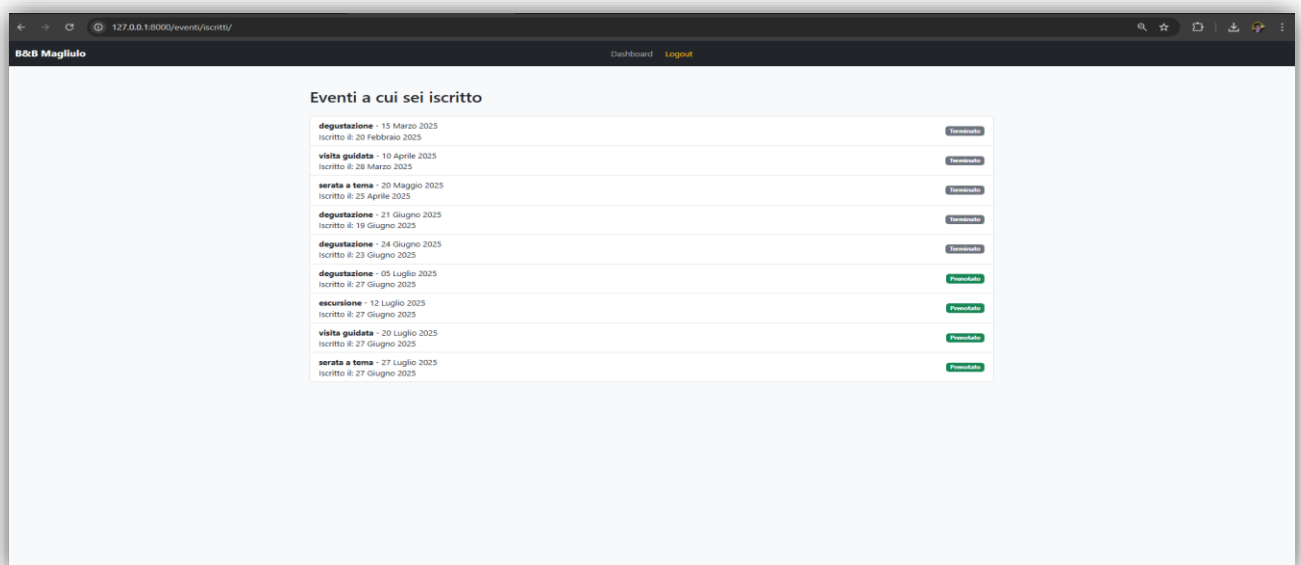


Dopo aver cliccato sul pulsante “Iscriviti” per un evento disponibile, l’utente viene reindirizzato a questa schermata di conferma.

Viene mostrato un messaggio che conferma l’avvenuta iscrizione, specificando il nome dell’evento e la relativa data. L’interfaccia include:

- un’icona visiva di conferma (check verde),
- un messaggio di ringraziamento e promemoria,
- un pulsante per accedere rapidamente alla sezione “I miei eventi”, dove l’utente può consultare tutte le iscrizioni attive.

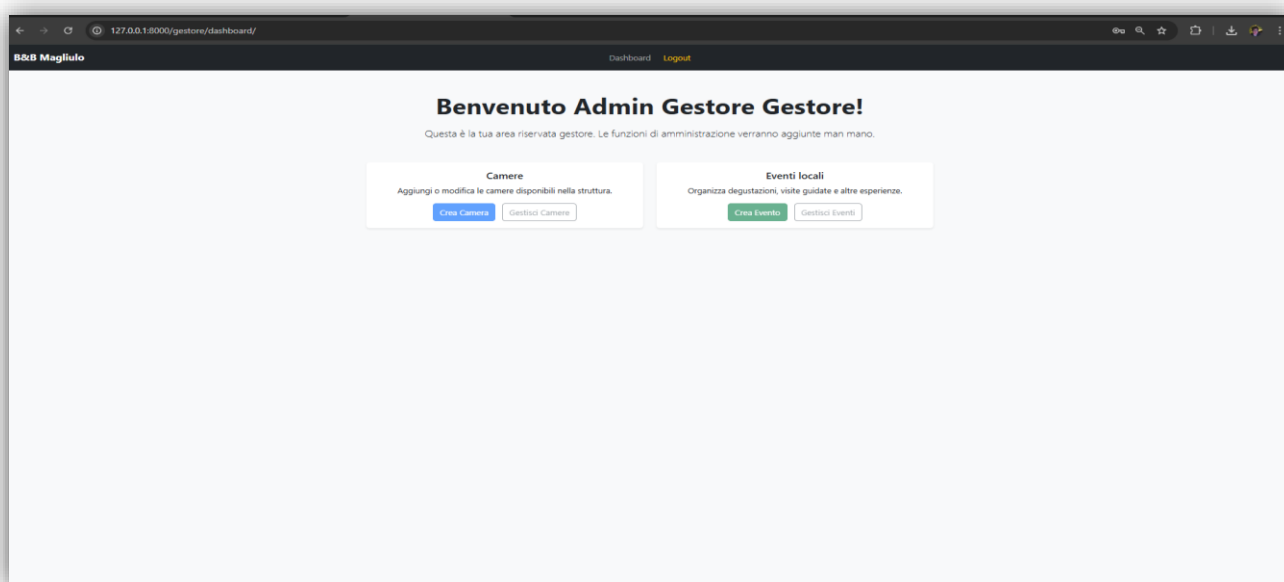
Storici eventi iscritti



In questa schermata, l'utente può visualizzare l'elenco completo di tutti gli eventi a cui ha effettuato l'iscrizione. Per ciascun evento vengono mostrati:

- il titolo e la data di svolgimento,
- la data di iscrizione,
- un badge che indica lo stato dell'evento:
 - Prenotato (verde): evento futuro a cui l'utente parteciperà;
 - Terminato (grigio): evento già svolto.

Dashboard Gestore

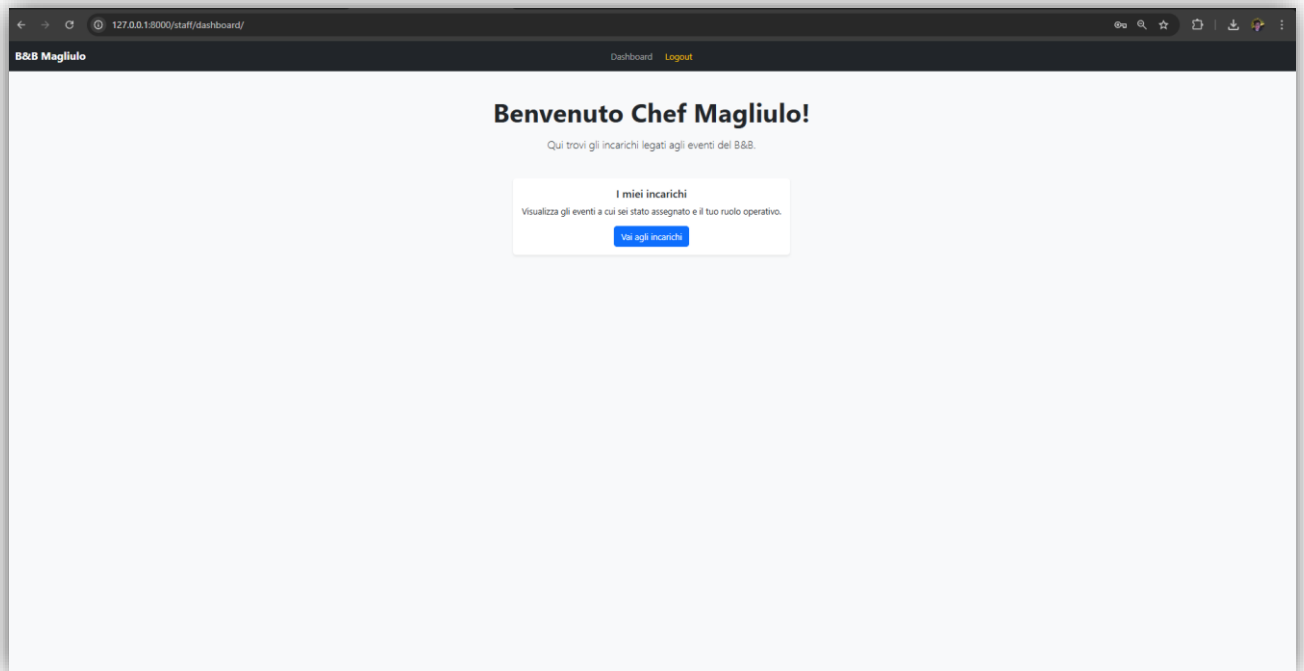


Questa schermata rappresenta l'area riservata al gestore della struttura, pensata per fornire una panoramica centralizzata delle principali funzionalità di amministrazione.

Attualmente, si tratta di una vista preliminare di tipo visivo: le funzioni mostrate non sono ancora operative, ma indicano i moduli che saranno implementati in futuro, come già anticipato in precedenza. In particolare, sono presenti due sezioni:

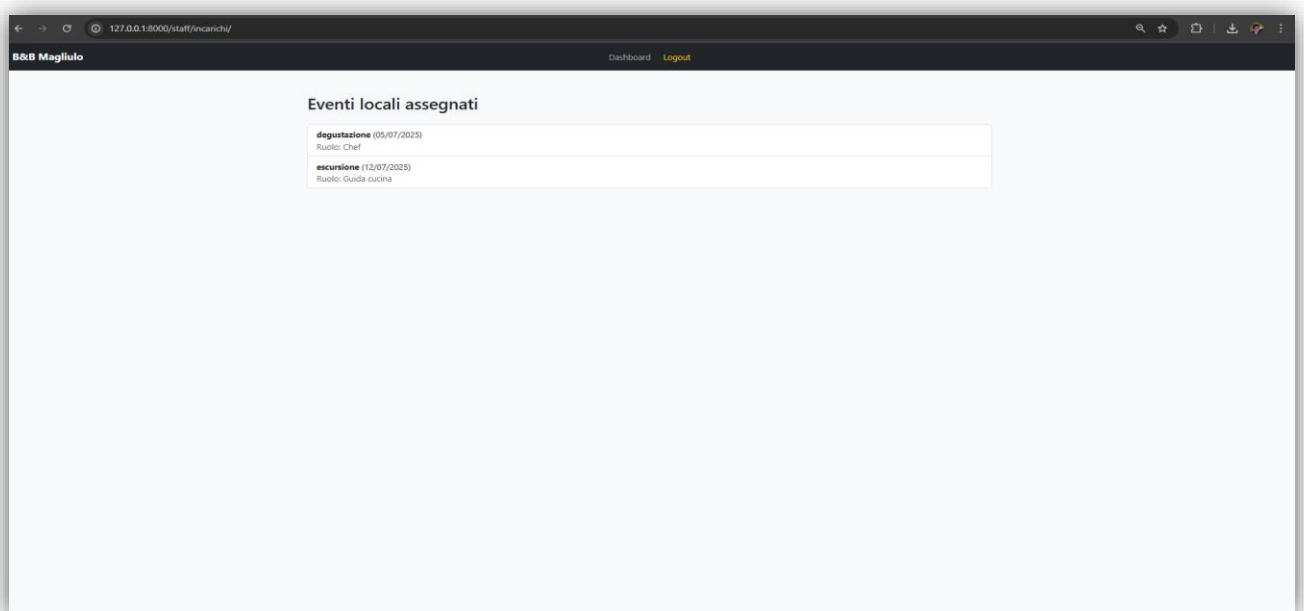
- Camere: permetterà di aggiungere nuove camere o modificare quelle esistenti nella struttura.
- Eventi locali: sarà possibile creare eventi come degustazioni, visite guidate o altri tipi di esperienze, e gestirne i dettagli.

Dashboard Staff



In questa schermata di benvenuto, uno staff con ruolo di chef accede alla propria area riservata. Viene mostrato un riepilogo sintetico che introduce alle attività legate alla struttura. Il pulsante “Vai agli incarichi” consente di visualizzare l’elenco degli eventi a cui il membro dello staff è stato assegnato, con il dettaglio del ruolo operativo previsto. In questo caso, il ruolo di chef implica partecipazione attiva in eventi locali (come serate a tema o degustazioni).

Storici eventi assegnati



In questa pagina lo staff può visualizzare l'elenco dettagliato degli eventi locali a cui è stato assegnato, con il ruolo specifico per ciascuna attività. Per ogni evento sono mostrati il nome e la data dell'evento e il ruolo operativo assegnato.

Sistema di gestione delle sessioni

Il sistema di gestione delle sessioni è stato implementato utilizzando i cookie di sessione, o cookie temporanei. Si tratta di file di testo che il browser salva sul computer dell'utente per la durata della sessione di navigazione, e che vengono eliminati automaticamente alla chiusura del browser. Non avendo una data di scadenza specifica, questi cookie vengono utilizzati per identificare l'utente durante la sessione, ad esempio memorizzando la sua email. Questa informazione viene aggiornata nel momento in cui effettua l'accesso un altro utente o un responsabile.

L'email memorizzata nella sessione viene poi utilizzata per interrogare il database, identificando l'utente o il gestore all'interno delle rispettive tabelle personalizzate, e permettendo il recupero dei dati associati (prenotazioni, recensioni, eventi assegnati, ecc.).

L'uso dei cookie di sessione è stato necessario in quanto, per il progetto, si è scelto di non utilizzare il sistema di autenticazione predefinito di Django (AbstractUser), ma di creare tabelle personalizzate per gestire separatamente clienti, gestori e staff. Di conseguenza, non è stato adottato il meccanismo integrato di login/logout di Django, bensì un sistema di autenticazione su misura, che prevede il salvataggio dell'email nella sessione per identificare l'utente attualmente loggato.

In particolare, per gestire l'accesso all'applicazione web da uno stesso dispositivo, si è resa necessaria la separazione delle sessioni. Ciò è stato possibile grazie all'utilizzo della funzione:

```
request.session.pop('email', None)
```

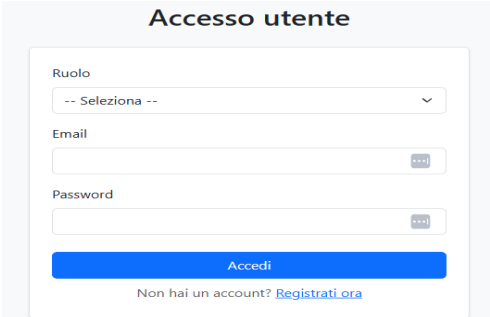
```
request.session.pop('admin_email', None)
```

```
request.session.pop('staff_email', None)
```

nel logout dei rispettivi utenti, che rimuovono la variabile email corrispondente, evitando interferenze tra le sessioni.

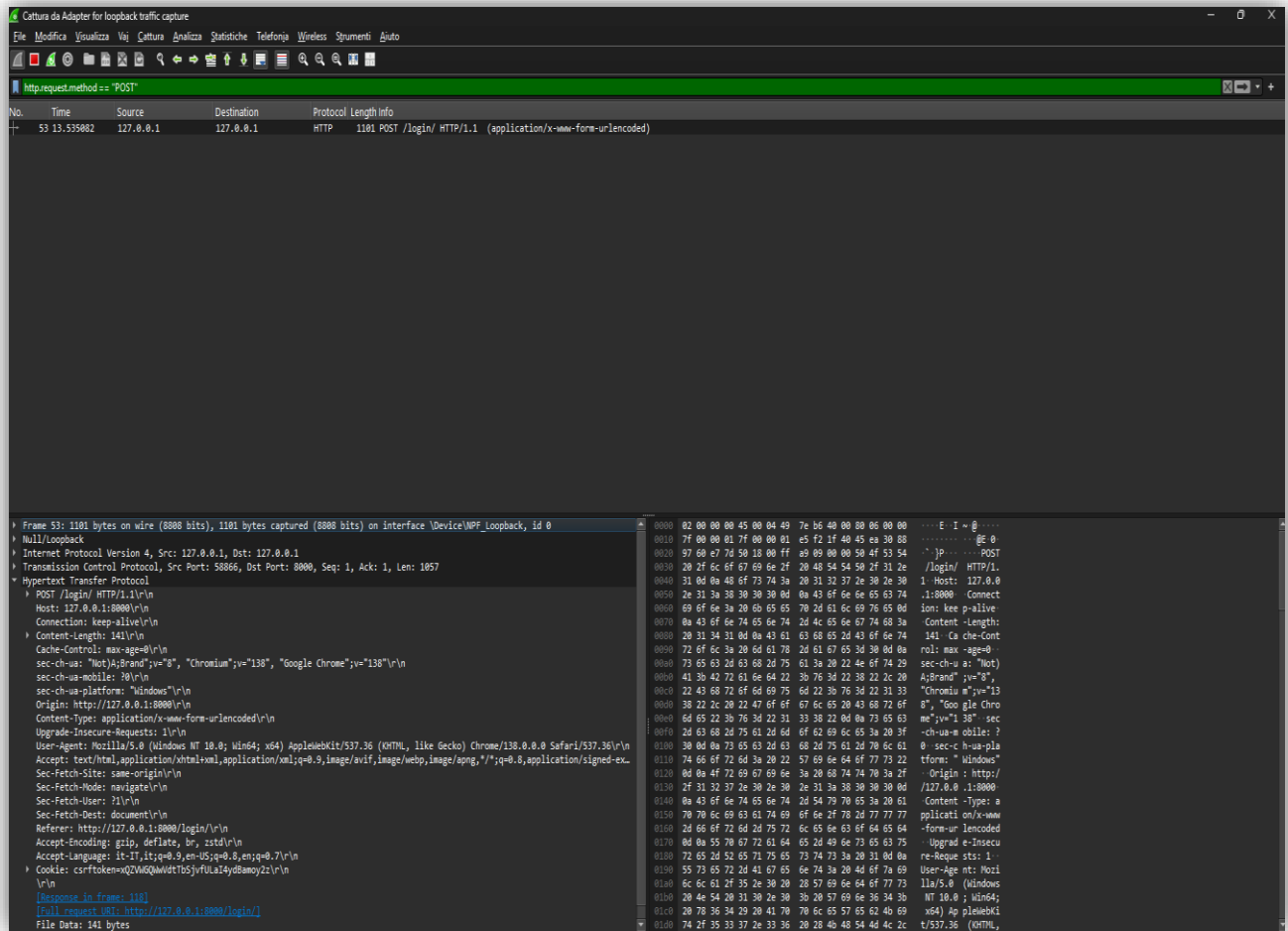
Esempio pratico:

Inizialmente si inseriscono i dati necessari per il login



The screenshot shows a login form titled "Accesso utente". It contains three input fields: "Ruolo" (a dropdown menu with "-- Seleziona --"), "Email", and "Password". Each field has a small icon to its right. Below the fields is a blue "Accedi" button. At the bottom, there is a link: "Non hai un account? [Registrati ora](#)".

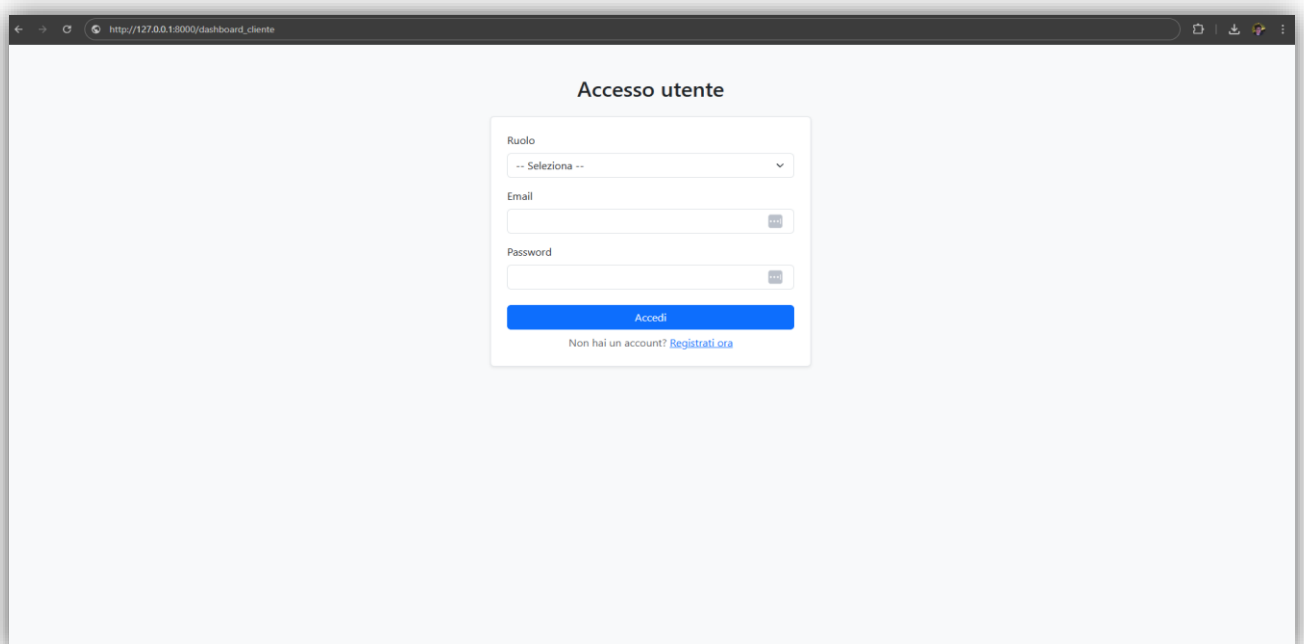
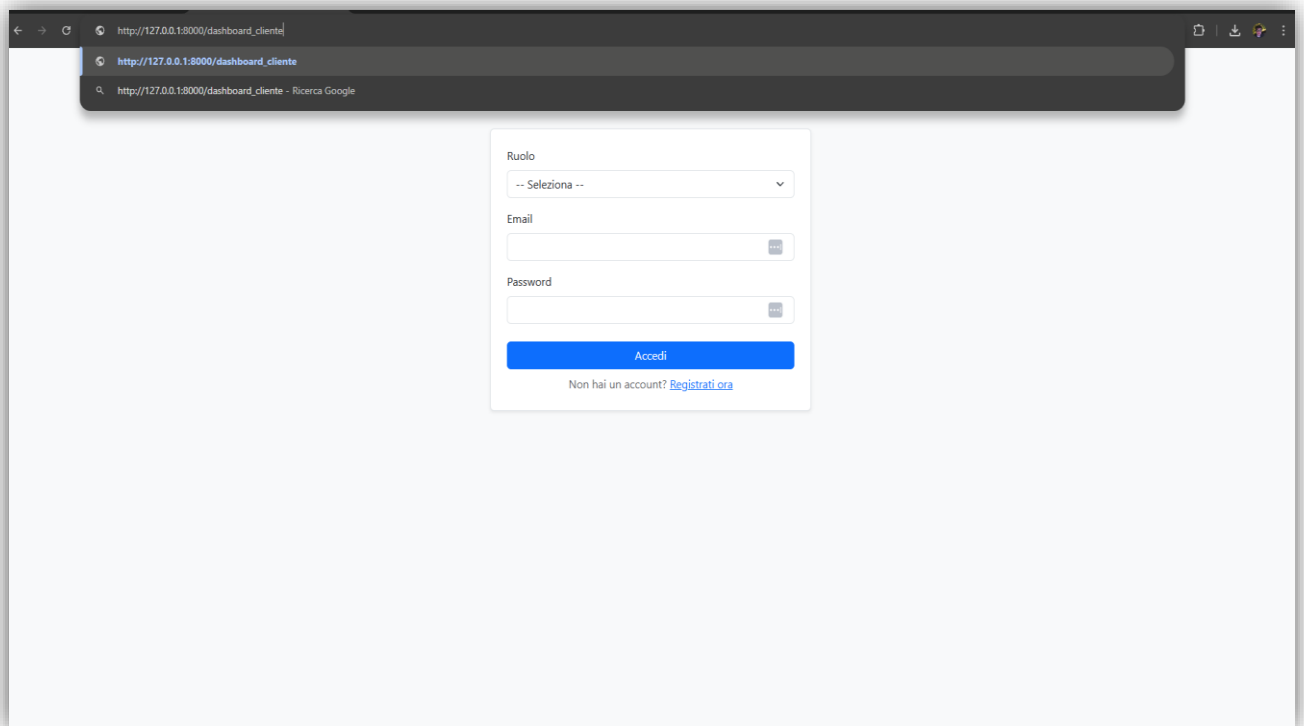
Grazie all'utilizzo dell'applicazione Wireshark è possibile analizzare il flusso del traffico HTTP. In particolare, si può osservare la presenza del parametro sessionid, che identifica la sessione dell'utente autenticato.



Inoltre, dopo il logout, se si tenta di accedere a una pagina dell'area amministrativa o personale senza aver effettuato il login, si viene automaticamente reindirizzati alla pagina di login corrispondente. Questo comportamento è dovuto al fatto che, nel momento in cui viene richiesta una di queste pagine tramite URL, viene effettuato un controllo sul cookie di sessione, in particolare sulla presenza di un flag che verifica se l'utente è autenticato, come ad esempio `is_authenticated`

```
if not request.session.get('is_authenticated'):
```

```
    return redirect('personale')
```



Implementazione SQL injection

Nel contesto dello sviluppo web, una delle vulnerabilità più comuni e pericolose è la SQL Injection. Questo tipo di attacco si verifica quando un'applicazione costruisce query SQL inserendo direttamente i dati provenienti dall'utente, senza effettuare un adeguato filtraggio o uso di meccanismi di protezione.

```
def login_vulnerabile(request):
    if request.method == "POST":
        ruolo = request.POST.get("ruolo")
        email = request.POST.get("email")
        password = request.POST.get("password")

        query = f"SELECT * FROM bb_app_utente WHERE ruolo = '{ruolo}' AND email = '{email}' AND password = '{password}'"
        print("Eseguendo query:", query)

        with connection.cursor() as cursor:
            cursor.execute(query)
            row = cursor.fetchone()

        if row:
            utente = {
                "nome": row[1],
                "cognome": row[2],
            }
            return render(request, template_name="bb_app/dashboard_cliente.html", context={"utente": utente})
        else:
            return render(request, template_name="bb_app/login_vulnerabile.html", context={
                "error": "Credenziali non valide (vulnerabile)",
                "hide_navbar": True
            })

    return render(request, template_name="bb_app/login_vulnerabile.html", context={"hide_navbar": True})
```

Questa funzione `login_vulnerabile` gestisce il login di un utente. Riceve email e password tramite il metodo POST e costruisce una query SQL concatenando direttamente i dati forniti dall'utente nella stringa SQL:

```
query = f"SELECT * FROM bb_app_utente WHERE ruolo = '{ruolo}' AND email = '{email}' AND password = '{password}'"
```

In questo modo, i valori immessi nel form (email e password) vengono inseriti senza alcun filtro all'interno della query SQL. Ciò rende il sistema vulnerabile a una **SQL Injection**, ovvero l'inserimento di comandi SQL da parte dell'utente per alterare il comportamento della query e ottenere accesso non autorizzato.

L'interazione con il database avviene tramite l'oggetto `cursor`, che rappresenta il canale di comunicazione tra Django e il database relazionale. Il costrutto:

with connection.cursor() as cursor:

gestisce automaticamente l'apertura e la chiusura della connessione, prevenendo eventuali perdite di risorse.

Dopo aver eseguito la query con `cursor.execute(query)`, viene utilizzato il metodo `cursor.fetchone()` per recuperare il risultato. Se le credenziali fornite corrispondono a un utente registrato, `row` conterrà i dati dell'utente. In caso contrario, sarà `None`.

Test con SQLmap

SQLmap è uno strumento open-source utilizzato per individuare e sfruttare automaticamente vulnerabilità di tipo SQL Injection in applicazioni web. In questo progetto è stato usato per testare la sicurezza del form di login creato intenzionalmente vulnerabile tramite la vista login_vulnerabile.

Esempio di comando usato per testare la vulnerabilità:

```
sqlmap.py -u "http://127.0.0.1:8000/login_vulnerabile" --data="email=test&password=test" --batch -dbs
```

Spiegazione dei parametri:

- -u: specifica l'URL vulnerabile, in questo caso la rotta del login non sicuro.
- --data: simula i dati POST (email e password) come se venissero inviati dal form.
- --batch: accetta le opzioni predefinite per l'automazione.
- --dbs: chiede a SQLmap di elencare i database disponibili.

Risultato dell'esecuzione del comando

Questi sono i tipi di SQL Injection che si possono provare:

- Union-Based SQLi

Esempio di payload: ' UNION SELECT NULL,NULL--

Nel log si vede che è stato testato l'attacco UNION-based, ma senza successo:

```
[20:47:58] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[20:47:58] [WARNING] POST parameter 'password' does not seem to be injectable
```

- Boolean-Based Blind SQLi

Esempio di payload: ' OR 1=1-- o ' AND 1=0— Questa tecnica verifica se la risposta cambia in base a condizioni vere o false. Nel log è evidente che sono stati testati payload boolean-based per i parametri email e password:

```
[20:47:57] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[20:47:57] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
```

- Time-Based Blind SQLi

Il tool tenta attacchi basati sul ritardo della risposta, usando funzioni come SLEEP() o IF() per capire se il parametro è vulnerabile anche senza errori evidenti.

Esempio payload: ' OR IF(1=1,SLEEP(5),0)—

Nel log vediamo che sono stati effettuati test time-based per vari database:

```
[20:47:57] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[20:47:57] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[20:47:57] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[20:47:57] [INFO] testing 'Oracle AND time-based blind'
```


- Error-Based SQLi

Si sfruttano errori del database per estrarre informazioni.

Esempio di payload: ' AND EXTRACTVALUE(1, CONCAT(0x3a, (SELECT database())))-

Nel log sono presenti test di error-based injection per diversi DBMS:

```
[20:47:57] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'\n[20:47:57] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'\n[20:47:57] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'\n[20:47:57] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
```

- Stacked Queries

Permettono di eseguire più query in una sola chiamata, separandole con:

Esempio di payload: '; DROP TABLE users—

Nel log sono testate diverse varianti per DBMS comuni:

```
[20:47:57] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'\n[20:47:57] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'\n[20:47:57] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
```

- Bypass WAF con Tamper Scripts

Il tool segnala che nessuno dei parametri testati sembra vulnerabile con i test standard e suggerisce di aumentare il livello e rischio di test o usare script di tampering per aggirare i filtri:

```
[20:47:58] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level/'\n'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mechanis\nm involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch\n'--random-agent'
```

Esempio di attacco SQL Injection

Supponiamo che un utente malevolo inserisca nella email:

' OR 1=1 –

Questa è una tautologia, poiché 1=1 è sempre vero. Di conseguenza, la condizione nel WHERE della query risulta sempre vera, restituendo risultati anche senza credenziali corrette. L'operatore -- commenta il resto della query, annullando il controllo sulla password.

Accesso utente vulnerabile

Ruolo

Cliente

Email

' OR 1=1 --

Password

Accedi (non sicuro)

Non hai un account? [Registrati](#)

È stato possibile eseguire un attacco di SQL Injection poiché nel form della pagine HTML è stato utilizzato l'attributo novalidate, che disattiva la validazione lato client, e il campo email è stato definito con type="text", invece di type="email". Questo consente all'utente di inserire input arbitrari, incluso codice SQL malevolo, senza alcun controllo preliminare.

```
<form method="post" novalidate>
  {% csrf_token %}

  <div class="mb-3">
    <label for="ruolo" class="form-label">Ruolo</label>
    <select class="form-select" name="ruolo" id="ruolo">
      <option value="cliente">Cliente</option>
      <option value="staff">Staff</option>
      <option value="gestore">Gestore</option>
    </select>
  </div>

  <div class="mb-3">
    <label for="email" class="form-label">Email</label>
    <input type="text" class="form-control" id="email" name="email">
  </div>

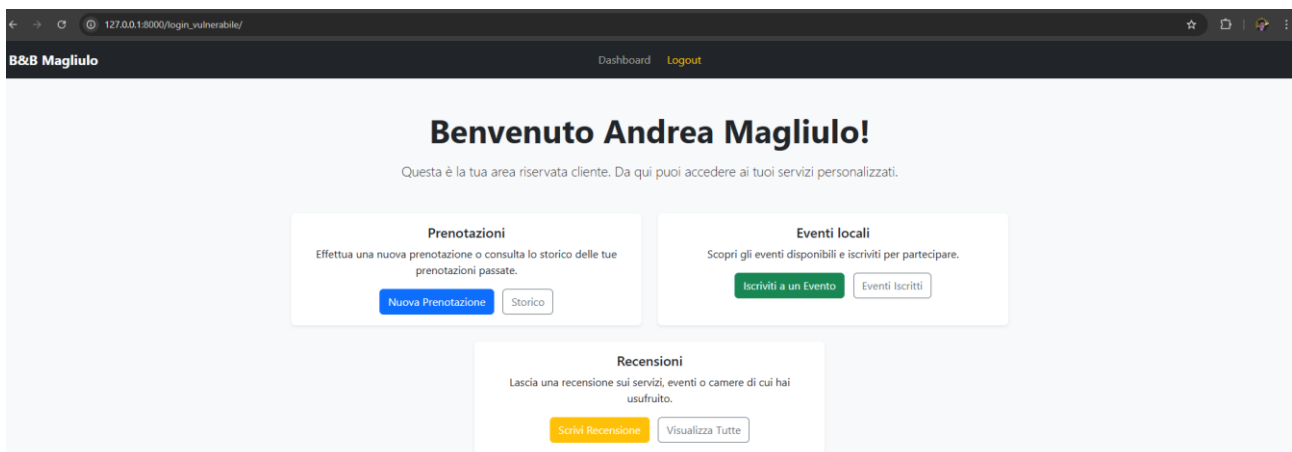
  <div class="mb-3">
    <label for="password" class="form-label">Password</label>
    <input type="password" class="form-control" id="password" name="password">
  </div>

  <button type="submit" class="btn btn-danger w-100">Accedi (non sicuro)</button>
</form>
```

Nel terminale di Django è stata eseguita la seguente query:

```
SELECT * FROM bb_app_utente WHERE ruolo = 'cliente' AND email = " OR 1=1 -- AND password = "
```

Poiché OR 1=1 è sempre vero e tutto ciò che segue dopo -- viene ignorato come commento, la condizione sulla password viene bypassata. Di conseguenza, anche senza inserire una password valida, la pagina viene mostrata con successo e riesco ad accedere comunque.



Soluzione 1: ORM di Django

Il modo più sicuro e consigliato in Django per gestire il login e prevenire SQL Injection è utilizzare l'ORM, che costruisce query parametrizzate automaticamente, evitando così l'inserimento diretto di dati non filtrati nelle query SQL.

Nella versione sicura, il login viene gestito con il filtro ORM:

```
user = Utente.objects.filter(email=email, password=hash_pwd(pwd), ruolo=ruolo).first()
```

Questa tecnica impedisce completamente le SQL injection, grazie all'utilizzo di query parametrizzate costruite da Django.

Invece, nella versione vulnerabile:

```
query = f'SELECT * FROM bb_app_utente WHERE ruolo = '{ruolo}' AND email = '{email}'  
AND password = '{password}'"
```

Un utente può facilmente aggirare l'autenticazione tramite input malevoli come: **Email: ' OR 1=1** – dimostrando così l'importanza di evitare query SQL concatenate manualmente.

Sicurezza dell'applicazione

Per garantire la protezione dei dati degli utenti e la resilienza contro attacchi comuni, sono state adottate le seguenti misure di sicurezza:

- Gestione sicura delle password

Le password non vengono mai salvate in chiaro nel database. Durante la registrazione e il login, le credenziali vengono sottoposte a una funzione di hashing basata sull'algoritmo SHA-256.

```
def _hash_pwd(raw: str) -> str:  
    return hashlib.sha256(raw.encode()).hexdigest()
```

- Protezione contro CSRF

Tutte le form HTML includono il token CSRF fornito da Django (`{% csrf_token %}`), che impedisce l'invio di richieste fraudolente da altri domini. Django verifica automaticamente questo token in ogni POST, impedendo attacchi di tipo Cross-Site Request Forgery.