

Relazione per il corso di

Basi di Dati

Agriturismo

Alessandro Rebosio

Filippo Ricciotti

27 settembre 2025

Anno Accademico 2024-2025

Indice

1	Analisi dei requisiti	3
1.1	Intervista	3
1.2	Estrazione dei concetti principali	4
2	Progettazione concettuale	5
2.1	Schema iniziale	5
2.2	Raffinamenti proposti	6
2.2.1	Utente e Dipendente	6
2.2.2	Prenotazione Servizi	7
2.2.3	Prodotti e ordini	8
2.3	Schema concettuale finale	9
3	Progettazione Logica	10
3.1	Stima del volume di dati	10
3.2	Descrizione operazioni	11
3.3	Analisi delle operazioni	11
3.4	Analisi delle ridondanze	11
3.5	Riepilogo Operazioni	11
3.6	Raffinamento dello schema	11
3.6.1	Rimozione gerarchie	11
3.6.2	Scelta degli identificatori principali	11
3.6.3	Scelte progettuali significative	12
3.7	Schema relazionale finale	13
4	Progettazione della Base di Dati	14
4.1	Check	14
4.2	Viste	14
4.3	Trigger	15
4.4	Traduzione delle operazioni	16
4.4.1	Visualizzazione statistiche dashboard	16
4.4.2	Prenotazione servizi	18
4.4.3	Gestione recensioni	19
4.4.4	Gestione iscrizioni eventi	20
4.4.5	Esecuzione prenotazioni confermate	21
4.4.6	Eliminazione prenotazioni e iscrizioni	23
4.4.7	Autenticazione utente	24
4.4.8	Gestione ordini prodotti	25

5	Progettazione dell'applicazione	26
5.1	Barra di Navigazione	26
5.2	Interfaccia Utente	28
5.3	Interfaccia Amministratore	32
A	Guida Utente	33
A.1	Clonazione del repository	33
A.2	Installazione delle dipendenze	33
A.3	Creazione del database	33
A.4	Avvio dell'applicazione	34

Capitolo 1

Analisi dei requisiti

1.1 Intervista

L'agriturismo intende dotarsi di una piattaforma digitale che razionalizzi le attività quotidiane e migliori l'esperienza dei clienti, integrando in un unico ambiente la gestione del personale, la vendita di prodotti e la promozione di eventi. Il titolare desidera uno strumento accessibile via web, utilizzabile da utenti registrati e dal personale autorizzato, in grado di offrire una visione chiara e centralizzata delle informazioni operative, riducendo errori e tempi di coordinamento.

Il cuore dell'applicativo è costituito da un catalogo di prodotti e da un calendario di eventi, visibili ai visitatori e consultabili dagli utenti registrati. I prodotti, identificati da un codice univoco, sono descritti da un nome e da un prezzo, con la garanzia che i valori economici rimangano sempre positivi. Gli eventi, invece, sono presentati con titolo, descrizione, data di svolgimento e un numero di posti disponibili; la loro pubblicazione è effettuata da dipendenti autorizzati, così da mantenere controllo e coerenza dell'offerta.

Gli utenti potranno creare un account fornendo un nome utente, un indirizzo email e una password; ogni profilo sarà associato a una persona identificata tramite codice fiscale, così da assicurare un'anagrafica pulita e non ridondante. Una volta autenticati, gli utenti potranno consultare il catalogo, comporre ordini di acquisto di prodotti e completarne la registrazione: ogni ordine sarà tracciato con data e ora, e conterrà le righe di dettaglio con quantità e prezzo unitario, in modo da consentire il calcolo del totale e la successiva rendicontazione. Gli acquisti rimarranno associati in modo permanente all'account dell'utente, così da poterli rivedere e analizzare nel tempo.

Per la dimensione esperienziale dell'agriturismo, la piattaforma offrirà una sezione dedicata agli eventi: gli utenti interessati potranno iscriversi indicando il numero di partecipanti; il sistema dovrà garantire che le prenotazioni non superino i posti disponibili e registrerà automaticamente data e ora di ciascuna iscrizione. In questo modo, il titolare potrà monitorare in tempo reale l'andamento delle adesioni e prevedere l'affluenza, ottimizzando l'organizzazione delle serate e delle attività tematiche.

La gestione del personale rappresenta un altro pilastro del sistema. Ciascun dipendente sarà un utente abilitato a funzioni specifiche e caratterizzato da un ruolo (ad esempio sala, cucina, reception), con la possibilità di tracciarne lo storico delle variazioni nel tempo. La pianificazione dei turni avverrà attraverso la definizione di modelli di turno (per giorno della settimana, con orari di inizio e fine) e la loro assegnazione a calendario per una data specifica. Ogni assegnazione prevede uno stato — programmato, completato o assente — così da fotografare l'effettiva presenza; inoltre, il sistema eviterà conflit-

ti, impedendo che uno stesso dipendente risulti assegnato a più turni nella medesima giornata.

Dal punto di vista direzionale, il titolare richiede una reportistica essenziale ma affidabile: l'andamento delle vendite per periodo, la partecipazione agli eventi e un quadro della presenza/assenza del personale sui turni. La piattaforma dovrà salvaguardare la sicurezza dei dati, conservando le password in forma sicura e applicando vincoli di integrità su prezzi e quantità; le operazioni frequenti — come consultare il catalogo, registrare un ordine o iscriversi a un evento — dovranno risultare rapide e semplici, privilegiando chiarezza e immediatezza d'uso.

1.2 Estrazione dei concetti principali

L'agriturismo intende realizzare una piattaforma digitale che unisca in un unico ecosistema la vendita di prodotti, la promozione e gestione degli eventi e l'organizzazione del personale. Il sistema sarà accessibile via web agli utenti registrati e al personale autorizzato, con l'obiettivo di offrire una vista centralizzata e coerente delle attività quotidiane, riducendo errori operativi e tempi di coordinamento. Il cuore dell'applicazione è rappresentato da un **catalogo di prodotti** e da un calendario eventi: i prodotti, identificati in modo univoco (**codice**), e descritti da **nome** e **prezzo**, saranno acquistabili dagli utenti autenticati; gli eventi, caratterizzati da **titolo**, **descrizione**, **data** e **posti disponibili**, saranno visibili e prenotabili secondo regole di capienza stabilite dall'azienda.

Gli utenti potranno creare un account fornendo **nome utente**, **email** e **password**; ogni account sarà associato a una persona identificata da **codice fiscale**, in modo da mantenere un'anagrafica solida e priva di duplicati. Una volta autenticati, gli utenti potranno consultare il catalogo e comporre ordini, che verranno registrati con **data** e **ora** e articolati in righe d'ordine di dettaglio con **quantità** e **prezzo unitario**, garantendo la correttezza dei totali e la tracciabilità nel tempo. Gli acquisti resteranno permanentemente associati al profilo dell'utente, consentendo storicizzazione e successive analisi gestionali.

La dimensione esperienziale sarà supportata da un modulo eventi: la creazione degli eventi è affidata a dipendenti autorizzati e prevede l'indicazione dei **posti disponibili**. Gli utenti potranno isciversi (iscrizione evento) specificando il **numero di partecipanti**, mentre il sistema dovrà prevenire il superamento della capienza e registrare automaticamente **data** e **ora** di ogni iscrizione. In parallelo, la gestione interna del personale è fondata su ruoli e turni: ogni dipendente possiede un **ruolo** corrente, con storico delle variazioni per fini di audit, e partecipa a una pianificazione che combina modelli di turno (**giorno della settimana**, **nome**, **orari**) con assegnazioni di turno a calendario per **date** specifiche. Ogni assegnazione registra lo **stato** effettivo (**programmato**, **completato**, **assente**) e impedisce conflitti, evitando che un dipendente risulti pianificato su più turni nello stesso giorno.

A livello trasversale, la piattaforma tutela integrità e sicurezza dei dati: **prezzi** e **quantità** devono essere sempre positivi, le relazioni fra utenti, dipendenti, ordini ed eventi rispettano vincoli referenziali, e le informazioni sensibili come le **password** sono gestite in modo sicuro.

Il titolare dispone di una visione complessiva tramite report essenziali su vendite, adesioni agli eventi e presenza del personale, mentre l'interfaccia privilegia semplicità e rapidità nelle operazioni più frequenti.

Capitolo 2

Progettazione concettuale

In questo capitolo presenteremo lo schema ER, partendo da una versione iniziale e migliorandola passo dopo passo ad arrivare a quella definitiva, attraverso dei raffinamenti.

2.1 Schema iniziale

Dopo aver eseguito l'analisi del dominio iniziale, abbiamo creato uno schema di base con le entità e le relazioni principali, che sarà poi perfezionato nei passaggi successivi.

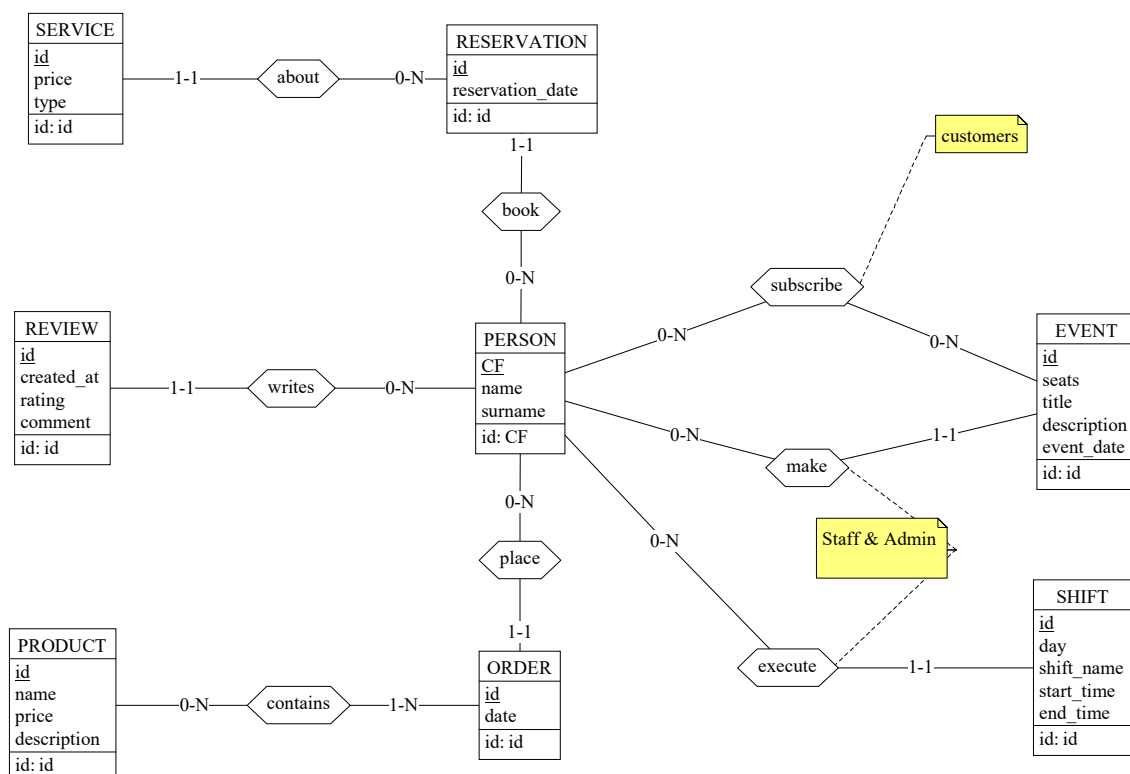


Figura 2.1: Schema ER iniziale

2.2 Raffinamenti proposti

2.2.1 Utente e Dipendente

Nel modello concettuale iniziale la **Persona** raggruppava tutte le possibili interazioni con il sistema: iscrizione, creazione di eventi, prenotazioni, ordini e recensioni. Questo approccio, sebbene corretto dal punto di vista logico, risultava poco chiaro perché attribuiva a un'unica entità responsabilità molto eterogenee.

Per migliorare la rappresentazione è stato introdotto un raffinamento mediante generalizzazione/specializzazione: la superclasse **Persona** è stata mantenuta per raccogliere gli attributi comuni (CF, nome, cognome), mentre le funzionalità specifiche sono state assegnate ai sottotipi **Cliente** e **Dipendente**.

In questo modo i clienti gestiscono attività come acquisti, recensioni, ordini e iscrizioni agli eventi, mentre i dipendenti si occupano della creazione degli eventi e della gestione dei servizi. Tale raffinamento migliora la chiarezza semantica del modello, riduce le ambiguità e riflette meglio la separazione dei ruoli reali all'interno del dominio applicativo.

Il raffinamento mette in evidenza anche le dipendenze temporali (come la gestione dei turni **Shift** o la cronologia del personale **Employee History**) e garantisce che ogni operazione rispetti vincoli di consistenza e cardinalità, rendendo il modello complessivo coerente, sicuro e facilmente estendibile.

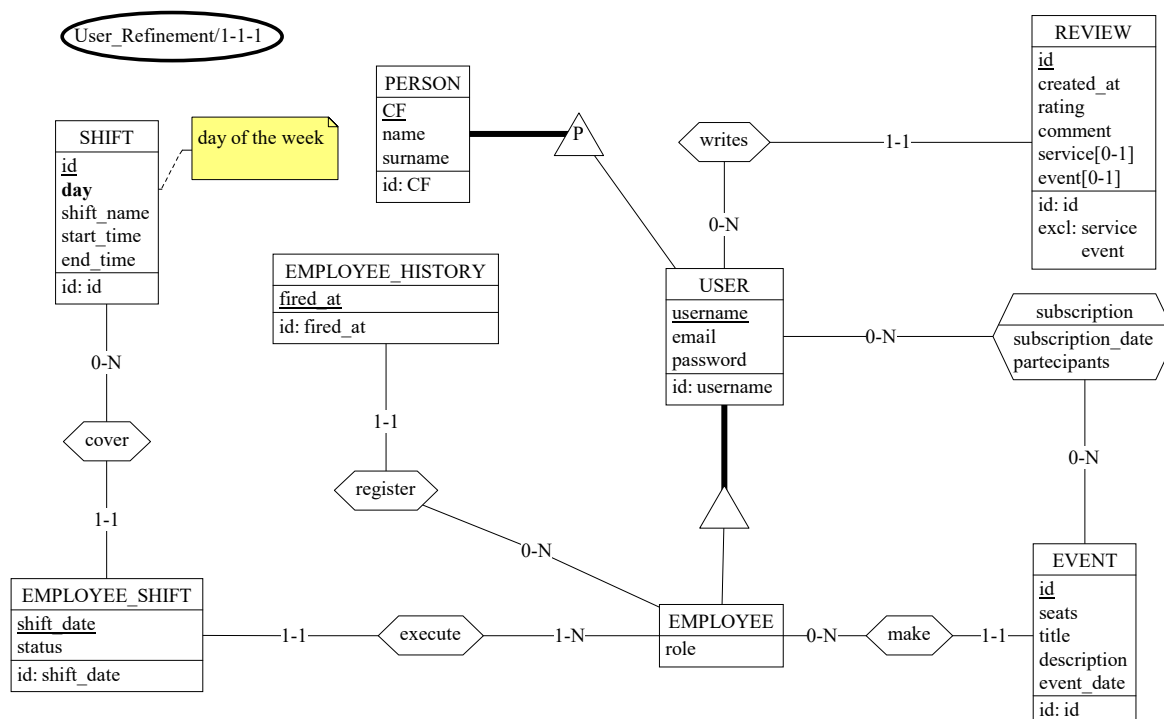


Figura 2.2: Raffinamento utente e dipendente

2.2.2 Prenotazione Servizi

Nel modello iniziale i diversi tipi di servizi potevano essere rappresentati come entità distinte, con il rischio però di ridondanza e frammentazione dei dati.

Con il raffinamento si è introdotta una **generalizzazione**: è stata creata la superclasse **Servizio**, che raccoglie gli attributi comuni (id, price, type), mentre ciascuna tipologia specifica di servizio (Camera e Ristorante) è modellata come sottoclasse.

Inoltre, è stato introdotto il legame con l'entità **Prenotazione**, che consente di registrare le informazioni su data di inizio e fine e di associare ogni prenotazione a uno o più servizi specifici tramite la relazione con **Dettagli Prenotazione**. Questo raffinamento permette di gestire correttamente scenari in cui un utente prenota più servizi differenti nello stesso arco temporale.

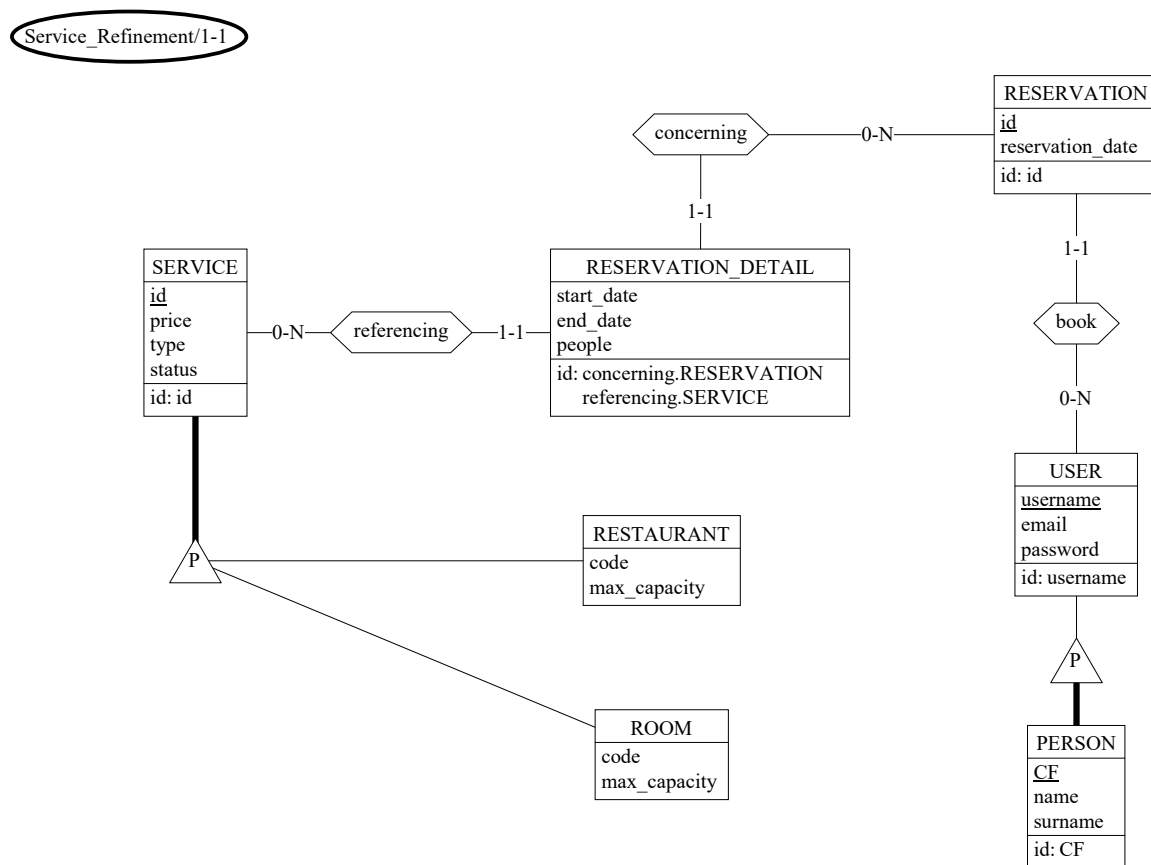


Figura 2.3: Raffinamento prenotazione e servizi

2.2.3 Prodotti e ordini

Nel modello concettuale iniziale, la gestione degli ordini e dei prodotti risultava poco dettagliata: un ordine era semplicemente collegato a uno o più prodotti, senza possibilità di specificare informazioni aggiuntive come quantità o prezzo unitario.

Con il raffinamento, è stata introdotta l'entità **Dettaglio Ordine**, che funge da associazione tra **Ordine** e **Prodotto**. Ogni dettaglio ordine consente di memorizzare, per ciascun prodotto incluso in un ordine, la quantità acquistata e il prezzo applicato. Questo permette di rappresentare in modo accurato scenari reali come ordini multiprodotto, applicazione di sconti o variazioni di prezzo nel tempo.

Inoltre, viene mantenuta la generalizzazione tra **Persona** e **Utente**, già introdotta nei raffinamenti precedenti, per distinguere i dati anagrafici comuni da quelli specifici per l'accesso al sistema e la gestione degli ordini. Questo approccio migliora la flessibilità e la chiarezza del modello, consentendo una gestione più efficace delle informazioni relative agli acquisti.

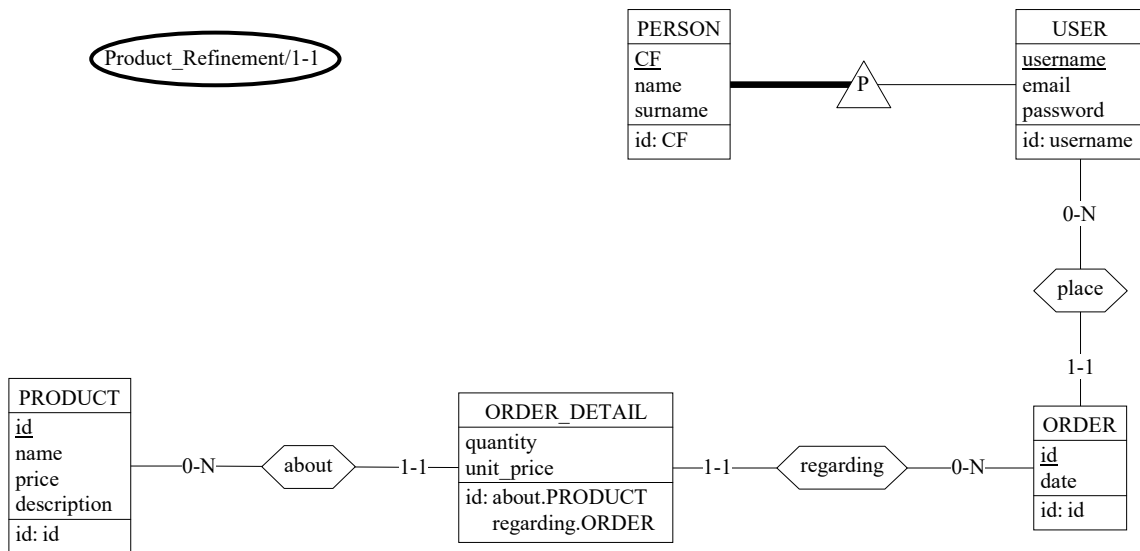


Figura 2.4: Raffinamento prodotti e ordini

2.3 Schema concettuale finale

Qui di seguito, è presente lo schema concettuale finale con tutti i raffinamenti.

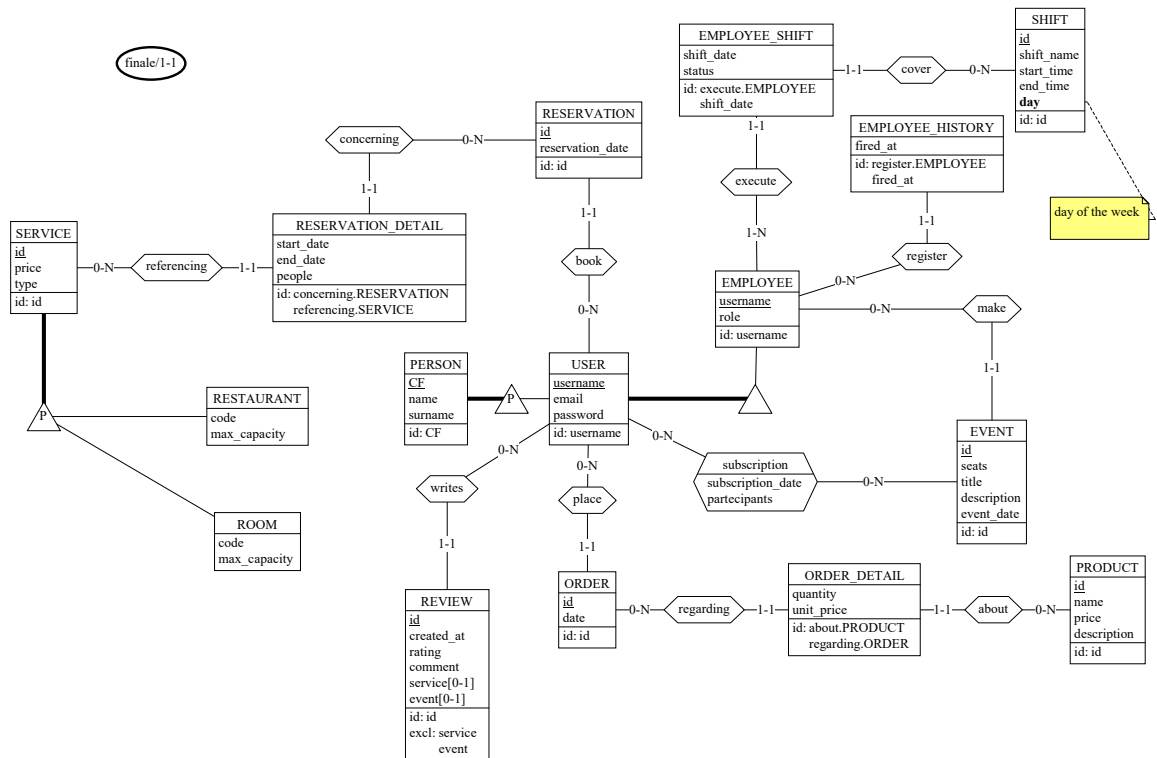


Figura 2.5: Schema ER, schema concettuale finale

Capitolo 3

Progettazione Logica

3.1 Stima del volume di dati

Sulla base dei requisiti e dell'analisi di dominio, abbiamo stimato l'ordine di grandezza dei dati iniziali e la crescita annua attesa per ciascuna entità principale. La Tabella 3.1 riassume i volumi e la crescita previsti; i valori sono indicativi e utili per dimensionare indici, partizionamento e politiche di archiviazione.

Tabella	Volume stimato	E/A
ORDER	15 000	E
RESERVATION	8 000	E
PERSON	6 000	E
USER	5 000	E
REVIEW	3 000	E
PRODUCT	500	E
EVENT	150	E
SERVICE	100	E
EMPLOYEE	50	E
ROOM	30	E
SHIFT	20	E
RESTAURANT	5	E
ORDER_DETAIL	45 000	A
EMPLOYEE_SHIFT	18 000	A
RESERVATION_DETAIL	12 000	A
SUBSCRIPTION	2 500	A
EMPLOYEE_HISTORY	80	A

Tabella 3.1: Stima volumi e classificazione Entità (E) o Associazione (A)

3.2 Descrizione operazioni

3.3 Analisi delle operazioni

3.4 Analisi delle ridondanze

3.5 Riepilogo Operazioni

3.6 Raffinamento dello schema

In questa sezione vengono illustrati i passaggi di raffinamento dello schema ER necessari per la sua traduzione nel modello relazionale. Il processo comprende la rimozione di attributi multivalore, la gestione delle gerarchie tramite generalizzazione/specializzazione, la reificazione delle associazioni molti-a-molti e la definizione degli identificatori principali per ciascuna entità. Questi interventi garantiscono la coerenza, la semplicità e l'efficienza dello schema relazionale risultante.

3.6.1 Rimozione gerarchie

Nel nostro schema ER sono presenti alcune gerarchie (generalizzazioni) che richiedono una traduzione appropriata nel modello relazionale. Di seguito analizziamo ciascun caso specifico e le relative scelte implementative.

Gerarchia di Servizio

La gerarchia tra SERVICE e le sottoclassi ROOM e RESTAURANT è totale ed esclusiva: ogni servizio è o una camera o un ristorante. SERVICE contiene gli attributi comuni, mentre ROOM e RESTAURANT sono tabelle specializzate collegate tramite chiave esterna. Il campo *type* in SERVICE identifica il tipo di servizio. Il modello è facilmente estendibile aggiungendo nuove tabelle specializzate per altri tipi di servizi.

Gerarchia di Persona

La gerarchia tra PERSON, USER ed EMPLOYEE è stata gestita con un approccio misto. La relazione tra PERSON e USER è stata collassata verso l'alto: ogni USER corrisponde necessariamente a una PERSONA, consentendo così di mantenere un'anagrafica centralizzata e priva di duplicati. Per quanto riguarda il passaggio da USER a EMPLOYEE, si è preferito sostituire la specializzazione con un'associazione: non tutti gli USER sono EMPLOYEE, ma solo quelli che ricoprono effettivamente un ruolo nel personale. Questa soluzione garantisce una chiara distinzione tra identità anagrafica e ruolo operativo, semplificando la gestione dei dati e delle funzionalità specifiche all'interno del sistema.

3.6.2 Scelta degli identificatori principali

Per ogni entità sono stati scelti identificatori che garantiscono univocità e stabilità nel tempo. La selezione degli identificatori è stata effettuata in modo da facilitare la gestione delle relazioni, assicurare la coerenza dei dati e supportare eventuali evoluzioni future dello schema.

Identificatori naturali

Gli identificatori naturali vengono utilizzati nelle tabelle in cui esiste un attributo intrinsecamente univoco e stabile nel tempo. In particolare, nella tabella **PERSON** si adotta il codice fiscale (**cf**) come chiave primaria naturale, garantendo l'unicità dell'anagrafica. Per la tabella **USER**, lo username rappresenta l'identificatore naturale, assicurando che ogni utente sia distinto in modo univoco. Analogamente, la tabella **EMPLOYEE** eredita lo username come chiave primaria, mantenendo la coerenza tra le entità correlate.

Identificatori artificiali

Gli identificatori artificiali vengono introdotti quando non è presente un attributo naturale sufficientemente stabile o univoco, oppure per semplificare la gestione delle relazioni e delle chiavi esterne. In questi casi si utilizza tipicamente un campo numerico auto-incrementale (**id**) come chiave primaria. Questo approccio è adottato per entità come **SERVICE**, **ORDER**, **RESERVATION**, **REVIEW**, **SHIFT**, **PRODUCT** ed **EVENT**, dove non esiste un attributo intrinseco che garantisca l'unicità e la stabilità nel tempo.

Identificatori composti

Per le entità derivate da reificazioni, sono stati adottati identificatori composti secondo la notazione dello schema logico:

- **ORDER_DETAIL**(order, product)
- **RESERVATION_DETAIL**(reservation, service)
- **EVENT_SUBSCRIPTION**(event, user)
- **EMPLOYEE_SHIFT**(employee, shift, shift_date)

3.6.3 Scelte progettuali significative

Flessibilità del catalogo servizi

La scelta di non collassare la gerarchia di **SERVICE** è significativa dal punto di vista progettuale. Questo approccio consente:

- **Espandibilità**: nuovi tipi di servizi possono essere aggiunti semplicemente creando nuove tabelle specializzate.
- **Separazione delle responsabilità**: ogni tipologia di servizio mantiene i propri attributi specifici.
- **Efficienza delle query**: il campo **type** in **SERVICE** permette filtri rapidi senza necessità di join aggiuntivi.
- **Integrità referenziale**: le prenotazioni referenziano sempre la tabella **SERVICE**, garantendo coerenza anche in presenza di nuovi servizi.

Gestione storica del personale

La tabella `EMPLOYEE_HISTORY` consente di verificare se un utente è un ex dipendente: se lo username è presente in `EMPLOYEE` l'utente è dipendente attivo, se compare anche in `EMPLOYEE_HISTORY` significa che non lavora più nell'azienda. L'intreccio tra le due tabelle permette di distinguere tra dipendenti attuali ed ex dipendenti, soddisfacendo il requisito di audit trail.

Separazione identità e autenticazione

La separazione tra `PERSON` e `USER` garantisce che i dati anagrafici siano gestiti indipendentemente dalle credenziali di accesso. Questo approccio migliora la sicurezza, evita duplicazioni e semplifica la manutenzione delle informazioni personali e di autenticazione.

3.7 Schema relazionale finale

Dopo aver applicato tutti i raffinamenti, lo schema relazionale finale è rappresentato dalle seguenti tabelle.

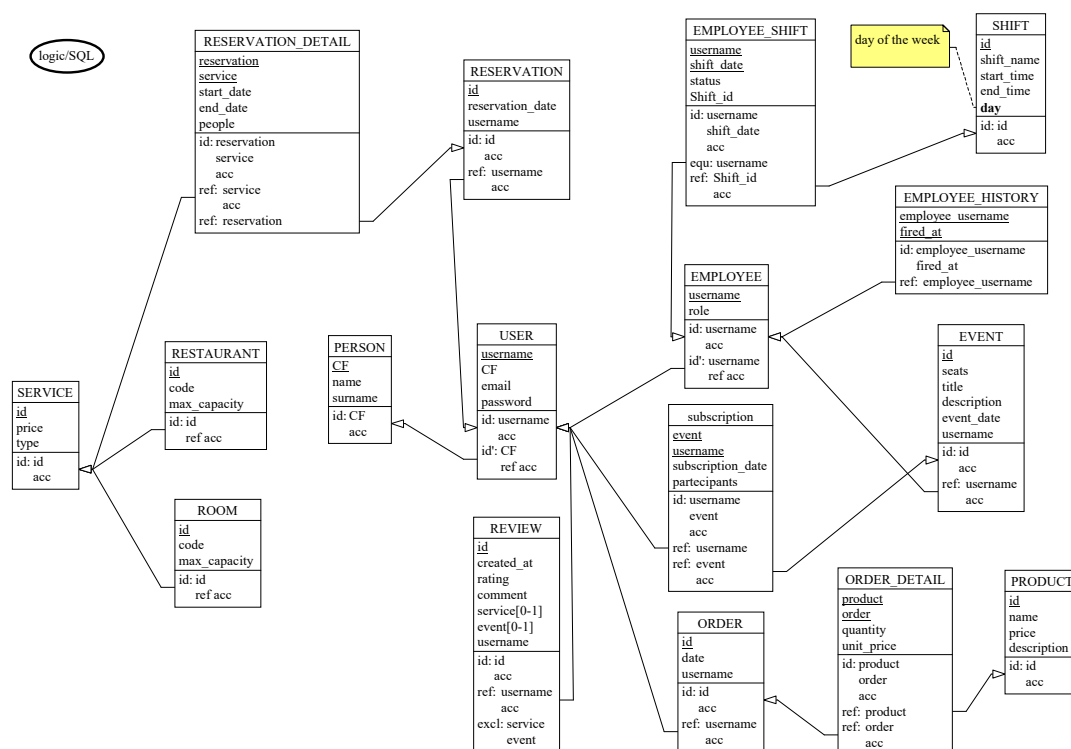


Figura 3.1: Schema relazionale finale

Capitolo 4

Progettazione della Base di Dati

Una volta creato il nostro database, riportiamo di seguito una parte del codice relazionale utilizzato per la sua implementazione.

4.1 Check

Sono stati utilizzati vincoli di tipo **CHECK** per definire alcuni domini e assicurare semplici proprietà degli attributi. Di seguito un esempio di vincolo **CHECK** utilizzato per assicurare che il prezzo di ogni prodotto sia maggiore zero:

```
1 CREATE TABLE PRODUCT (  
2     id INT AUTO_INCREMENT PRIMARY KEY,  
3     name VARCHAR(100) NOT NULL,  
4     description TEXT NOT NULL,  
5     price DECIMAL(8,2) NOT NULL CHECK (price > 0)  
6 );
```

4.2 Viste

La seguente vista `active_employees` restituisce l'elenco dei dipendenti attivi, mostrando per ciascuno username, email, nome, cognome e ruolo. Un dipendente è considerato attivo se il suo username non compare nella tabella `EMPLOYEE_HISTORY`, che traccia lo storico delle variazioni di stato.

```
1 CREATE VIEW active_employees AS  
2 SELECT  
3     e.username,  
4     u.email,  
5     p.name,  
6     p.surname,  
7     e.role  
8 FROM EMPLOYEE e  
9 JOIN USER u ON e.username = u.username  
10 JOIN PERSON p ON u.cf = p.cf  
11 WHERE e.username NOT IN (  
12     SELECT username FROM EMPLOYEE_HISTORY  
13 );
```

4.3 Trigger

Esempio di trigger per vincolare le recensioni: impedisce di recensire sia evento che servizio insieme, e consente la recensione solo se l'utente ha partecipato all'evento (già svolto) o ha usufruito del servizio.

```
1 DROP TRIGGER IF EXISTS trg_review_before_insert;
2 DELIMITER $$
3 CREATE TRIGGER trg_review_before_insert
4 BEFORE INSERT ON REVIEW
5 FOR EACH ROW
6 BEGIN
7     DECLARE cnt INT DEFAULT 0;
8
9     IF (NEW.event IS NOT NULL AND NEW.service IS NOT NULL) OR (NEW.
event IS NULL AND NEW.service IS NULL) THEN
10         SIGNAL SQLSTATE '45000'
11         SET MESSAGE_TEXT = 'Set either event or service (not both)
for the review.';
12     END IF;
13
14     IF NEW.event IS NOT NULL THEN
15         SELECT COUNT(*)
16         INTO cnt
17         FROM EVENT e
18         JOIN EVENT_SUBSCRIPTION es
19         ON es.event = e.id
20         AND es.'user' = NEW.'user'
21         WHERE e.id = NEW.event
22         AND e.event_date < CURDATE();
23
24     IF cnt = 0 THEN
25         SIGNAL SQLSTATE '45000'
26         SET MESSAGE_TEXT = 'You can review the event only if
you were subscribed and the event date is in the past.';
27     END IF;
28     END IF;
29
30     IF NEW.service IS NOT NULL THEN
31         SELECT COUNT(*)
32         INTO cnt
33         FROM RESERVATION r
34         JOIN RESERVATION_DETAIL rd
35         ON rd.reservation = r.id
36         AND rd.service = NEW.service
37         WHERE r.username = NEW.'user'
38         AND rd.end_date < NOW();
39
40     IF cnt = 0 THEN
41         SIGNAL SQLSTATE '45000'
42         SET MESSAGE_TEXT = 'You can review the service only
after you have used it (completed reservation).';
43     END IF;
44     END IF;
45 END$$
46 DELIMITER ;
```


4.4 Traduzione delle operazioni

Vengono presentate le query SQL che implementano le principali operazioni del sistema agriturismo. Le query sono state progettate per essere efficienti e sfruttare gli indici e i vincoli definiti nello schema. Di seguito sono riportate le principali operazioni raggruppate per area funzionale: statistiche, prenotazioni, recensioni, iscrizioni eventi, autenticazione e gestione ordini.

4.4.1 Visualizzazione statistiche dashboard

Le seguenti query sono utilizzate per popolare la dashboard amministrativa con le metriche principali del sistema.

Top servizi per prenotazioni

Analizza le prenotazioni per identificare i servizi più richiesti, distinguendo tra ristoranti e camere attraverso un'articolata procedura di join e raggruppamento.

```
1 SELECT CASE
2     WHEN s.type = 'RESTAURANT' THEN CONCAT('Restaurant - ', r.code)
3     WHEN s.type = 'ROOM' THEN CONCAT('Room - ', ro.code)
4     ELSE s.type
5 END AS service_name,
6 COUNT(rd.service) AS booking_count
7 FROM SERVICE AS s
8 LEFT JOIN RESTAURANT AS r ON s.id = r.service
9 LEFT JOIN ROOM AS ro ON s.id = ro.service
10 JOIN RESERVATION_DETAIL AS rd ON s.id = rd.service
11 GROUP BY s.id, s.type, r.code, ro.code
12 ORDER BY booking_count DESC
13 LIMIT 5;
```

Top prodotti per quantità venduta

Prodotti più venduti in base alla quantità totale ordinata.

```
1 SELECT
2     p.name AS product_name,
3     SUM(od.quantity) AS total_quantity
4 FROM PRODUCT AS p
5 JOIN ORDER_DETAIL AS od ON p.id = od.product
6 GROUP BY p.id, p.name
7 ORDER BY total_quantity DESC
8 LIMIT 5;
```

Top eventi per partecipanti

Determina gli eventi con il maggior numero di partecipanti totali.

```
1 SELECT
2   e.title AS event_title,
3   e.event_date,
4   SUM(es.participants) AS total_participants
5 FROM EVENT AS e
6 JOIN EVENT_SUBSCRIPTION AS es ON e.id = es.event
7 GROUP BY e.id, e.title, e.event_date
8 ORDER BY total_participants DESC
9 LIMIT 5;
```

Top prodotti per fatturato

Calcola i 5 prodotti che generano il maggior fatturato.

```
1 SELECT
2   p.name AS product_name,
3   SUM(od.quantity * od.unit_price) AS total_revenue
4 FROM PRODUCT AS p
5 JOIN ORDER_DETAIL AS od ON p.id = od.product
6 GROUP BY p.id, p.name
7 ORDER BY total_revenue DESC
8 LIMIT 5;
```

Fatturato totale

Calcolo del ricavo complessivo generato dalle vendite dei prodotti.

```
1 SELECT ROUND(SUM(od.quantity * od.unit_price), 2) AS
2   overall_total_revenue
3 FROM ORDER_DETAIL od;
```

Statistiche generali del sistema

Query composita che fornisce un riepilogo completo delle metriche di sistema tramite sottoselezioni multiple, aggregando dati da diverse tabelle per offrire una visione d'insieme immediata.

```
1 SELECT
2   (SELECT COUNT(*) FROM USER) AS total_customers,
3   (SELECT COUNT(*) FROM EMPLOYEE) AS total_employees,
4   (SELECT COUNT(*) FROM ORDERS) AS total_orders,
5   (SELECT ROUND(SUM(od.quantity * od.unit_price), 2) FROM ORDER_DETAIL
6    AS od) AS total_revenue,
7   (SELECT COUNT(*) FROM RESERVATION) AS total_reservations;
```

4.4.2 Prenotazione servizi

Le principali query per la prenotazione di servizi, come camere e tavoli al ristorante, includono la verifica della disponibilità, la creazione della prenotazione e la gestione dei dettagli associati, garantendo il rispetto dei vincoli di capacità e delle regole temporali definite dal sistema.

Verifica disponibilità camere

La disponibilità delle camere viene verificata analizzando le prenotazioni esistenti e selezionando solo quelle con capacità sufficiente e libere nel periodo richiesto. Il controllo si basa sulla non sovrapposizione temporale tra le prenotazioni già registrate e l'intervallo desiderato, così da garantire che la camera sia effettivamente disponibile.

```
1 SELECT
2     ro.code AS room,
3     s.price AS price,
4     ro.max_capacity
5 FROM ROOM AS ro
6 JOIN SERVICE AS s
7     ON s.id = ro.service
8 WHERE
9     ro.max_capacity >= @n_people
10    AND ro.service NOT IN (
11        SELECT rd.service
12        FROM RESERVATION_DETAIL AS rd
13        WHERE NOT (rd.end_date <= @start_date OR rd.start_date >= @end_date
14    );
```

Verifica disponibilità tavoli

Calcola i posti disponibili considerando le prenotazioni esistenti che si sovrappongono all'intervallo richiesto, utilizzando un left join condizionato e una clausola HAVING per filtrare i ristoranti con posti sufficienti.

```
1 SELECT
2     r.code AS restaurant,
3     s.price AS price,
4     r.max_capacity,
5     (r.max_capacity - IFNULL(SUM(rd.people), 0)) AS available_seats
6 FROM RESTAURANT AS r
7 JOIN SERVICE AS s ON s.id = r.service
8 LEFT JOIN RESERVATION_DETAIL AS rd ON rd.service = r.service
9     AND NOT (rd.end_date <= @start_date OR rd.start_date >= @end_date)
10 GROUP BY r.service, r.code, s.price, r.max_capacity
11 HAVING available_seats >= @n_people;
```

4.4.3 Gestione recensioni

Gli utenti possono recensire solo eventi conclusi a cui hanno partecipato o servizi già prenotati e utilizzati, garantendo valutazioni autentiche.

Inserimento recensione evento

Questa query consente di inserire una recensione per un evento solo se l'utente è iscritto, l'evento si è concluso e non esiste già una recensione per quell'evento da parte dello stesso utente. In questo modo si garantisce la correttezza e l'integrità dei dati.

```
1 INSERT INTO REVIEW (user, event, rating, comment)
2 SELECT
3     'mrossi' AS user,
4     e.id AS event,
5     5 AS rating,
6     'Amazing experience! Will definitely come again.' AS comment
7 FROM EVENT AS e
8 INNER JOIN EVENT_SUBSCRIPTION AS es ON e.id = es.event
9     AND es.user = 'mrossi'
10 WHERE
11     e.title = 'Farm Open Day'
12     AND e.event_date < CURDATE()
13     AND NOT EXISTS (
14         SELECT 1
15         FROM REVIEW AS r
16         WHERE r.user = 'mrossi' AND r.event = e.id
17     )
18 LIMIT 1;
```

Inserimento recensione servizio

Consente di inserire una recensione su un servizio solo se l'utente ha completato una prenotazione per quel servizio e non esiste già una recensione associata, garantendo così la correttezza referenziale ed evitando duplicati.

```
1 INSERT INTO REVIEW (user, service, rating, comment)
2 SELECT
3     'aneri' AS user,
4     s.id AS service,
5     4 AS rating,
6     'Good service and friendly staff.' AS comment
7 FROM SERVICE AS s
8 INNER JOIN RESERVATION_DETAIL AS rd ON s.id = rd.service
9 INNER JOIN RESERVATION AS r ON rd.reservation = r.id
10 WHERE
11     r.username = 'aneri'
12     AND rd.end_date < NOW()
13     AND s.type = 'RESTAURANT'
14     AND NOT EXISTS (
15         SELECT 1
16         FROM REVIEW AS rev
17         WHERE rev.user = 'aneri' AND rev.service = s.id
18     )
19 LIMIT 1;
```

Aggiornamento recensione

Modifica il voto e il commento di una recensione esistente.

```
1 UPDATE REVIEW
2 SET
3     rating = 4,
4     comment = 'Very good event, but could use more activities. Overall
5         enjoyed it!',
6     created_at = NOW()
7 WHERE
8     user = 'mrossi'
9     AND event = (
10        SELECT id
11        FROM EVENT
12        WHERE title = 'Farm Open Day'
13    )
14    AND id IS NOT NULL;
```

4.4.4 Gestione iscrizioni eventi

Le operazioni sulle iscrizioni agli eventi includono la registrazione di nuovi partecipanti, l'aggiornamento del numero di iscritti e la cancellazione delle iscrizioni. Il sistema garantisce che il numero totale di partecipanti non superi la capienza dell'evento e consente agli utenti di modificare o annullare la propria iscrizione fino all'inizio dell'evento.

Update e Iscrizione utente a evento

Permette a un utente di iscriversi a un evento specificando il numero di partecipanti. Il meccanismo "ON DUPLICATE KEY UPDATE" trasforma l'insert in un update che modifica solo il numero di partecipanti, nel caso in cui l'utente abbia già effettuato un'iscrizione al dato evento.

```
1 INSERT INTO EVENT_SUBSCRIPTION (event, user, participants)
2 SELECT
3     e.id,
4     u.username,
5     4
6 FROM EVENT AS e
7 CROSS JOIN USER AS u
8 WHERE e.title = 'Harvest Festival' AND u.username = 'lblu'
9 ON DUPLICATE KEY UPDATE participants = 4;
```

4.4.5 Esecuzione prenotazioni confermate

Le prenotazioni di servizi (camere e ristoranti) vengono gestite tramite query che verificano la disponibilità, creano la prenotazione principale e aggiungono i dettagli relativi al servizio scelto. Il sistema assicura che non vi siano sovrapposizioni e che la capacità sia rispettata, garantendo integrità e correttezza dei dati.

Creazione prenotazione principale

Crea il record principale di prenotazione per un utente, con un controllo che evita la creazione di prenotazioni duplicate nella stessa giornata.

```
1 INSERT INTO RESERVATION (username, reservation_date)
2 SELECT
3     'gverdi',
4     NOW()
5 WHERE NOT EXISTS (
6     SELECT 1
7     FROM RESERVATION
8     WHERE
9         username = 'gverdi'
10    AND DATE(reservation_date) = CURDATE()
11 );
```

Prenotazione tavolo ristorante

Prenotazione di tavoli al ristorante, con join sulla tabella RESTAURANT per identificare correttamente il servizio.

```
1 SET @new_reservation_id = LAST_INSERT_ID();
2
3 INSERT INTO RESERVATION_DETAIL (reservation, service, start_date,
4     end_date, people)
5 SELECT
6     @new_reservation_id AS reservation,
7     s.id AS service,
8     '2024-01-25 19:00:00' AS start_date,
9     '2024-01-25 21:00:00' AS end_date,
10    2 AS people
11 FROM SERVICE AS s
12 INNER JOIN RESTAURANT AS r ON s.id = r.service
13 WHERE r.code = 'T01'
14 LIMIT 1;
```

Prenotazione camera con controllo duplicati

Gestione della prenotazione di una camera, con creazione della prenotazione principale e dei dettagli. Sono previsti controlli per evitare duplicati e viene gestito correttamente l'ID generato per la prenotazione.

```
1 INSERT INTO RESERVATION (username, reservation_date)
2 SELECT 'fbianchi', DATE_ADD(NOW(), INTERVAL 1 HOUR)
3 WHERE NOT EXISTS (
4     SELECT 1 FROM RESERVATION
5     WHERE username = 'fbianchi'
6     AND DATE(reservation_date) = CURDATE()
7 );
8
9 SET @room_reservation_id = LAST_INSERT_ID();
10
11 INSERT INTO RESERVATION_DETAIL (reservation, service, start_date,
12     end_date, people)
13 SELECT
14     @room_reservation_id as reservation,
15     s.id as service,
16     '2024-01-26 15:00:00' as start_date,
17     '2024-01-28 11:00:00' as end_date,
18     2 as people
19 FROM SERVICE s
20 INNER JOIN ROOM r ON s.id = r.service
21 WHERE r.code = 'R03'
22 LIMIT 1;
```

4.4.6 Eliminazione prenotazioni e iscrizioni

Le operazioni di eliminazione permettono agli utenti e agli amministratori di rimuovere prenotazioni di servizi e iscrizioni agli eventi, garantendo il rispetto dei vincoli temporali e di integrità referenziale. È possibile cancellare una prenotazione solo se non è già iniziata, mentre le iscrizioni agli eventi possono essere annullate fino all'inizio dell'evento. Queste funzionalità assicurano una gestione sicura e corretta delle risorse e delle partecipazioni.

Eliminazione prenotazione servizio

Permette di cancellare una prenotazione di servizio esistente, rimuovendo prima i dettagli della prenotazione per rispettare i vincoli di integrità referenziale e successivamente il record principale della prenotazione.

```
1 DELETE FROM RESERVATION_DETAIL
2 WHERE reservation = @reservation_id;
3
4 DELETE FROM RESERVATION
5 WHERE id = @reservation_id
6 AND username = @username;
```

Eliminazione iscrizione evento

Rimuove l'iscrizione di un utente a un evento specifico, verificando che l'iscrizione esista e che l'evento non sia già iniziato; in questo modo è consentita la cancellazione solo per eventi futuri.

```
1 DELETE FROM EVENT_SUBSCRIPTION
2 WHERE
3     user = @username
4     AND event = @event_id
5     AND EXISTS (
6         SELECT 1
7         FROM EVENT AS e
8         WHERE
9             e.id = @event_id
10            AND e.event_date > CURDATE()
11    );
```

Eliminazione prenotazione con controllo temporale

L'eliminazione di una prenotazione è consentita solo se non è già iniziata; viene implementato un controllo temporale che previene la cancellazione di prenotazioni in corso o concluse.

```
1 DELETE FROM RESERVATION
2 WHERE
3     id = @reservation_id
4     AND username = @username
5     AND NOT EXISTS (
6         SELECT 1
7         FROM RESERVATION_DETAIL AS rd
8         WHERE
9             rd.reservation = @reservation_id
10            AND rd.start_date <= NOW()
11    );
```


4.4.7 Autenticazione utente

L'autenticazione utente consente l'accesso sicuro alla piattaforma tramite verifica di username e password. Per distinguere tra clienti e dipendenti, viene utilizzata la vista `active_employees`: se il campo `role` restituito dalla vista è `NULL`, l'utente è considerato un cliente normale; se invece è valorizzato, l'utente è un dipendente attivo e il ruolo viene mostrato.

Verifica credenziali di login

La query seguente verifica le credenziali di login e determina il profilo utente, sfruttando la vista `active_employees` per identificare i dipendenti attivi.

```
1 SELECT
2     u.username ,
3     u.email ,
4     p.name ,
5     p.surname ,
6     CASE WHEN ae.role IS NOT NULL THEN 'employee' ELSE 'customer' END AS
       user_type ,
7     ae.role AS employee_role
8 FROM USER AS u
9 JOIN PERSON AS p ON u.cf = p.cf
10 LEFT JOIN active_employees AS ae ON u.username = ae.username
11 WHERE u.username = 'mrossi' OR u.email = 'mrossi@farm.com';
```

4.4.8 Gestione ordini prodotti

Gli ordini prodotti vengono gestiti tramite una transazione che crea l'ordine principale e inserisce i dettagli dei prodotti selezionati, con quantità e prezzo corrente. Il sistema assicura che ogni ordine sia associato all'utente e che i dati siano registrati in modo consistente.

Creazione nuovo ordine

La seguente procedura crea un nuovo ordine, recupera automaticamente l'ID generato e inserisce i prodotti con i prezzi correnti, gestendo tutta la logica di ordine in un'unica sequenza di operazioni.

```
1 SET @new_order_id = LAST_INSERT_ID();
2
3 INSERT INTO ORDER_DETAIL (order, product, quantity, unit_price)
4 SELECT
5     @new_order_id AS order_id,
6     p.id AS product_id,
7     3 AS quantity,
8     p.price AS unit_price
9 FROM PRODUCT AS p
10 WHERE p.name = 'Farm Eggs (12 pcs)'
11 LIMIT 1;
12
13 INSERT INTO ORDER_DETAIL (order, product, quantity, unit_price)
14 SELECT
15     @new_order_id AS order_id,
16     p.id AS product_id,
17     2 AS quantity,
18     p.price AS unit_price
19 FROM PRODUCT AS p
20 WHERE p.name = 'Fresh Bread'
21 LIMIT 1;
22
23 INSERT INTO ORDER_DETAIL (order, product, quantity, unit_price)
24 SELECT
25     @new_order_id AS order_id,
26     p.id AS product_id,
27     1 AS quantity,
28     p.price AS unit_price
29 FROM PRODUCT AS p
30 WHERE p.name = 'Honey Jar (500g)'
31 LIMIT 1;
```

Capitolo 5

Progettazione dell'applicazione

L'applicazione è stata sviluppata con il framework **Django**, che gestisce routing, database e autenticazione in modo sicuro e scalabile.

5.1 Barra di Navigazione

La **barra di navigazione** permette un accesso rapido alle principali sezioni del sito, come prodotti, eventi, servizi, area personale e funzioni amministrative.

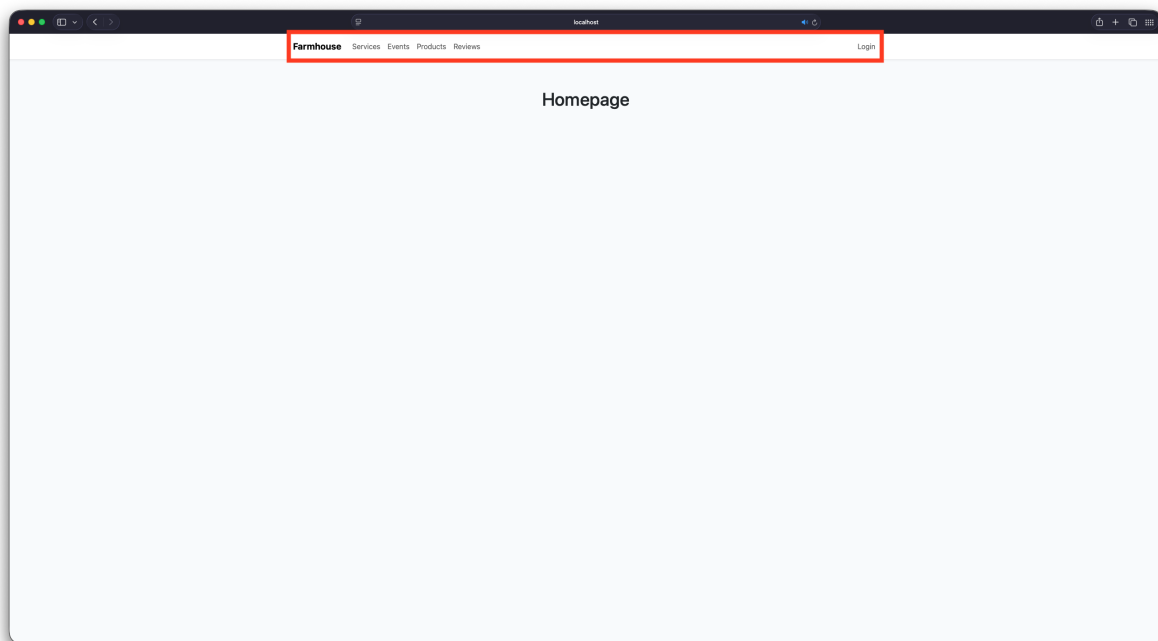
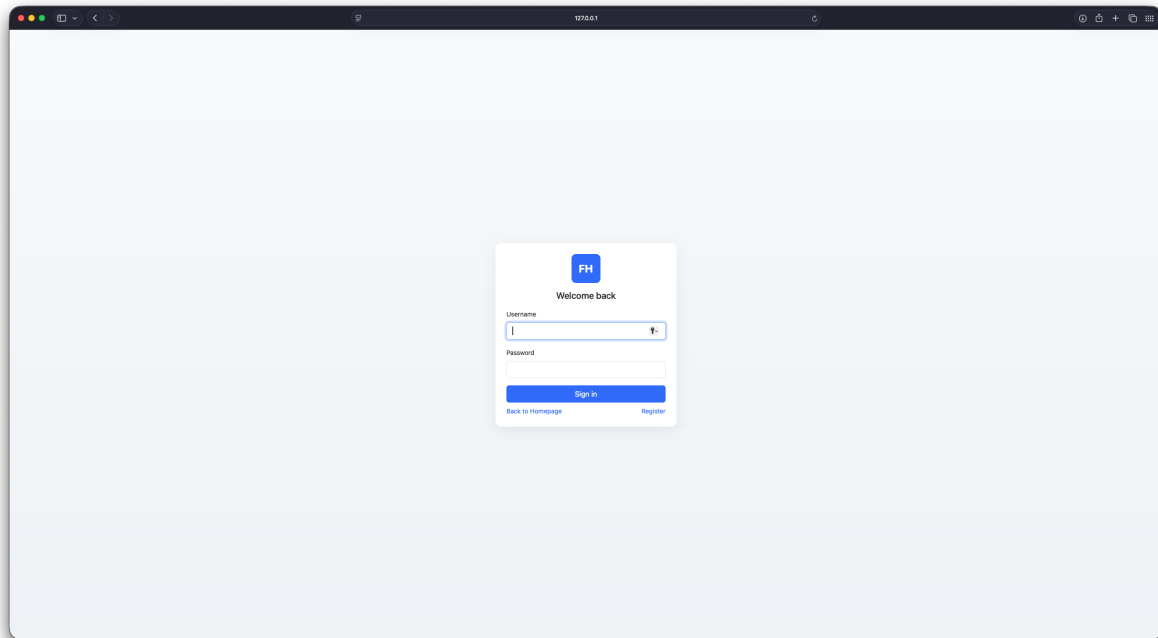


Figura 5.1: Barra di navigazione

Login

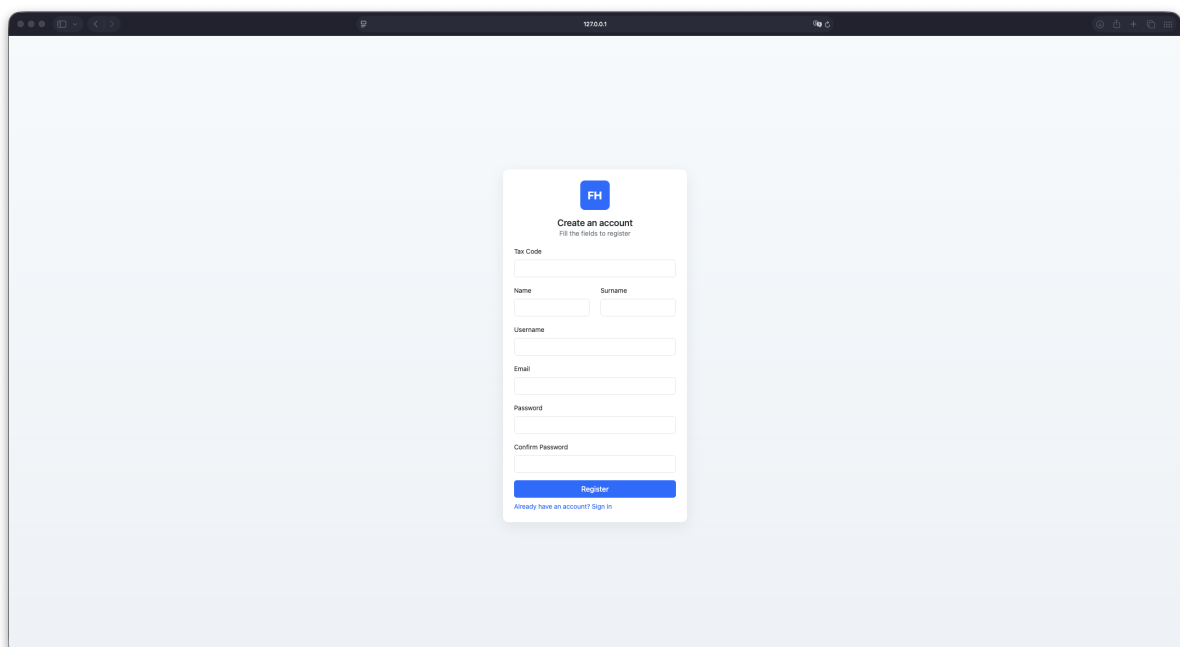
Il form di login consente agli utenti registrati di accedere rapidamente alla piattaforma inserendo username e password. Il sistema verifica le credenziali e, in caso di errore, mostra un messaggio di avviso.



The screenshot shows a web browser window with a light blue background. In the center, there is a white login form titled "Welcome back" with a blue "FH" logo above it. The form contains a "Username" field with the letter "l" entered, a "Password" field, and a blue "Sign in" button. Below the button are two links: "Back to Homepage" and "Register".

Registrazione

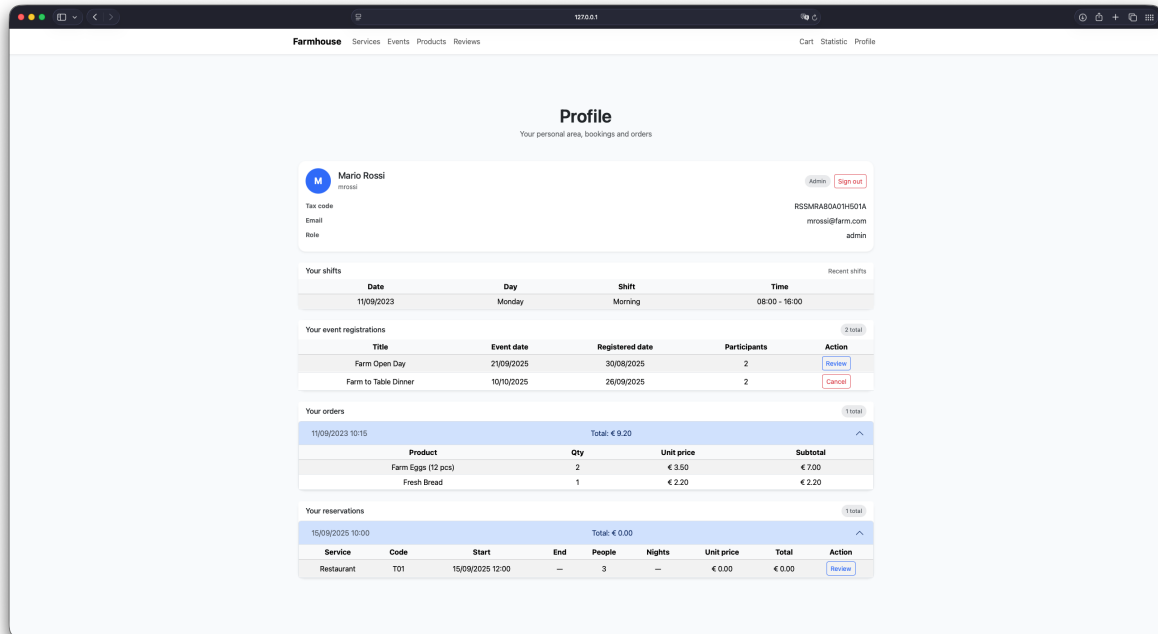
Anche per registrarsi è disponibile un form semplice e intuitivo, che permette agli utenti di creare un nuovo account inserendo i dati richiesti. Dopo la registrazione, l'utente potrà accedere a tutte le funzionalità della piattaforma.



The screenshot shows a web browser window with a light blue background. In the center, there is a white registration form titled "Create an account" with a blue "FH" logo above it. The form includes a subtitle "Fill the fields to register" and several input fields: "Tax Code", "Name", "Surname", "Username", "Email", "Password", and "Confirm Password". A blue "Register" button is at the bottom. Below the button is a link: "Already have an account? Sign in".

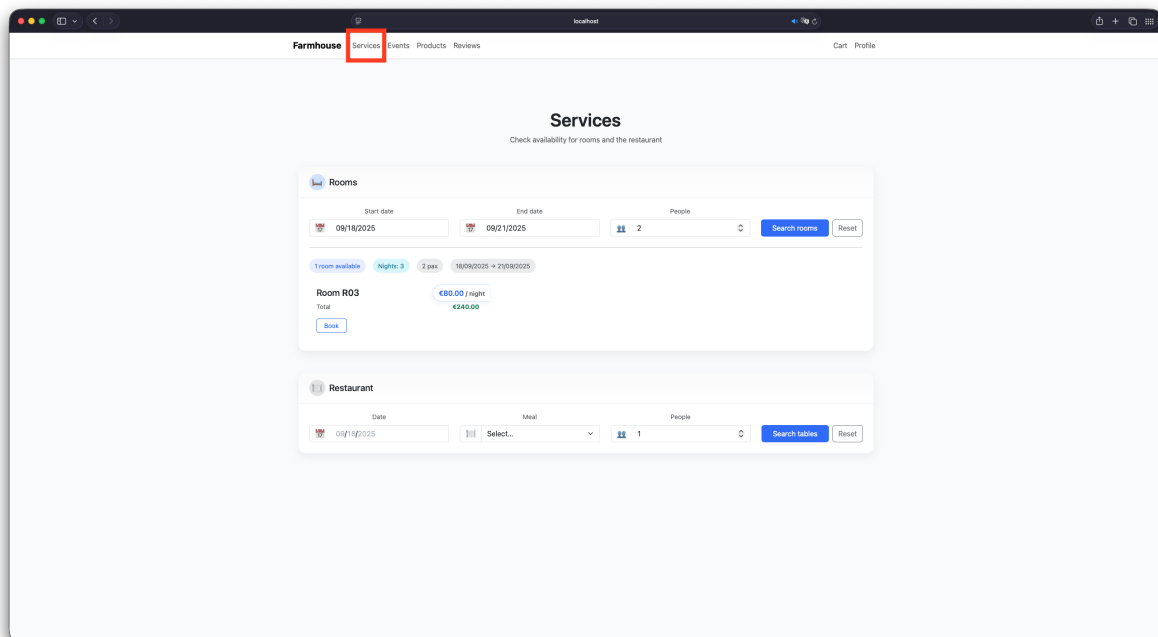
5.2 Interfaccia Utente

Dopo l'accesso, l'utente potrà visualizzare il profilo, con le prenotazioni e gli ordini, con la possibilità di recensire o annullare prenotazioni future.



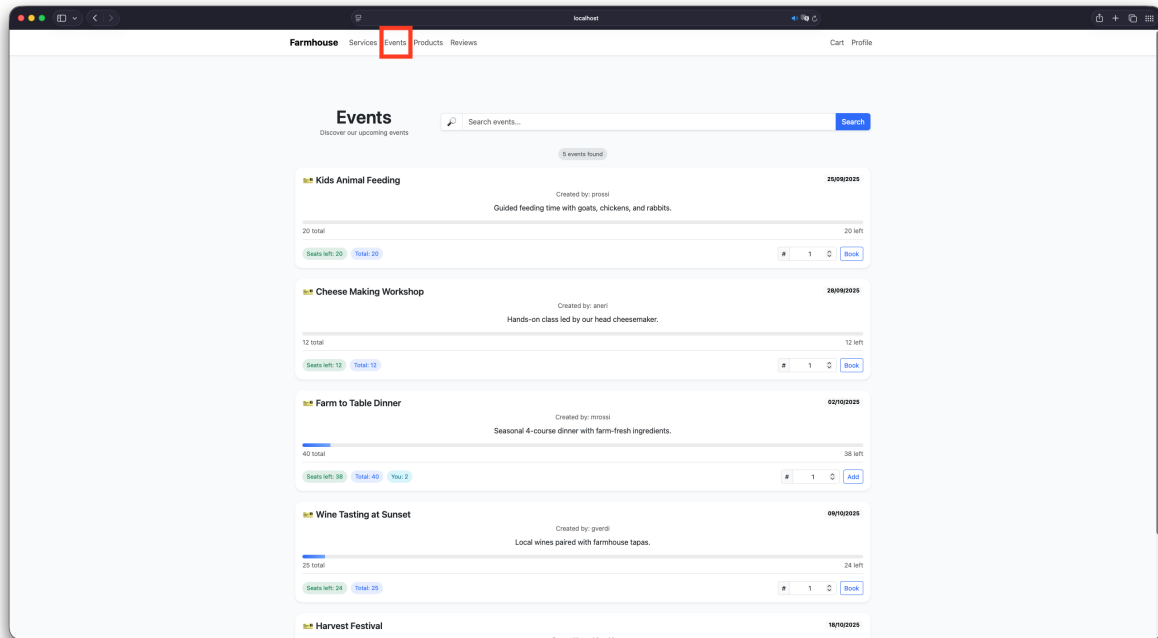
Servizi

Dopo aver scelto il servizio da prenotare, è sufficiente inserire i dati necessari; il sistema mostrerà la disponibilità aggiornata del servizio selezionato.



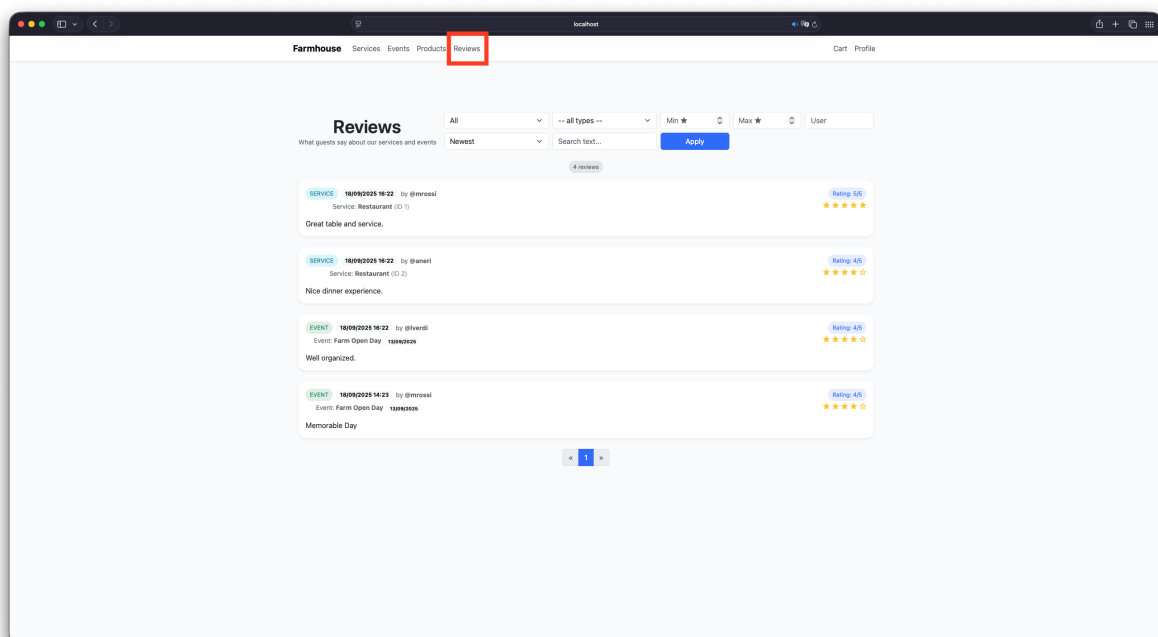
Eventi

Nella sezione eventi, viene mostrato l'elenco degli eventi disponibili. L'utente può selezionare l'evento di interesse, specificare il numero di partecipanti e procedere con la prenotazione.

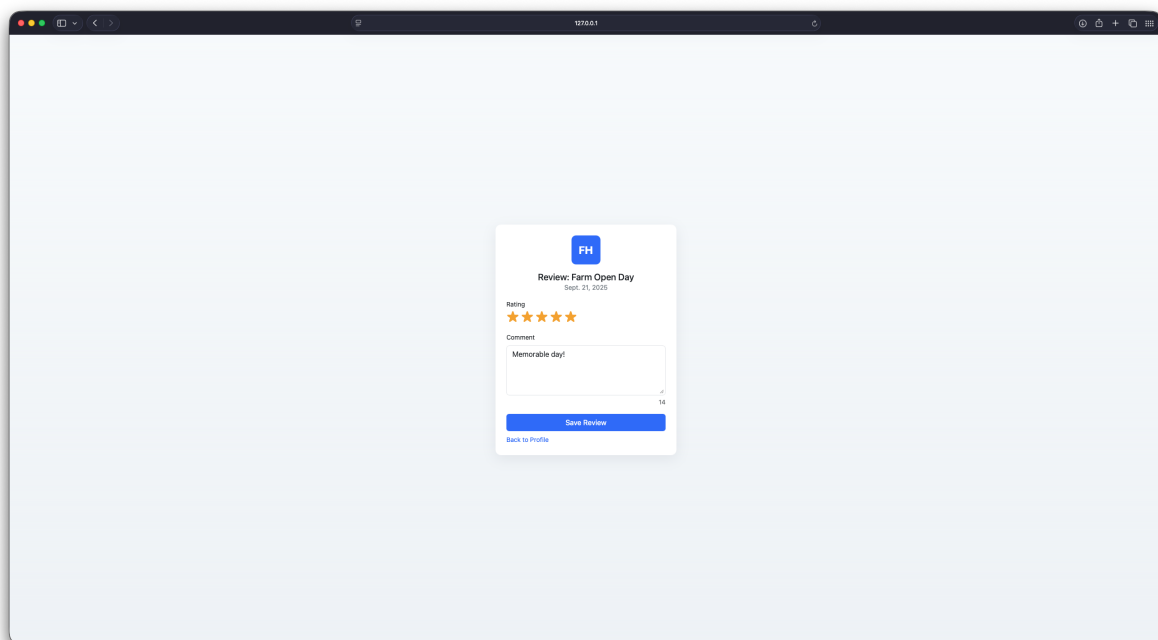


Recensioni

Gli utenti possono visualizzare tutte le recensioni e filtrarle per evento o servizio.

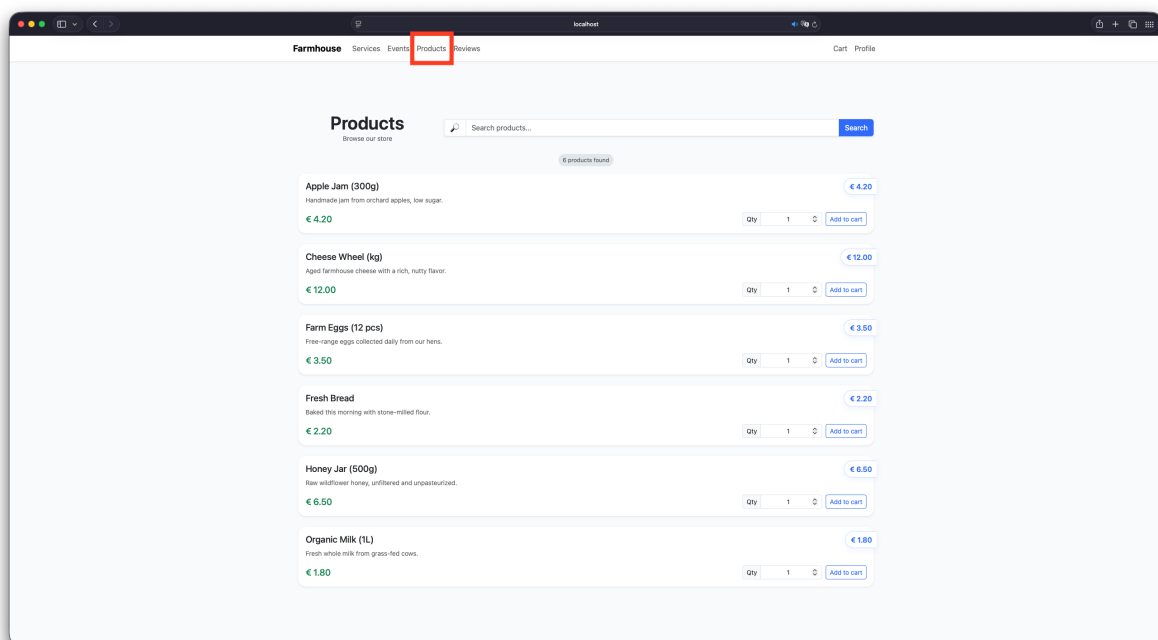


Il form permette agli utenti di lasciare una recensione su eventi o servizi a cui hanno partecipato, inserendo commento e voto. La recensione è consentita solo dopo la partecipazione effettiva.



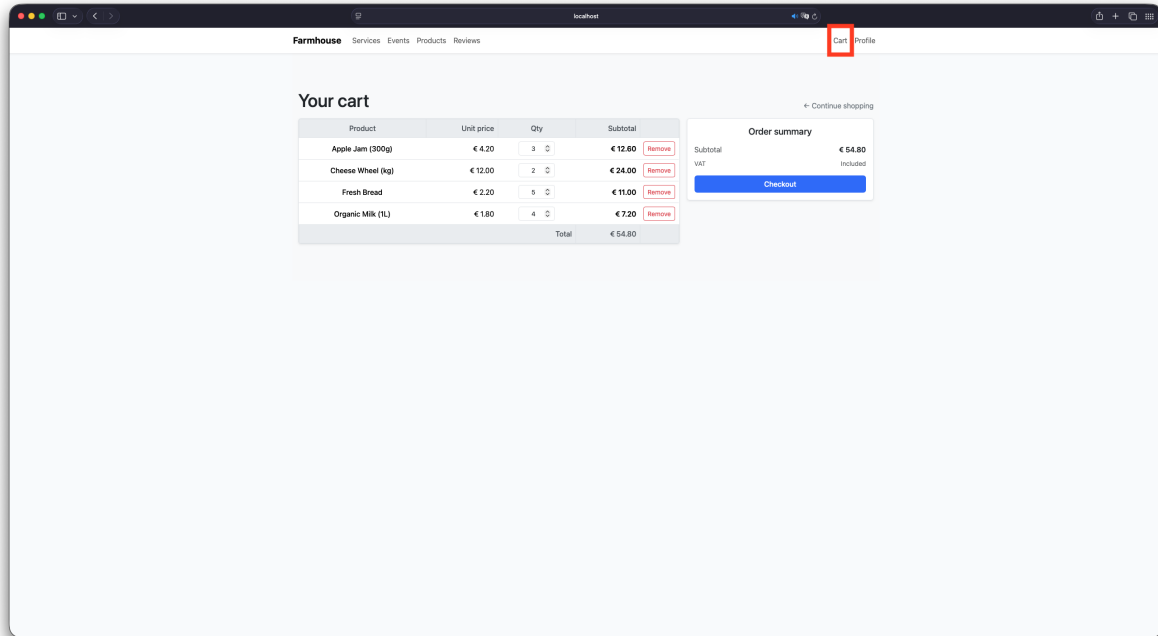
Prodotti

La sezione prodotti consente agli utenti di consultare il catalogo aggiungerli al carrello per l'acquisto. Il sistema mostra in tempo reale il contenuto del carrello e il totale dell'ordine. Fatto il checkout sarà visibile il riepilogo nella sezione profilo.



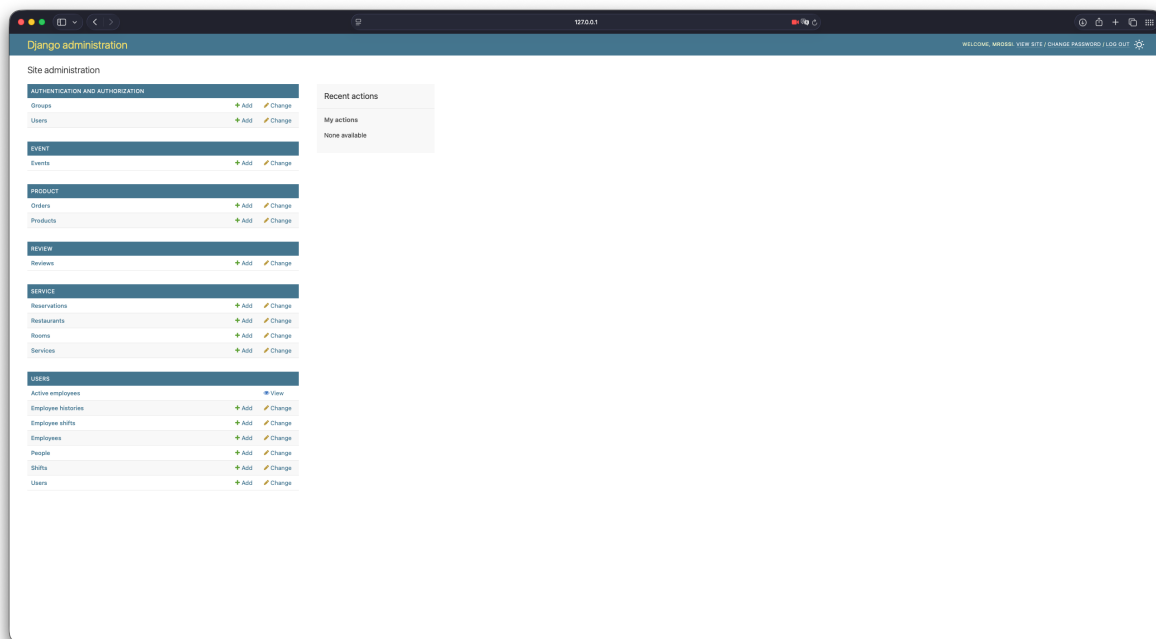
Carrello

Il **carrello** è una funzionalità applicativa che consente agli utenti di selezionare e gestire i prodotti da acquistare prima di confermare l'ordine. Il carrello non è rappresentato nel database, ma viene gestito lato applicazione: i prodotti selezionati vengono memorizzati temporaneamente fino al checkout, momento in cui viene creato l'ordine definitivo e registrato nel sistema.

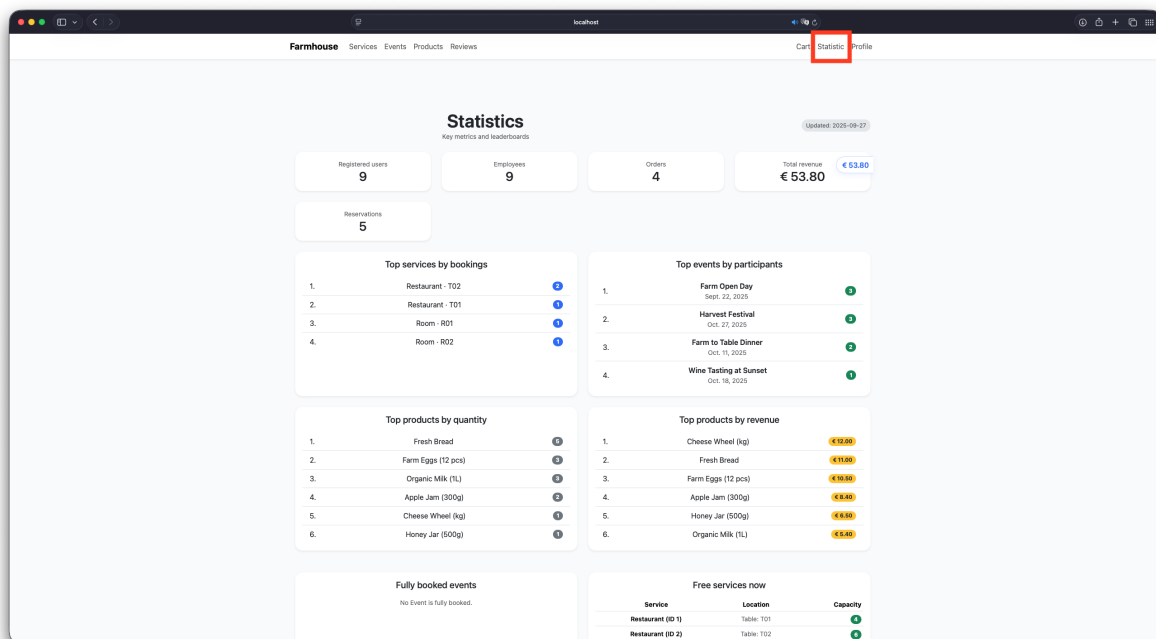


5.3 Interfaccia Amministratore

Per la gestione amministrativa, l'applicazione sfrutta la sezione **Django Admin**, che consente agli amministratori di accedere rapidamente a tutte le tabelle del database, modificare dati, tramite un'interfaccia web sicura e strutturata.



Oltre al pannello standard di Django Admin, è stata realizzata una pagina web dedicata alla visualizzazione delle statistiche principali del sistema, come l'andamento delle vendite, la partecipazione agli eventi e la presenza del personale. Questa pagina presenta tabelle riepilogative.



Appendice A

Guida Utente

A.1 Clonazione del repository

Clonare il progetto da GitHub e accedere alla cartella:

```
> git clone https://github.com/alessandrorebosio/D25-farmhouse.git
> cd DB25-farmhouse
```

A.2 Installazione delle dipendenze

Si consiglia di utilizzare un ambiente virtuale Python per isolare le dipendenze del progetto.

```
> python3 -m venv venv
```

Attivazione dell'ambiente virtuale

```
# Su Linux/macOS:
> source venv/bin/activate
# Su Windows:
> venv\Scripts\activate
```

Installazione delle dipendenze dal file requirements.txt

```
> pip install -r requirements.txt
```

A.3 Creazione del database

Per creare il database MySQL a partire dagli script SQL forniti, assicurarsi di avere MySQL installato e in esecuzione.

```
> mysql -u root -p < app/sql/db.sql
> mysql -u root -p < app/sql/demo.sql
```

Verrà richiesta la password dell'utente `root`. Il comando eseguirà tutte le istruzioni SQL contenute nel file `db.sql`, creando tabelle, vincoli e dati di esempio necessari per l'applicazione.

A.4 Avvio dell'applicazione

Per avviare l'applicazione Django, assicurarsi che l'ambiente virtuale sia attivo e che il database sia stato creato correttamente.

```
> python manage.py migrate  
> python manage.py runserver
```

L'applicazione sarà accessibile all'indirizzo `http://localhost:8000/` tramite browser. Effettuare il login o la registrazione per iniziare a utilizzare il sistema.