

Relazione per il corso di

*Basi di Dati*

# **Agriturismo**

Alessandro Rebosio  
Filippo Ricciotti

27 settembre 2025

Anno Accademico 2024-2025

# Indice

<b>1</b>	<b>Analisi dei requisiti</b>	<b>3</b>
1.1	Intervista . . . . .	3
1.2	Estrazione dei concetti principali . . . . .	4
<b>2</b>	<b>Progettazione concettuale</b>	<b>5</b>
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
<b>3</b>	<b>Progettazione Logica</b>	<b>10</b>
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
3.7	Schema relazionale finale . . . . .	11
<b>4</b>	<b>Progettazione della Base di Dati</b>	<b>12</b>
4.1	Check . . . . .	12
4.2	Viste . . . . .	12
4.3	Trigger . . . . .	13
4.4	Traduzione delle operazioni . . . . .	14
4.4.1	Visualizzazione statistiche dashboard . . . . .	14
4.4.2	Prenotazione servizi . . . . .	15
4.4.3	Gestione recensioni . . . . .	16
4.4.4	Gestione iscrizioni eventi . . . . .	17
4.4.5	Gestione prenotazioni servizi . . . . .	18
4.4.6	Eliminazione prenotazioni e iscrizioni . . . . .	19
4.4.7	Autenticazione utente . . . . .	20
4.4.8	Gestione ordini prodotti . . . . .	20
<b>5</b>	<b>Progettazione dell'applicazione</b>	<b>22</b>
5.1	Barra di Navigazione . . . . .	22
5.2	Interfaccia Utente . . . . .	24
5.3	Interfaccia Amministratore . . . . .	28

<b>A</b>	<b>Guida Utente</b>	<b>29</b>
A.1	Clonazione del repository . . . . .	29
A.2	Installazione delle dipendenze . . . . .	29
A.3	Creazione del database . . . . .	29
A.4	Avvio dell'applicazione . . . . .	30

# 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 iscriversi (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.

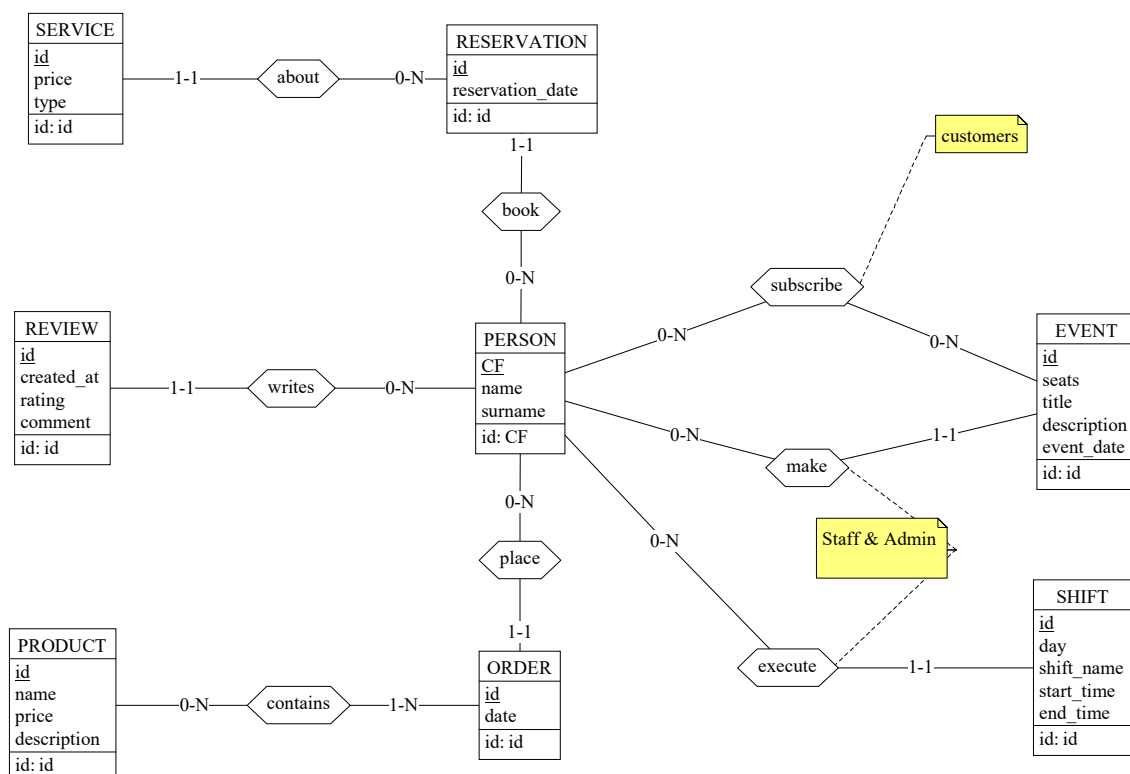


Figura 2.1: Schema ER iniziale

## 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.

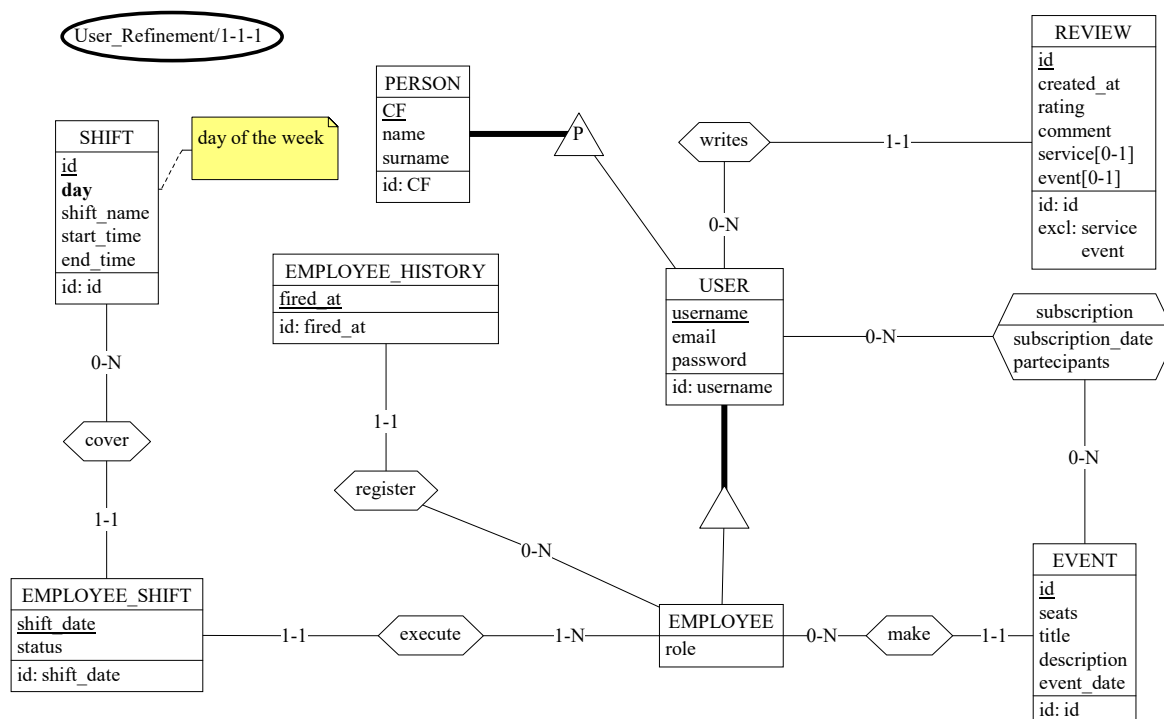


Figura 2.2: Raffinamento utente e dipendente

### 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 affinamento permette di gestire correttamente scenari in cui un utente prenota più servizi differenti nello stesso arco temporale.

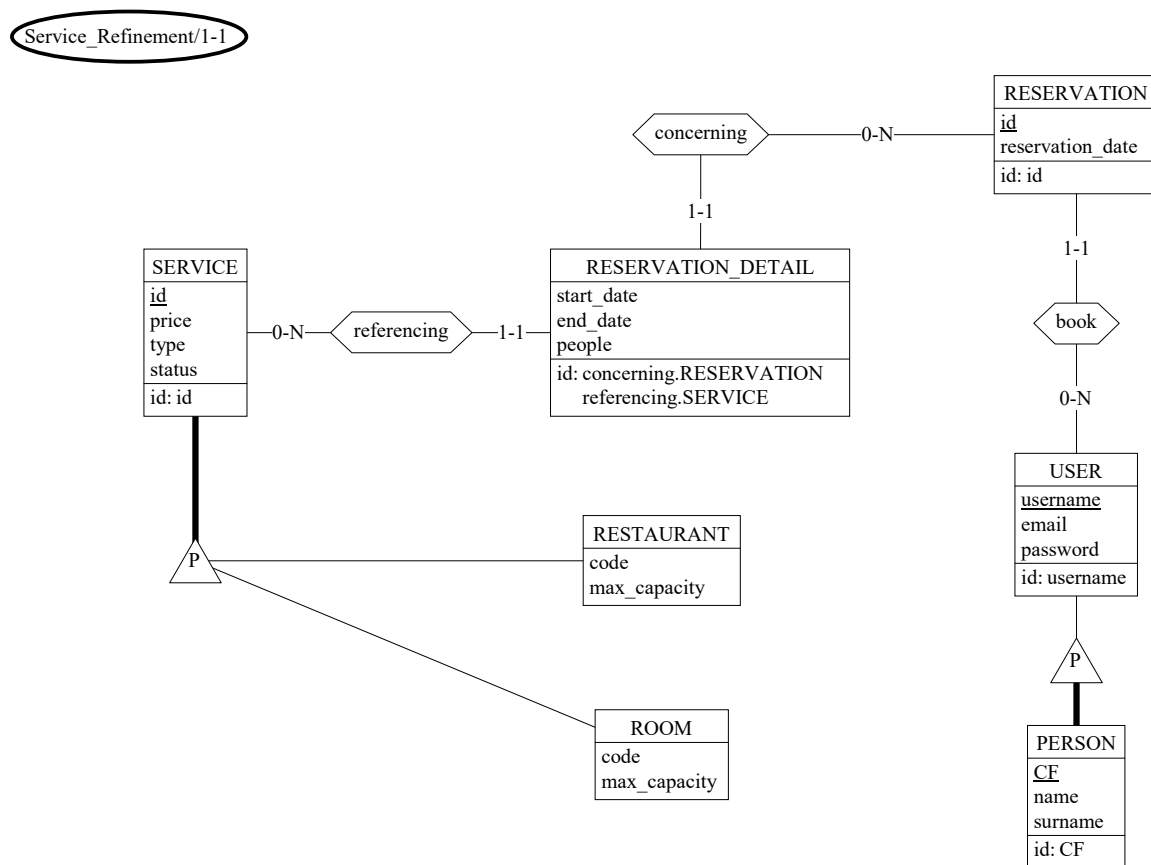


Figura 2.3: Raffinamento prenotazione e servizi



## 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.

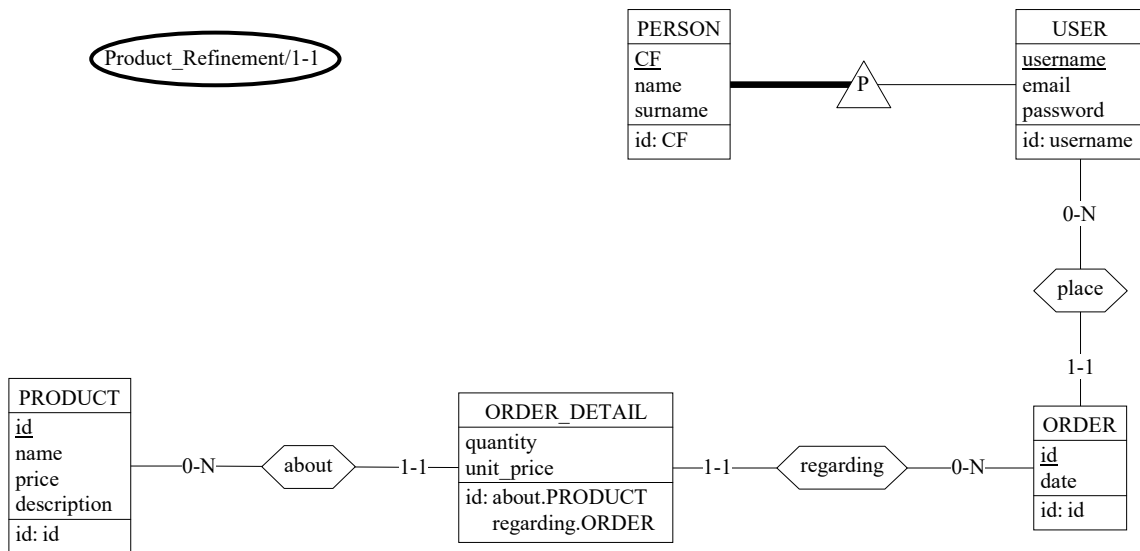


Figura 2.4: Raffinamento prodotti e ordini

## 2.3 Schema concettuale finale

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

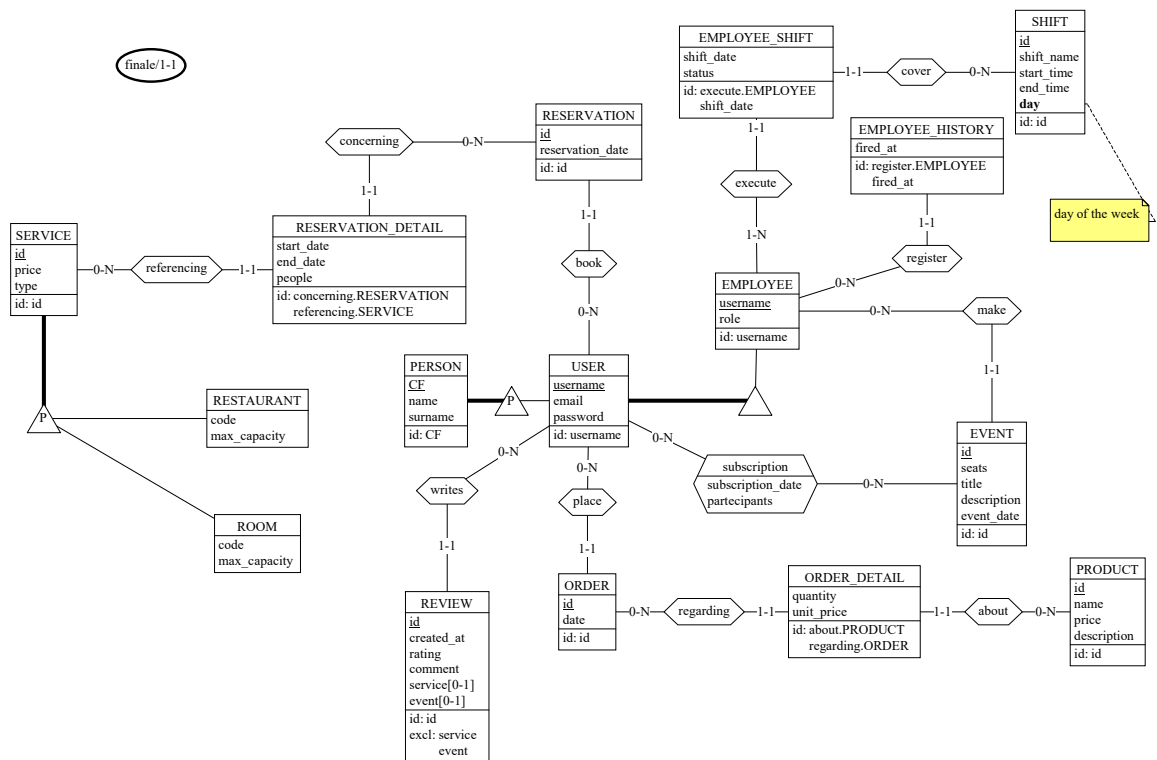


Figura 2.5: Schema ER, schema concettuale finale

# 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

## 3.7 Schema relazionale finale

La fase di riorganizzazione è ora conclusa e lo schema ottenuto è immediatamente traducibile in relazioni relazionali. Di seguito presentiamo lo schema logico.

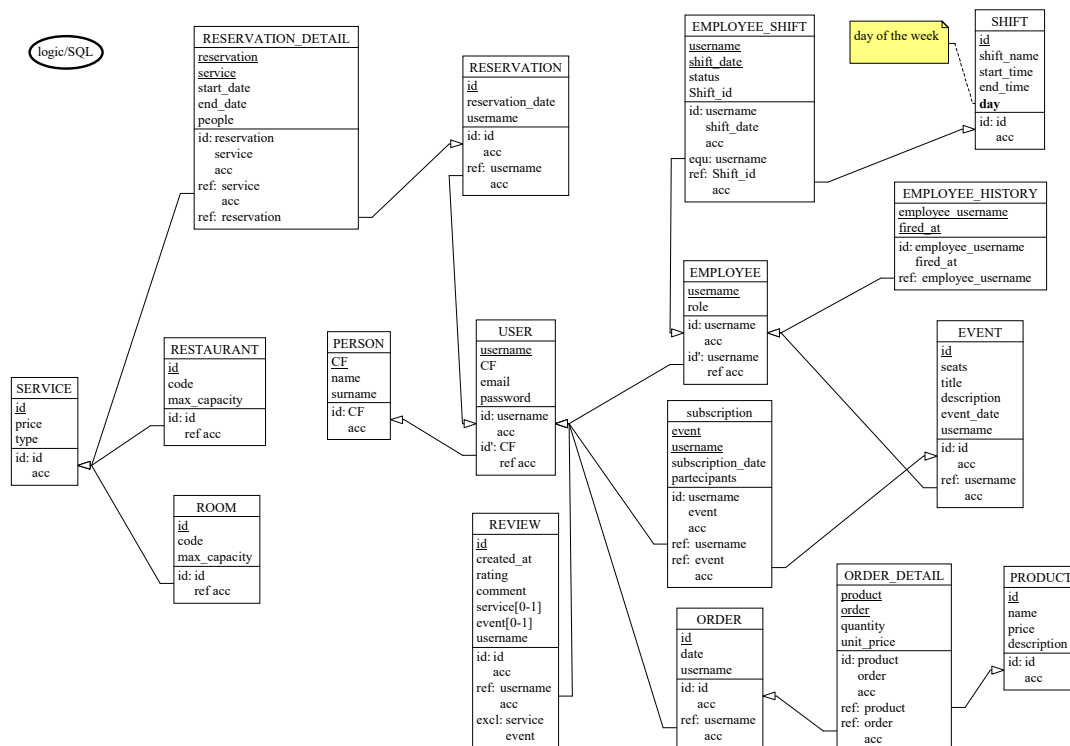


Figura 3.1: Schema relazione finale

# 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

In questa sezione 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.

### 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

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

Listing 4.1: Query per i servizi più prenotati

#### Top prodotti per quantità venduta

Questa operazione calcola i prodotti più venduti in base alla quantità totale ordinata.

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

Listing 4.2: Query per i prodotti più venduti

#### Top eventi per partecipanti

Questa operazione determina gli eventi con il maggior numero di partecipanti totali.

```
1 SELECT e.title as event_title, e.event_date, SUM(es.participants) as  
   total_participants  
2 FROM EVENT e  
3 JOIN EVENT_SUBSCRIPTION es ON e.id = es.event  
4 GROUP BY e.id, e.title, e.event_date  
5 ORDER BY total_participants DESC
```

```
6 LIMIT 4;
```

Listing 4.3: Query per gli eventi più partecipati

## Top prodotti per fatturato

Questa operazione calcola i prodotti che generano il maggior fatturato.

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

Listing 4.4: Query per i prodotti per fatturato

## Fatturato totale

Questa operazione calcola il ricavo complessivo generato dalle vendite dei prodotti.

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

Listing 4.5: Query per il fatturato totale

## Statistiche generali del sistema

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

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

Listing 4.6: Query per le statistiche generali

## 4.4.2 Prenotazione servizi

### Verifica disponibilità camere

La query verifica la disponibilità camere analizzando le prenotazioni esistenti attraverso una condizione di non sovrapposizione temporale, garantendo che vengano selezionate solo le camere con capacità sufficiente e libere nel periodo richiesto.

```
1 SELECT ro.code AS camera, s.price AS prezzo, ro.max_capacity
2 FROM ROOM ro
3 JOIN SERVICE s ON s.id = ro.service
4 WHERE ro.max_capacity >= @n_persone
5 AND ro.service NOT IN (
6     SELECT rd.service
```



```

7      FROM RESERVATION_DETAIL rd
8      WHERE NOT (rd.end_date <= @data_inizio OR rd.start_date >=
      @data_fine)
9  );

```

Listing 4.7: Query per la disponibilità camere

## Verifica disponibilità tavoli

Questa complessa query 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 r.code AS ristorante, s.price AS prezzo, r.max_capacity,
2      (r.max_capacity - IFNULL(SUM(rd.people), 0)) AS
      posti_disponibili
3  FROM RESTAURANT r
4  JOIN SERVICE s ON s.id = r.service
5  LEFT JOIN RESERVATION_DETAIL rd ON rd.service = r.service
6      AND NOT (rd.end_date <= @data_inizio OR rd.start_date >= @data_fine)
7  GROUP BY r.service, r.code, s.price, r.max_capacity
8  HAVING posti_disponibili >= @n_partecipanti;

```

Listing 4.8: Query per la disponibilità tavoli

## 4.4.3 Gestione recensioni

### Inserimento recensione evento

L'operazione verifica tramite multiple condizioni che l'utente sia iscritto all'evento, che questo sia concluso, e che non esista già una recensione da parte dell'user riguardante quell'evento, implementando così un controllo di integrità complesso prima dell'inserimento.

```

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

```

Listing 4.9: Query per inserimento recensione evento

### Inserimento recensione servizio

La query implementa un meccanismo di controllo articolato che verifica l'esistenza di una prenotazione completata e l'assenza di recensioni duplicate, garantendo la correttezza referenziale dell'inserimento.

```

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

```

Listing 4.10: Query per inserimento recensione servizio

## Aggiornamento recensione

Questa operazione modifica il voto e il commento di una recensione esistente.

```

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

```

Listing 4.11: Query per aggiornamento recensione

## 4.4.4 Gestione iscrizioni eventi

### Update e Iscrizione utente a evento

Questa operazione permette a un utente di iscriversi a un evento specificando il numero di partecipanti, con un meccanismo di controllo "ON DUPLICATE KEY UPDATE" siamo in grado di trasformare questa insert in un update che modifica solo il numero di partecipanti, nel caso in cui l'utente avesse 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 e
7 CROSS JOIN USER u
8 WHERE e.title = 'Harvest Festival'
9       AND u.username = 'lblu'
10 ON DUPLICATE KEY UPDATE participants = 4;

```

Listing 4.12: Query per iscrizione evento

## 4.4.5 Gestione prenotazioni servizi

### Creazione prenotazione principale

Questa operazione 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 'gverdi', NOW()
3 WHERE NOT EXISTS (
4     SELECT 1 FROM RESERVATION
5     WHERE username = 'gverdi'
6     AND DATE(reservation_date) = CURDATE()
7 );
```

Listing 4.13: Query per creazione prenotazione principale

### Prenotazione tavolo ristorante

Query specializzata per la prenotazione di tavoli al ristorante, con join sulla tabella RESTAURANT per identificare il servizio corretto.

```
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 s
12 INNER JOIN RESTAURANT r ON s.id = r.service
13 WHERE r.code = 'T01'
14 LIMIT 1;
```

Listing 4.14: Query per prenotazione tavolo ristorante

### Prenotazione camera con controllo duplicati

Operazione completa per la prenotazione di camere che include sia la creazione della prenotazione principale che dei dettagli, con controlli anti-duplicato e gestione degli ID generati.

```
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
```

```

13      @room_reservation_id as reservation,
14      s.id as service,
15      '2024-01-26 15:00:00' as start_date,
16      '2024-01-28 11:00:00' as end_date,
17      2 as people
18 FROM SERVICE s
19 INNER JOIN ROOM r ON s.id = r.service
20 WHERE r.code = 'R03'
21 LIMIT 1;

```

Listing 4.15: Query per prenotazione camera

## 4.4.6 Eliminazione prenotazioni e iscrizioni

### Eliminazione prenotazione servizio

Questa operazione 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  -- Prima eliminiamo i dettagli della prenotazione per evitare
   -- violazioni dei vincoli
2  DELETE FROM RESERVATION_DETAIL
3  WHERE reservation = @reservation_id;
4
5  -- Poi eliminiamo la prenotazione principale
6  DELETE FROM RESERVATION
7  WHERE id = @reservation_id
8  AND username = @username;

```

Listing 4.16: Query per eliminazione prenotazione servizio

### Eliminazione iscrizione evento

Questa operazione rimuove l'iscrizione di un utente a un evento specifico, verificando che l'iscrizione esista e che l'evento non sia già iniziato per consentire solo cancellazioni di eventi futuri.

```

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

```

Listing 4.17: Query per eliminazione iscrizione evento

### Eliminazione prenotazione con controllo temporale

Questa query più sofisticata elimina una prenotazione solo se non è già iniziata, implementando un controllo temporale che previene la cancellazione di prenotazioni in corso o concluse.

```

1  -- Eliminazione condizionata: solo se la prenotazione non e' iniziata
2  DELETE FROM RESERVATION
3  WHERE id = @reservation_id
4         AND username = @username
5         AND NOT EXISTS (
6             SELECT 1 FROM RESERVATION_DETAIL rd
7             WHERE rd.reservation = @reservation_id
8             AND rd.start_date <= NOW()
9         );

```

Listing 4.18: Query per eliminazione prenotazione con controllo temporale

## Eliminazione multipla iscrizioni evento

Operazione amministrativa che permette di cancellare tutte le iscrizioni a un evento specifico, utile in caso di cancellazione dell'evento da parte degli amministratori del sistema.

```

1  DELETE FROM EVENT_SUBSCRIPTION
2  WHERE event = @event_id;

```

Listing 4.19: Query per eliminazione multipla iscrizioni evento

## 4.4.7 Autenticazione utente

### Verifica credenziali di login

La query combina informazioni da multiple tabelle per validare le credenziali e determinare il profilo utente completo, distinguendo tra dipendenti e clienti attraverso un left join strategico.

```

1  SELECT u.username, u.email, u.password, p.name, p.surname,
2         CASE WHEN e.username IS NOT NULL THEN 'employee' ELSE 'customer'
3         END as user_type,
4         e.role as employee_role
5  FROM USER u
6  INNER JOIN PERSON p ON u.cf = p.cf
7  LEFT JOIN EMPLOYEE e ON u.username = e.username
8  WHERE u.username = 'mrossi' OR u.email = 'mrossi@farm.com';

```

Listing 4.20: Query per verifica login

## 4.4.8 Gestione ordini prodotti

### Creazione nuovo ordine

Questa complessa transazione articolata in multiple operazioni crea un nuovo ordine, recupera automaticamente l'ID generato, e inserisce i prodotti con i prezzi correnti, gestendo così tutta la logica di ordine in un'unica procedura.

```

1  INSERT INTO ORDERS (username, date)
2  SELECT 'aneri', NOW()
3  WHERE NOT EXISTS (
4      SELECT 1
5      FROM ORDERS
6      WHERE username = 'aneri' AND DATE(date) = CURDATE()

```

```

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

```

Listing 4.21: Query per creazione ordine

# 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.

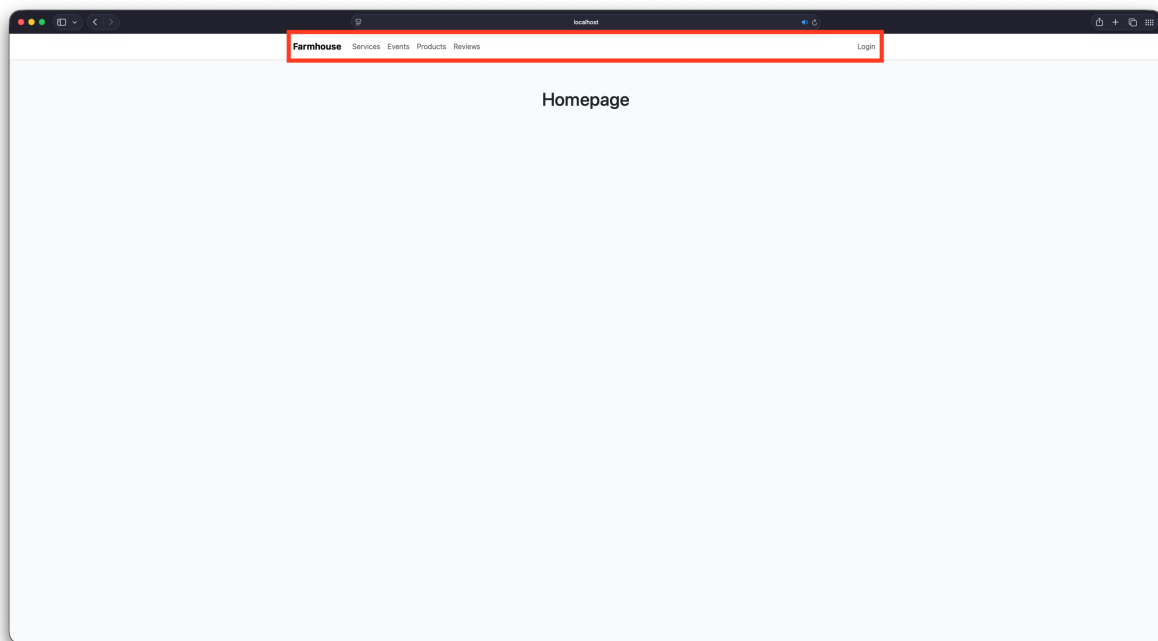
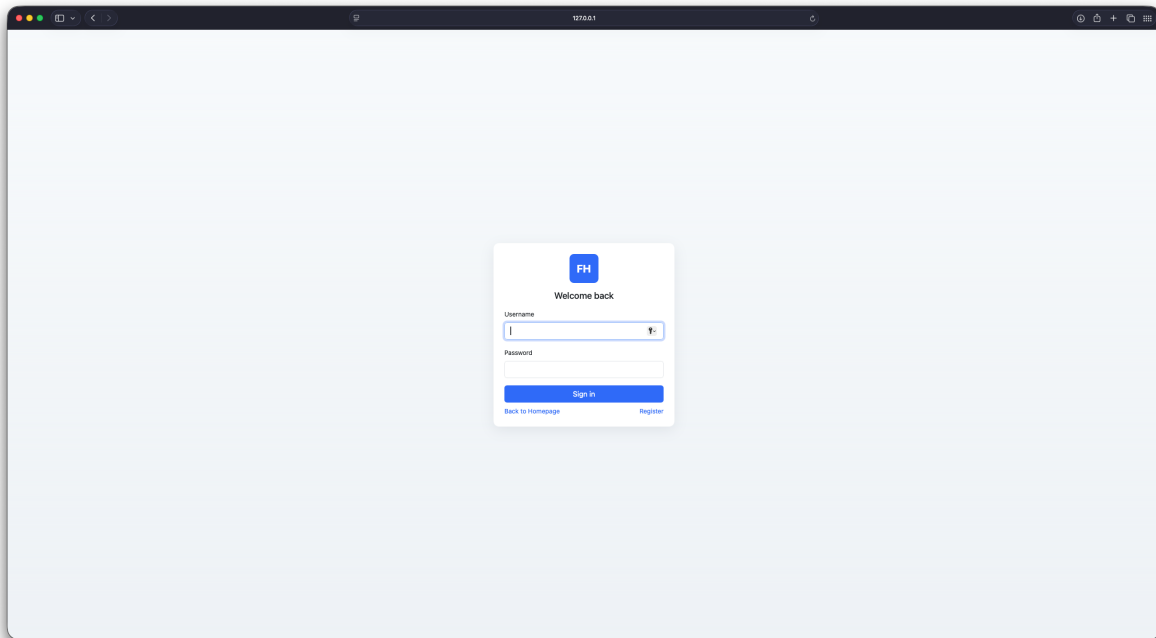


Figura 5.1: Barra di navigazione

## Login

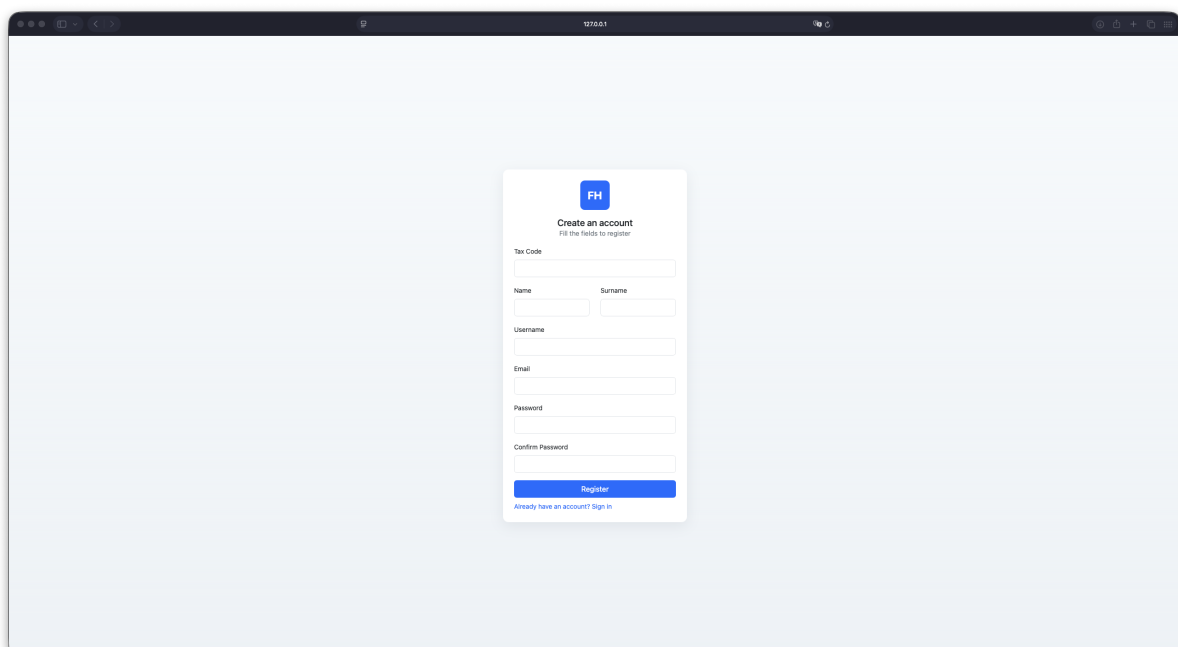
Il form di login consente agli utenti registrati di accedere rapidamente alla piattaforma inserendo username e password. Il sistema verifica le credenziali e, in caso di errore, mostra un messaggio di avviso.



The screenshot shows a web browser window with a light blue background. In the center, there is a white login form. At the top of the form is a blue square logo with the letters 'FH' in white. Below the logo, the text 'Welcome back' is displayed. The form contains two input fields: 'Username' and 'Password'. The 'Username' field has a blue border and a small blue icon on the right. Below the 'Password' field is a blue button labeled 'Sign in'. At the bottom of the form, there 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.

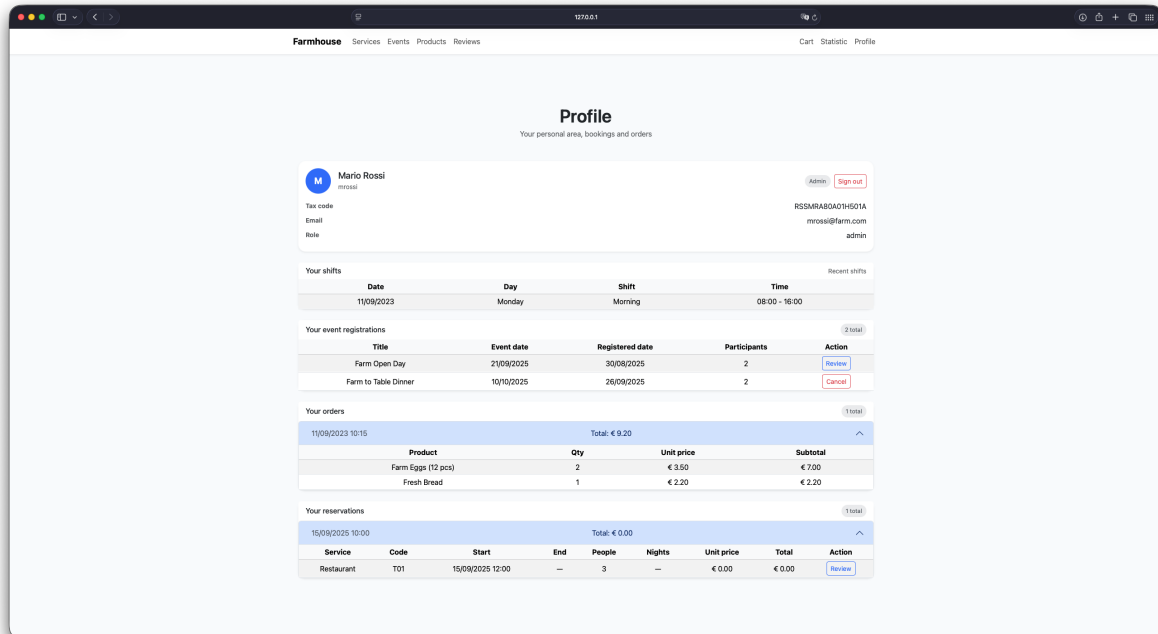


The screenshot shows a web browser window with a light blue background. In the center, there is a white registration form. At the top of the form is a blue square logo with the letters 'FH' in white. Below the logo, the text 'Create an account' is displayed, followed by the instruction 'Fill the fields to register'. The form contains several input fields: 'Tax Code', 'Name', 'Surname', 'Username', 'Email', 'Password', and 'Confirm Password'. At the bottom of the form is a blue button labeled 'Register'. Below the button, there is a link: 'Already have an account? Sign in'.



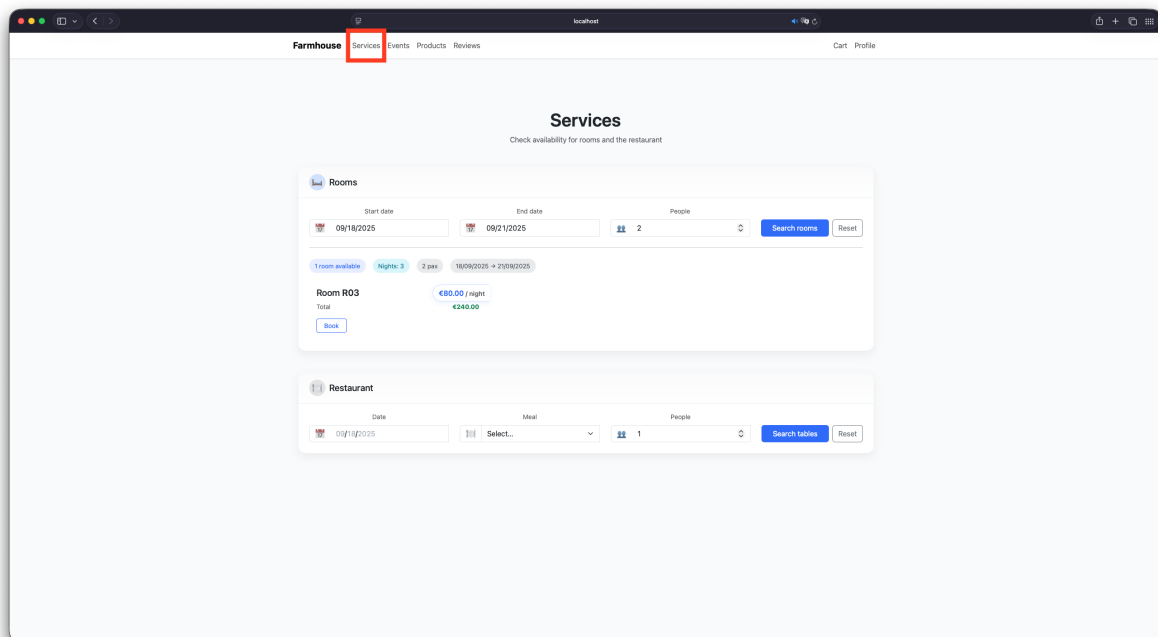
## 5.2 Interfaccia Utente

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



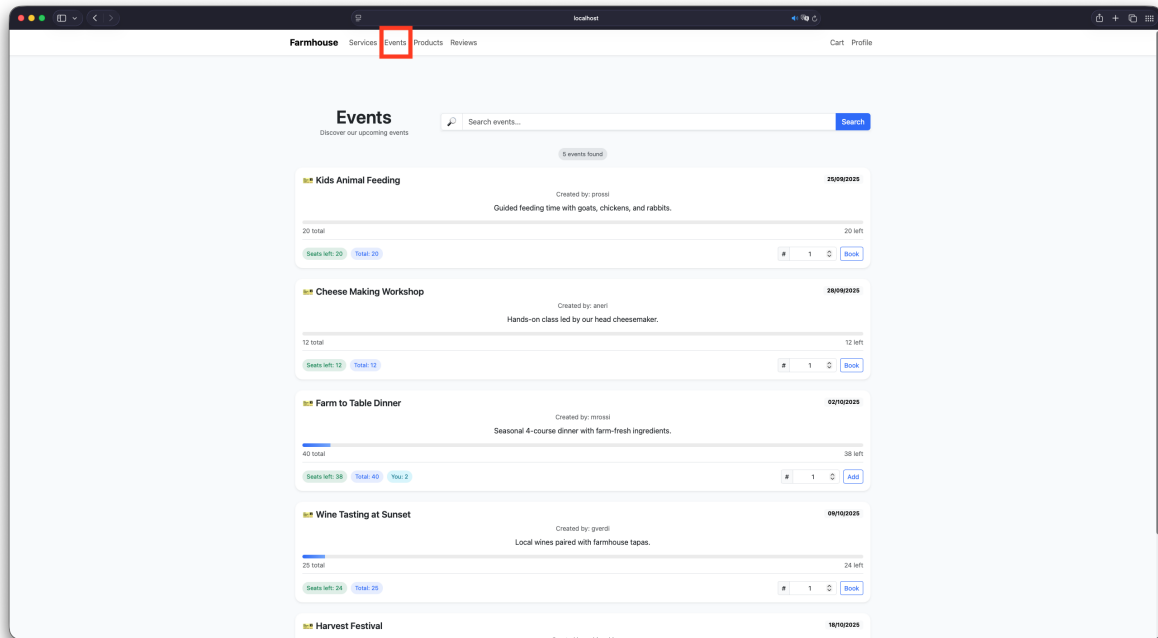
## Servizi

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



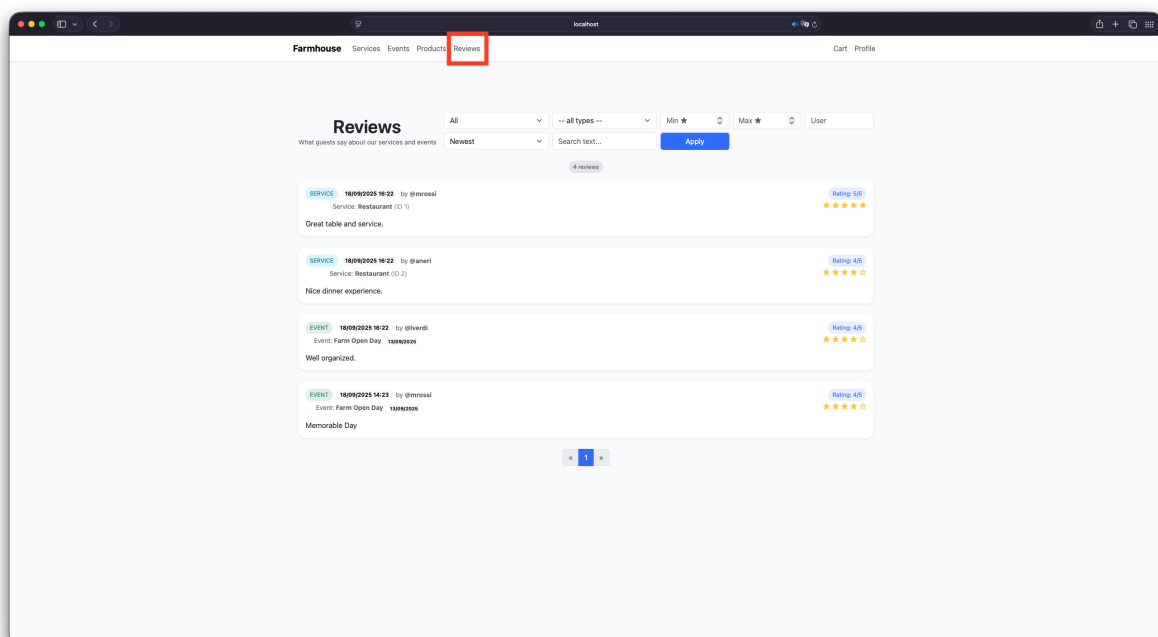
## Eventi

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.

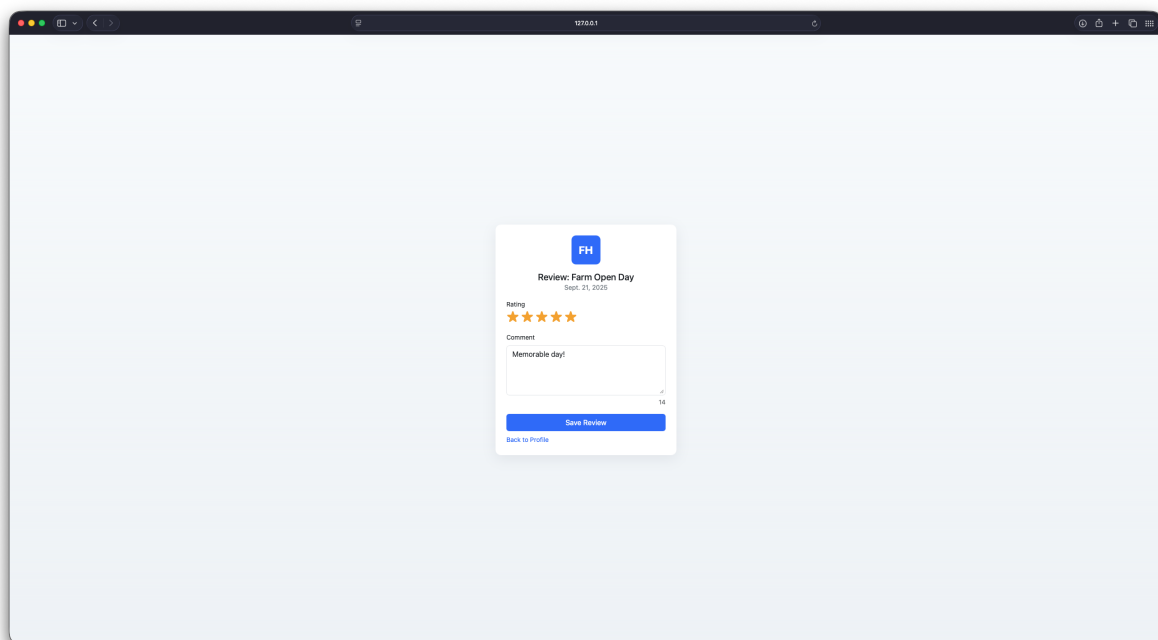


## Recensioni

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

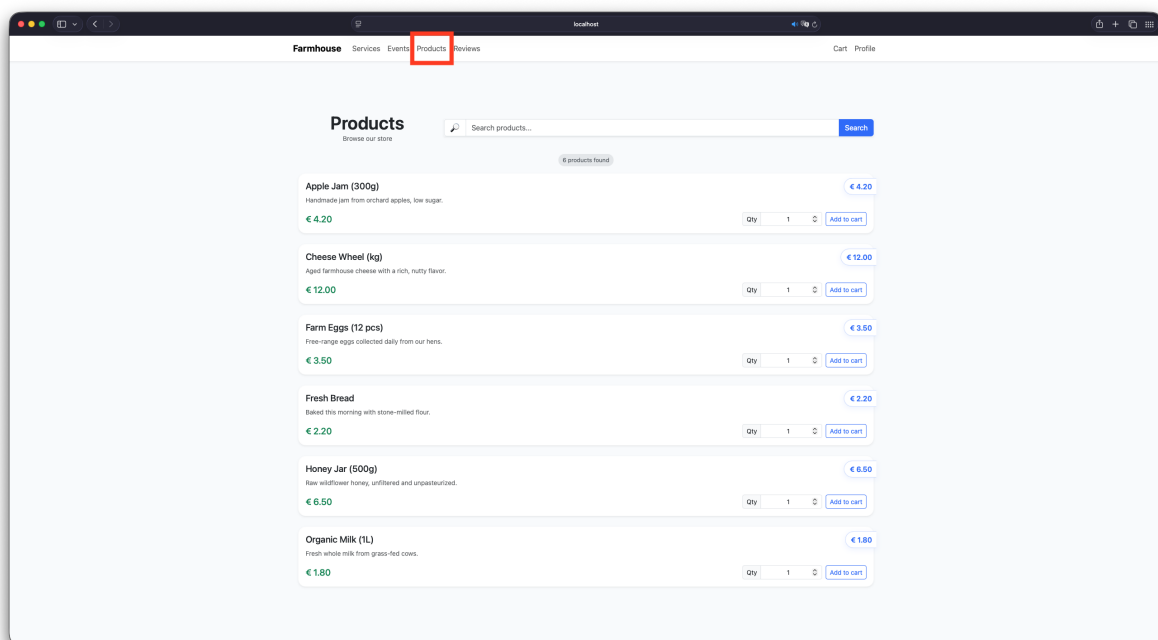


Il form permette agli utenti di lasciare una recensione su eventi o servizi a cui hanno partecipato, inserendo commento e voto. La recensione è consentita solo dopo la partecipazione effettiva.



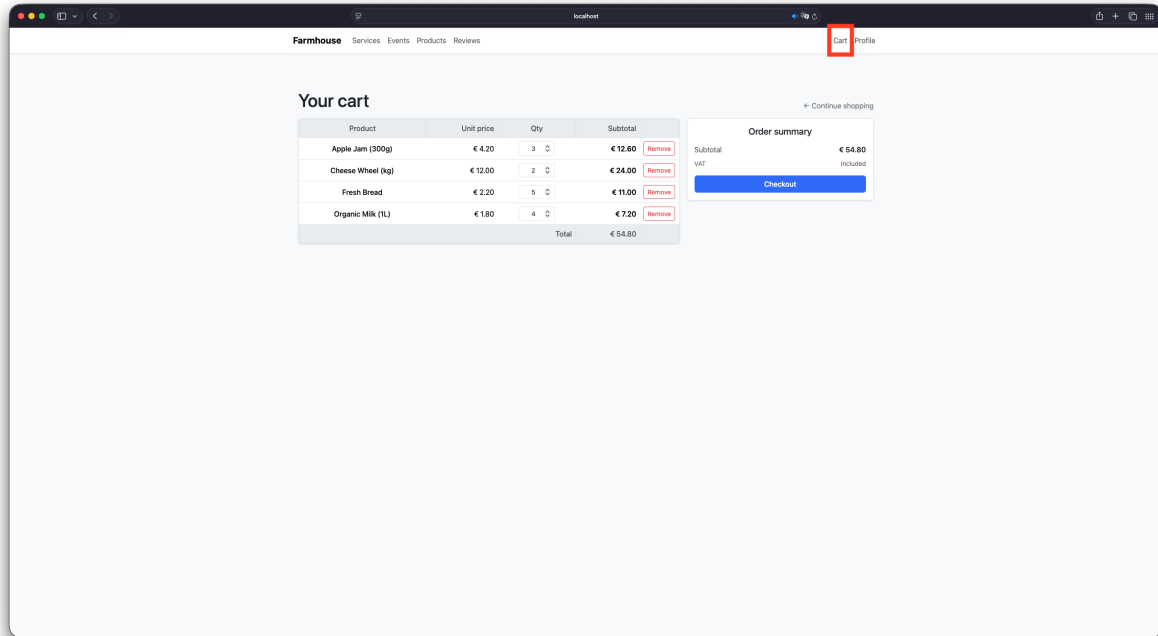
## Prodotti

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.



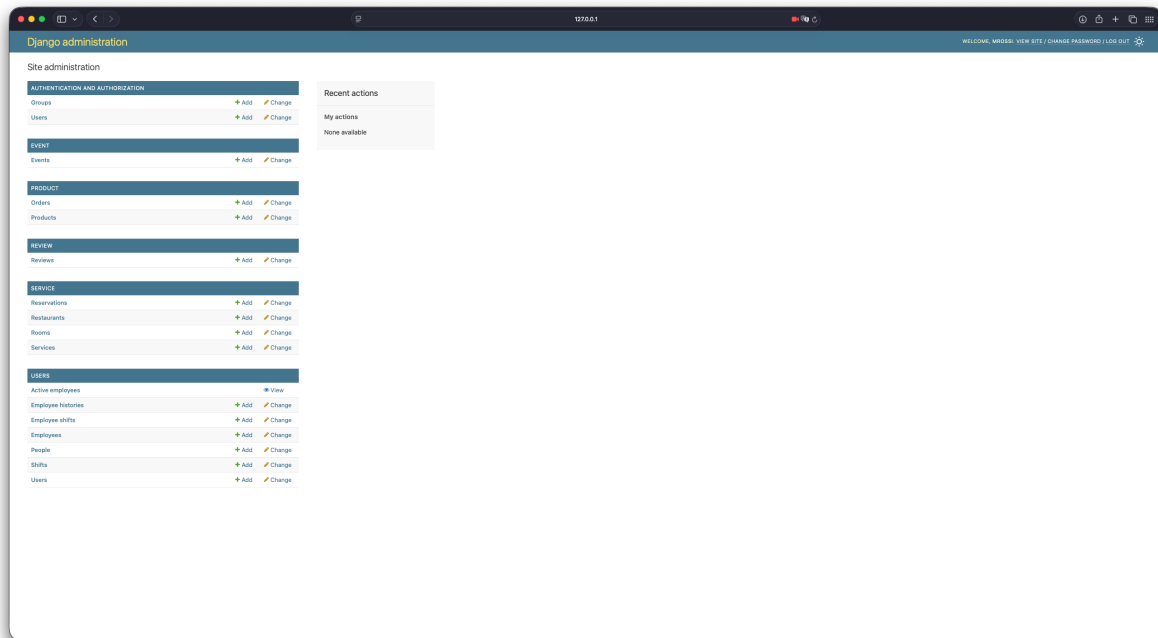
## Carrello

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.

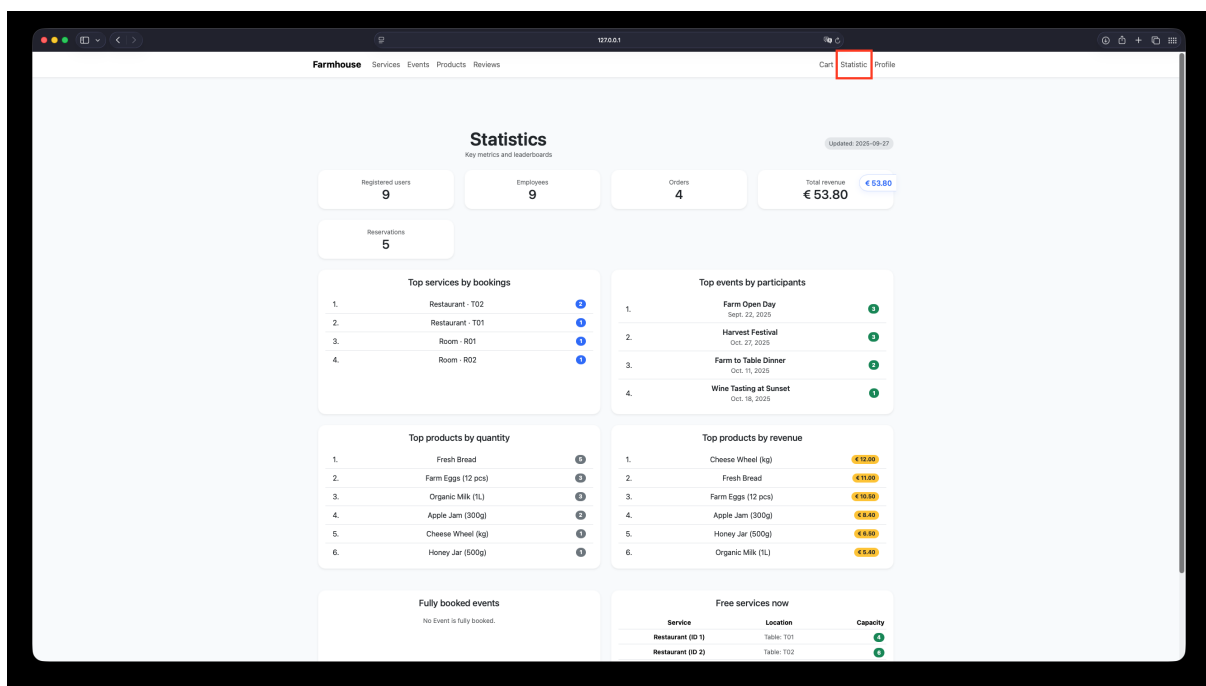


## 5.3 Interfaccia Amministratore

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.