



# UNIVERSITÀ DEGLI STUDI DELL'AQUILA

Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica

CORSO DI LAUREA IN INFORMATICA

Insegnamento: Laboratorio di Base Di Dati

## Gruppo di lavoro:

\* **Nota di coordinamento** – Tutte le attività descritte (analisi, progettazione concettuale e logica, stesura dello schema fisico, implementazione delle funzionalità SQL e della piccola interfaccia web) sono state costantemente supervisionate dall'intero gruppo di lavoro. Ogni modifica al database o al codice applicativo è stata condivisa in tempo reale, garantendo così la coerenza del progetto e l'allineamento delle scelte tecniche fra tutti i membri del team.

Matricola	Cognome	Nome	Contributo al progetto
261682	Marinucci	Alessandro	<ul style="list-style-type: none"><li>Analisi dei requisiti e redazione "Analisi dei requisiti"</li><li>Formalizzazione dei vincoli non esprimibili nel modello ER</li><li>Coordinamento complessivo della documentazione</li></ul>
292216	Odoardi	Davide	<ul style="list-style-type: none"><li>Sviluppo interfaccia web (PHP + HTML + CSS) e mapping delle funzionalità</li><li>Implementazione delle query operative (file <i>market_queries.sql</i> e <i>/operations/</i>)</li><li>Test funzionali e messa a punto ambiente di esecuzione</li></ul>
291659	Ramondo	Mattia	<ul style="list-style-type: none"><li>Disegno del modello ER iniziale e versione ristrutturata</li><li>Traduzione in schema relazionale e script DDL (progettazione fisica)</li><li>Ottimizzazione indici e viste di supporto</li></ul>

Data di consegna del progetto: 25/06/2025

# Analisi dei Requisiti

## Requisiti funzionali (RF)

- RF1 – Inserimento di una richiesta di acquisto da parte dell'ordinante. La richiesta comprende la categoria di prodotto, i valori (o l'opzione "indifferente") per ogni caratteristica prevista dalla categoria e un campo note libero.
- RF2 – Assegnazione di un tecnico incaricato a una richiesta non ancora presa in carico.
- RF3 – Inserimento o modifica di un prodotto candidato da parte del tecnico incaricato, con descrizione completa (produttore, modello, codice, prezzo, URL, note).
- RF4 – Approvazione o rifiuto del prodotto candidato da parte dell'ordinante, con registrazione di una motivazione in caso di rifiuto.
- RF5 – Eliminazione di una richiesta fintanto che non è stata chiusa o non si trova in fase di evasione.
- RF6 – Visualizzazione di liste di richieste filtrate in base a diversi criteri (ad esempio: richieste in corso di un ordinante, richieste non assegnate, richieste di un tecnico con prodotto approvato).
- RF7 – Produzione di statistiche operative: numero di richieste gestite da un tecnico, spesa annuale di un ordinante, tempo medio di evasione.
- RF8 – Registrazione della chiusura delle richieste con esito (accettato, non conforme, non funzionante).
- RF9 – Gestione degli utenti a cura dell'amministratore (creazione account, assegnazione ruoli, autenticazione).

## Requisiti non funzionali e vincoli

- RNF1 – Tecnologia. Il DBMS obbligatorio è MySQL (o, in alternativa, MariaDB). L'interfaccia dimostrativa potrà essere implementata in PHP o altro linguaggio.
- RNF2 – Integrità. I vincoli di dominio devono essere esplicitati nel modello tramite vincoli di tabella o trigger; se impossibile, possono essere delegati all'applicazione.
- RNF3 – Tracciabilità. Sono necessari campi *timestamp* per tutte le fasi chiave del workflow (creazione richiesta, assegnazione tecnico, proposta candidato, approvazione, ordine, consegna, chiusura).
- RNF4 – Sicurezza. Le password degli utenti devono essere salvate con funzione di hash salata (nel progetto è fornito `utils/hash.php`).
- RNF5 – Usabilità. Il sistema deve permettere il valore "indifferente" per ogni caratteristica, così da non costringere l'ordinante a specificare ogni dettaglio.

## Scelte progettuali e assunzioni

- Le categorie di prodotto sono organizzate in un albero modellato con una tabella autoriferita. Non viene fissata una profondità massima.
- Le caratteristiche tecniche sono gestite tramite una relazione molti-a-molti fra categorie e caratteristiche; per ogni coppia si specifica se la caratteristica è obbligatoria o facoltativa.
- Lo stato della richiesta è memorizzato in un campo enumerato che identifica la fase del workflow: NUOVA, ASSEGNATA, CANDIDATO\_PROPOSTO, APPROVATA, ORDINATA, CONSEGNATA, CHIUSA.
- Ogni modifica del prodotto candidato genera un nuovo record, in modo da conservare lo storico delle proposte e delle motivazioni di rifiuto.
- La cancellazione delle richieste avviene in modo logico tramite un flag, così da non perdere dati rilevanti e mantenere la referenzialità.
- Si adottano chiavi surrogate `AUTO_INCREMENT`; eventuali chiavi naturali (per esempio il codice prodotto) sono comunque soggette a vincoli di unicità.
- I prezzi sono salvati in centesimi di euro, utilizzando un intero senza segno, per evitare problemi di arrotondamento.

## Ambiguità individuate e relative decisioni

- Profondità dell'albero delle categorie. Non viene imposta alcuna limitazione; un vincolo `CHECK` e un trigger evitano la formazione di cicli.
- Cambio del tecnico incaricato. È consentito soltanto finché il prodotto candidato non è stato approvato; dopo l'approvazione la responsabilità rimane invariata.

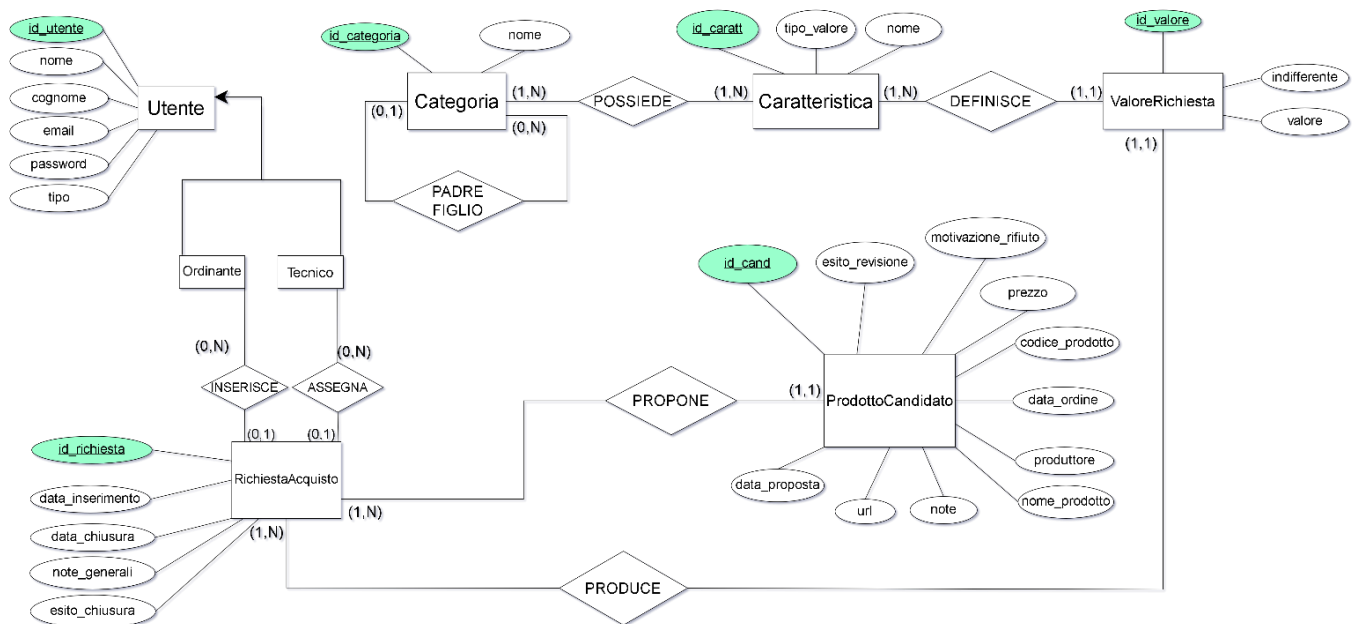
- Più prodotti candidati simultanei. È consentito mantenere uno solo candidato “attivo”; le versioni precedenti restano archiviate come non correnti.
- Caratteristiche obbligatorie o opzionali. Nella relazione categoria-caratteristica si indica se la specifica è obbligatoria. Il valore “indifferente” è sempre ammesso.
- Definizione del tempo di evasione. È calcolato come la differenza fra la data di inserimento della richiesta e quella di emissione dell'ordine. Se l'ordine non viene emesso, il tempo di evasione non è definito.
- Annullamento di una richiesta già approvata. È possibile solo prima dell'emissione dell'ordine; dopo tale momento la richiesta può concludersi soltanto con l'esito di consegna.

### Glossario del dominio

- Ordinate. Utente che inserisce le richieste di acquisto e approva i prodotti candidati; nella struttura organizzativa può corrispondere al “buyer interno”.
- Tecnico. Utente che ricerca il prodotto sul mercato, lo propone e gestisce l'ordine; sinonimi: operatore, responsabile acquisti.
- Amministratore. Utente che crea e gestisce gli account e assegna i ruoli; non partecipa al workflow di acquisto.
- Categoria. Nodo dell'albero merceologico che raggruppa prodotti omogenei.
- Caratteristica. Attributo tecnico (nome, tipo, unità) che descrive un aspetto misurabile di un prodotto, ad esempio RAM o CPU.
- Richiesta di acquisto. Documento iniziale contenente categoria, caratteristiche desiderate e note; è il punto d'avvio del processo.
- Prodotto candidato. Proposta di prodotto reale individuata dal tecnico (marca, modello, codice, prezzo, URL).
- Stato della richiesta. Valore enumerato che indica la fase di avanzamento nel workflow.
- Motivazione di rifiuto. Commento scritto dall'ordinante quando respinge il prodotto candidato.
- Tempo di evasione. Intervallo fra inserimento della richiesta e emissione dell'ordine.
- Spesa. Somma dei prezzi dei prodotti candidati approvati e ordinati per un ordinante all'interno di un anno solare.

## Progettazione concettuale

Di seguito viene presentato il **modello Entity–Relationship** realizzato per il dominio di riferimento



### Entità principali

- **Utente** – Identificato da *id\_utente*, possiede gli attributi anagrafici (*nome*, *cognome*), *email* (univoca), *password* (hash) e *tipo* di utente. È specializzato in due sottoclassi disgiunte e totali:
  - *Ordinante*
  - *Tecnico* Il discriminante di specializzazione è l'attributo *tipo* (valori ammessi: ORDINANTE, TECNICO, ADMIN).
- **Categoria** – Rappresenta una classe merceologica di prodotti, con chiave *id\_categoria* e attributo *nome*. È organizzata in un albero mediante la relazione autoriferita **PADRE–FIGLIO**.
- **Caratteristica** – Attributo parametrico di una categoria. Ogni caratteristica (*id\_caratt*) ha *nome* e *tipo\_valore* (enumerato: BOOLEAN, INTEGER, DECIMAL, TEXT, ENUM).
- **RichiestaAcquisto** – Documento che avvia il processo di approvvigionamento. Chiave *id\_richiesta*; registra *data\_inserimento*, *note\_generali*, *data\_chiusura* ed *esito\_chiusura* (enumerato: ACCETTATO, NON\_CONFORME, NON\_FUNZIONANTE).
- **ProdottoCandidato** – Proposta di acquisto generata dal tecnico. Chiave *id\_cand*; contiene *produttore*, *nome\_prodotto*, *codice\_prodotto*, *prezzo* (in centesimi), *url*, *note*, *data\_proposta*, *esito\_revisione* (APPROVATO, RESPINTO), *motivazione\_rifiuto*, *data\_ordine*.
- **ValoreRichiesta** – Istanza dei valori desiderati dall'ordinante per una caratteristica. Chiave *id\_valore*; campi *valore* (dipende da *tipo\_valore*) e flag *indifferente*.

#### Relazioni chiave e cardinalità

- **INSERISCE** (*Ordinante*, *RichiestaAcquisto*) – Un ordinante può inserire molte richieste (0...N); ogni richiesta è inserita da esattamente un ordinante (1).
- **ASSEGNA** (*Tecnico*, *RichiestaAcquisto*) – Un tecnico può essere assegnato a molte richieste (0...N); una richiesta ha al massimo un tecnico incaricato (0...1) finché non viene conclusa.
- **PROPONE** (*RichiestaAcquisto*, *ProdottoCandidato*) – Ogni richiesta ha uno (e uno solo) prodotto candidato "corrente" (0...1); un prodotto candidato è sempre legato a una singola richiesta (1).
- **PRODUCE** (*ProdottoCandidato*, *RichiestaAcquisto*) – Rappresenta la fase di acquisto vero e proprio; cardinalità 1:1 con PROPONE per semplicità (modellata come stato del candidato).
- **POSSIEDE** (*Categoria*, *Caratteristica*) – Ogni categoria possiede 0...N caratteristiche; una caratteristica può essere usata da più categorie (1...N).
- **DEFINISCE** (*Caratteristica*, *ValoreRichiesta*) – Ogni valore si riferisce a esattamente una caratteristica (1); una caratteristica può essere specificata in molti valori (0...N).
- **PADRE–FIGLIO** (*Categoria*, *Categoria*) – Relazione ricorsiva 1...N che modella la gerarchia.

#### Attributi composti e derivati

- *nome\_completo* di **Utente** è derivabile concatenando *nome* e *cognome*; non viene materializzato.
- *tempo\_evasione* è un attributo derivato calcolato dinamicamente come *data\_ordine* – *data\_inserimento* di **RichiestaAcquisto**.

#### Scelte di modellazione rilevanti

- **ISA vs attributo di ruolo** – È stata scelta una **specializzazione** per separare il comportamento di ordinanti e tecnici, semplificando vincoli diversi (ad esempio solo i tecnici possono proporre un prodotto).
- **Gestione delle caratteristiche** – L'uso di entità separate *Caratteristica* e *ValoreRichiesta* consente di aggiungere nuove caratteristiche senza alterare lo schema fisico, garantendo flessibilità.
- **Opzione "indifferente"** – Implementata con un booleano in *ValoreRichiesta*; se impostato, il campo *valore* può rimanere NULL.
- **Storico dei candidati** – Ogni modifica genera un nuovo record; l'attributo *esito\_revisione* della versione precedente viene aggiornato a RESPINTO, preservando la cronologia.
- **Enumerazioni** – Gli esiti sono modellati con ENUM MySQL per vincolare i valori ammessi e velocizzare confronti.

#### Assunzioni aggiuntive

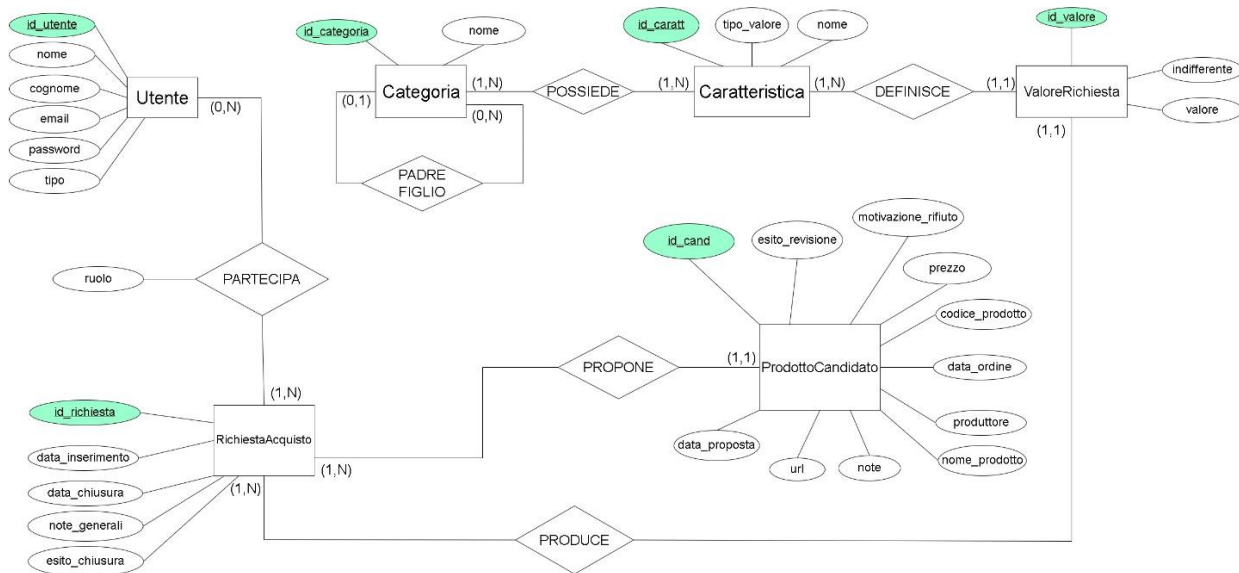
1. Un *tecnico* può essere assegnato a una richiesta soltanto dopo che l'ordinante l'ha completata.
2. Un ordinante non può approvare più prodotti contemporaneamente per la stessa richiesta.
3. La data di consegna, al momento, non è registrata perché esterna al sistema; si assume coincida con il momento in cui l'ordinante chiude la richiesta.

Con questa descrizione, il modello risponde a tutti i requisiti individuati e costituisce la base per la successiva progettazione logica (schema relazionale).

# Progettazione logica

## Ristrutturazione ed ottimizzazione del modello ER

Di seguito si presenta il **modello ER ristrutturato** (Figura 2) ottenuto dalla fase di ottimizzazione. L'obiettivo principale è ridurre il numero di tabelle, semplificare i vincoli di integrità e migliorare le performance di interrogazione senza perdere la semantica del dominio.



### • Principali trasformazioni rispetto al modello concettuale

#### 1. Eliminazione della generalizzazione Ordinate/ Tecnico

- Nel nuovo schema l'entità **Utente** rimane unica; le funzioni di *ordinante* e *tecnico* sono rappresentate dall'attributo **ruolo** all'interno della relazione **PARTECIPA** con **RichiestaAcquisto**.
- Ciò evita due tabelle figlie o l'uso di colonne nullable, semplifica la gestione dei ruoli multipli (uno stesso utente potrebbe partecipare come tecnico ad alcune richieste e come ordinante ad altre) e riduce il numero di join frequenti.

#### 2. Accorpamento delle relazioni INSERISCE e ASSEGNA in PARTECIPA

- Le due interazioni distinte sono confluite in una sola relazione **Utente** ↔ **RichiestaAcquisto** con cardinalità  $(0,N) - (1,N)$  e attributo **ruolo** (ORDINANTE | TECNICO).
- In questo modo un'unica tabella ponte sostituisce due junction table mantenendo l'informazione di ruolo tramite ENUM, con indice composto (*id\_richiesta*, *ruolo*) per garantire al massimo un ordinante e un tecnico per richiesta.

#### 3. Consolidamento dei legami PROPONE / PRODUCE

- Le relazioni rimangono ma sono state riviste le cardinalità per riflettere l'invarianza «un candidato corrente per richiesta».
- L'attributo *esito\_revisione* governa lo stato, permettendo viste indicizzate per filtrare rapidamente i candidati correnti.

#### 4. Normalizzazione di ValoreRichiesta

- Le cardinalità sono state fissate a  $(1,1)$  su entrambi i lati (Figura 2) così da garantire che ogni valore sia sempre agganciato sia alla caratteristica sia alla richiesta, evitando record orfani.

#### 5. Introduzione di indici mirati

- Chiavi composite su (*id\_categoria*, *id\_caratt*) per **POSSIEDE**; (*id\_richiesta*, *id\_caratt*) su **ValoreRichiesta**;

(id\_richiesta, esito\_revisione) su *ProdottoCandidato* per accelerare reportistica.

6. **Denormalizzazione controllata dei prezzi**

- Il prezzo resta in centesimi ma viene duplicato in una vista materializzata *vw\_stat\_spesa* (totale per ordinante/anno) così da evitare somme su grandi volumi a ogni interrogazione.

7. **Rafforzamento della gerarchia delle categorie**

- Il trigger anti-ciclo è stato mantenuto; si è aggiunto un indice path-enumeration (campo path calcolato) per query *descendant-of* più veloci.

**Motivazioni delle scelte**

Le principali motivazioni che hanno guidato la ristrutturazione sono:

- Eliminazione della gerarchia ISA: riduce il numero di tabelle e di join necessari; tutte le interrogazioni sugli utenti avvengono su una sola entità filtrando sul campo ruolo.
- Relazione unica "PARTECIPA": evita la duplicazione di chiavi esterne e consente di recuperare le richieste di un utente – sia come ordinante sia come tecnico – con la stessa sintassi.
- Indici composti mirati: rafforzano i vincoli di integrità e accelerano la reportistica senza introdurre query aggiuntive.
- Vista materializzata della spesa: pre-calcola le aggregazioni più onerose, alleggerendo i job periodici di business intelligence.

Nel complesso il modello logico mantiene la flessibilità necessaria a introdurre nuovi ruoli o stati del processo senza alterare la struttura portante, e nello stesso tempo assicura tempi di risposta adeguati per le query operative e di business intelligence.

## Traduzione del modello ER nel modello relazionale

Questa sezione presenta lo **schema relazionale finale** che sarà implementato in SQL. Per ciascuna tabella sono indicate in grassetto le **chiavi primarie** e in corsivo le **chiavi esterne**.

**Categoria**(id\_categoria, nome, id\_padre)

**Caratteristica**(id\_caratt, tipo\_valore, nome)

**Possiede**(id\_categoria, id\_caratt)

**Utente**(id\_utente, nome, cognome, email, password, tipo)

**Partecipa**(id\_utente, id\_richiesta, ruolo)

**RichiestaAcquisto**(id\_richiesta, data\_inserimento, data\_chiusura, note\_generali, esito\_chiusura)

**ValoreRichiesta**(id\_valore, indifferente, valore, id\_caratt, id\_richiesta)

**ProdottoCandidato**(id\_cand, data\_proposta, esito\_revisione, motivazione\_rifiuto, prezzo, codice\_prodotto, data\_ordine, produttore, nome\_prodotto, note, url, id\_richiesta)

N.B.: le Primary Key sono quelle colorate di **rosso**, le Foreign Key sono quelle colorate di **blu**.

Questo modello relazionale recepisce le ottimizzazioni della fase logica, mantenendo le stesse regole di integrità tramite chiavi primarie, esterne e constraint aggiuntivi.

# Progettazione fisica - Implementazione del modello relazionale

Di seguito è riportato lo **script SQL** completo per la creazione del database *Market* in MySQL 8. Tutti i vincoli esprimibili direttamente nel DDL sono inclusi.

---

-- DROP & CREATE DATABASE

---

```
DROP DATABASE IF EXISTS market;
CREATE DATABASE market
  CHARACTER SET utf8mb4
  COLLATE utf8mb4_unicode_ci;
USE market;
```

---

-- TABELLA: Categoria

---

```
CREATE TABLE Categoria (
  id_categoria INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  id_padre INT NULL,
  CONSTRAINT uq_categoria_nome UNIQUE (nome),
  CONSTRAINT fk_categoria_padre FOREIGN KEY (id_padre)
    REFERENCES Categoria(id_categoria)
    ON UPDATE CASCADE
    ON DELETE SET NULL,
  CHECK (id_padre IS NULL OR id_padre <> id_categoria)
) ENGINE = InnoDB;
```

---

-- TABELLA: Caratteristica

---

```
CREATE TABLE Caratteristica (
  id_caratt INT AUTO_INCREMENT PRIMARY KEY,
  nome VARCHAR(100) NOT NULL,
  tipo_valore ENUM('BOOLEAN', 'INTEGER', 'DECIMAL', 'TEXT', 'ENUM') NOT NULL,
  CONSTRAINT uq_caratteristica_nome UNIQUE (nome)
) ENGINE = InnoDB;
```

---

-- TABELLA: Possiede (junction Categoria ↔ Caratteristica)

---

```
CREATE TABLE Possiede (
  id_categoria INT NOT NULL,
  id_caratt INT NOT NULL,
  PRIMARY KEY (id_categoria, id_caratt),
  CONSTRAINT fk_possiede_categoria FOREIGN KEY (id_categoria)
```

```

        REFERENCES Categoria(id_categoria)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_possiede_caratt FOREIGN KEY (id_caratt)
        REFERENCES Caratteristica(id_caratt)
        ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE = InnoDB;

```

---

-- *TABELLA: Utente*

---

```

CREATE TABLE Utente (
    id_utente INT AUTO_INCREMENT PRIMARY KEY,
    nome      VARCHAR(60) NOT NULL,
    cognome   VARCHAR(60) NOT NULL,
    email     VARCHAR(120) NOT NULL,
    password  CHAR(60) NOT NULL,
    tipo      ENUM('ADMIN', 'ORDINANTE', 'TECNICO') NOT NULL,
    CONSTRAINT uq_utente_email UNIQUE (email)
) ENGINE = InnoDB;

```

---

-- *TABELLA: RichiestaAcquisto*

---

```

CREATE TABLE RichiestaAcquisto (
    id_richiesta INT AUTO_INCREMENT PRIMARY KEY,
    data_inserimento DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    note_generali    TEXT NULL,
    data_chiusura    DATETIME NULL,
    esito_chiusura   ENUM('ACCETTATO', 'NON_CONFORME', 'NON_FUNZIONANTE') NULL,
    CHECK ( (data_chiusura IS NULL AND esito_chiusura IS NULL) OR
            (data_chiusura IS NOT NULL AND esito_chiusura IS NOT NULL) )
) ENGINE = InnoDB;

```

---

-- *TABELLA: Partecipa (Utente ↔ RichiestaAcquisto)*

---

```

CREATE TABLE Partecipa (
    id_utente INT NOT NULL,
    id_richiesta INT NOT NULL,
    ruolo      ENUM('ORDINANTE', 'TECNICO') NOT NULL,
    PRIMARY KEY (id_utente, id_richiesta),
    CONSTRAINT fk_partecipa_utente FOREIGN KEY (id_utente)
        REFERENCES Utente(id_utente)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT fk_partecipa_richiesta FOREIGN KEY (id_richiesta)
        REFERENCES RichiestaAcquisto(id_richiesta)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT uq_utente_ruolo_per_richiesta UNIQUE (id_richiesta, ruolo)
) ENGINE = InnoDB;

```

---



-- TABELLA: ProdottoCandidato

---

```
CREATE TABLE ProdottoCandidato (  
  id_cand          INT AUTO_INCREMENT PRIMARY KEY,  
  id_richiesta     INT NOT NULL,  
  data_proposta    DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  esito_revisione  ENUM('IN_ATTESA','APPROVATO','RESPINTO') NOT NULL DEFAULT 'IN_A  
TTESA',  
  motivazione_rifiuto TEXT NULL,  
  prezzo          INT UNSIGNED NOT NULL CHECK (prezzo > 0),  
  codice_prodotto  VARCHAR(100) NOT NULL,  
  data_ordine     DATETIME NULL,  
  produttore      VARCHAR(100) NOT NULL,  
  nome_prodotto   VARCHAR(120) NOT NULL,  
  note           TEXT NULL,  
  url            VARCHAR(255) NULL,  
  CONSTRAINT fk_cand_richiesta FOREIGN KEY (id_richiesta)  
    REFERENCES RichiestaAcquisto(id_richiesta)  
    ON UPDATE CASCADE ON DELETE CASCADE,  
  INDEX idx_cand_richiesta_esito (id_richiesta, esito_revisione)  
) ENGINE = InnoDB;
```

---

-- TABELLA: ValoreRichiesta

---

```
CREATE TABLE ValoreRichiesta (  
  id_valore      INT AUTO_INCREMENT PRIMARY KEY,  
  id_caratt      INT NOT NULL,  
  id_richiesta   INT NOT NULL,  
  indifferente   BOOLEAN NOT NULL DEFAULT FALSE,  
  valore         VARCHAR(255) NULL,  
  CONSTRAINT fk_valore_caratt FOREIGN KEY (id_caratt)  
    REFERENCES Caratteristica(id_caratt)  
    ON UPDATE CASCADE ON DELETE CASCADE,  
  CONSTRAINT fk_valore_richiesta FOREIGN KEY (id_richiesta)  
    REFERENCES RichiestaAcquisto(id_richiesta)  
    ON UPDATE CASCADE ON DELETE CASCADE,  
  CONSTRAINT uq_valore_richiesta_caratt UNIQUE (id_richiesta, id_caratt),  
  CHECK ( (indifferente = TRUE AND valore IS NULL) OR  
    (indifferente = FALSE AND valore IS NOT NULL) )  
) ENGINE = InnoDB;
```

---

-- INDICI AGGIUNTIVI

---

```
CREATE INDEX idx_possiede_caratt  ON Possiede(id_caratt);  
CREATE INDEX idx_valore_caratt    ON ValoreRichiesta(id_caratt);  
CREATE INDEX idx_categoria_padre  ON Categoria(id_padre);
```

---

*-- FINE SCRIPT CREAZIONE DATABASE*

---

Questo script può essere eseguito così com'è su MySQL 8 (o MariaDB 10.4+) per ottenere una base dati coerente con il modello logico e con i vincoli dichiarati nelle sezioni precedenti.

## Implementazione funzionalità richieste

Di seguito sono riportate – in ordine – le dodici funzionalità previste dalla specifica, ciascuna accompagnata dal relativo frammento SQL (o pseudocodice) coerente con lo schema fisico presentato nel capitolo precedente. I parametri di input sono indicati con il prefisso : secondo la convenzione delle prepared-statement in MySQL/PHP.

---

### Funzionalità 1 - Inserimento di una richiesta di acquisto

*Definizione:* inserimento di una richiesta (categoria, valori delle caratteristiche, note) da parte dell'ordinante.

```
-- Inserimento richiesta + valori caratteristiche (transazione)
START TRANSACTION;
INSERT INTO RichiestaAcquisto (note_generali)
VALUES (:note_generali);
SET @id_richiesta := LAST_INSERT_ID();

INSERT INTO Partecipa (id_utente, id_richiesta, ruolo)
VALUES (:ordinante, @id_richiesta, 'ORDINANTE');

-- :caratteristiche è una lista (id_caratt, indifferente, valore)
INSERT INTO ValoreRichiesta (id_caratt, id_richiesta, indifferente, valore)
VALUES
  -- verrà espanso dal client con tanti record quante sono le caratteristiche
  (:id_caratt_1, @id_richiesta, :indiff_1, :val_1),
  (...);
COMMIT;
```

---

### Funzionalità 2 - Associazione di una richiesta a un tecnico incaricato

```
UPDATE Partecipa
SET id_utente = :tecnico
WHERE id_richiesta = :richiesta
AND ruolo = 'TECNICO';

-- Se ancora non esiste la riga (richiesta non assegnata):
INSERT IGNORE INTO Partecipa (id_utente, id_richiesta, ruolo)
VALUES (:tecnico, :richiesta, 'TECNICO');
```

---

## Funzionalità 3 - Approvazione del prodotto candidato

```
UPDATE ProdottoCandidato
SET esito_revisione = 'APPROVATO'
WHERE id_cand = :id_cand
    AND esito_revisione = 'IN_ATTESA';
```

---

## Funzionalità 4 - Eliminazione di una richiesta (soft-delete)

```
UPDATE RichiestaAcquisto
SET note_generali = CONCAT('[CANCELLATA] ', note_generali),
    data_chiusura = NOW(),
    esito_chiusura = 'NON_CONFORME'
WHERE id_richiesta = :richiesta
    AND data_chiusura IS NULL;
```

---

## Funzionalità 5 - Liste richieste in corso di un ordinante con candidato non ancora valutato

```
SELECT r.*
FROM RichiestaAcquisto r
JOIN Partecipa p ON p.id_richiesta = r.id_richiesta
JOIN ProdottoCandidato c ON c.id_richiesta = r.id_richiesta
WHERE p.id_utente = :ordinante
    AND p.ruolo = 'ORDINANTE'
    AND r.data_chiusura IS NULL
    AND c.esito_revisione = 'IN_ATTESA';
```

---

## Funzionalità 6 - Richieste non ancora assegnate ad alcun tecnico

```
SELECT r.*
FROM RichiestaAcquisto r
LEFT JOIN Partecipa p
    ON p.id_richiesta = r.id_richiesta
    AND p.ruolo = 'TECNICO'
WHERE p.id_utente IS NULL
    AND r.data_chiusura IS NULL;
```

---

## Funzionalità 7 - Richieste di un tecnico con prodotto approvato ma non ancora ordinato

```
SELECT r.*
FROM RichiestaAcquisto r
JOIN Partecipa p ON p.id_richiesta = r.id_richiesta
```

```
JOIN    ProdottoCandidato c ON c.id_richiesta = r.id_richiesta
WHERE   p.id_utente = :tecnico
        AND p.ruolo = 'TECNICO'
        AND c.esito_revisione = 'APPROVATO'
        AND c.data_ordine IS NULL;
```

---

## Funzionalità 8 - Dettaglio completo di una richiesta

```
-- Informazioni testata richiesta
SELECT *
FROM    RichiestaAcquisto
WHERE   id_richiesta = :richiesta;

-- Valori caratteristiche
SELECT vr.*, c.nome AS nome_caratteristica
FROM    ValoreRichiesta vr
JOIN    Caratteristica c USING (id_caratt)
WHERE   vr.id_richiesta = :richiesta;

-- Prodotti candidati (storico)
SELECT *
FROM    ProdottoCandidato
WHERE   id_richiesta = :richiesta
ORDER BY data_proposta DESC;
```

---

## Funzionalità 6 - Conteggio richieste gestite da un tecnico

```
SELECT COUNT(*) AS tot_richieste
FROM    Partecipa
WHERE   id_utente = :tecnico
        AND ruolo = 'TECNICO';
```

---

## Funzionalità 10 - Somma spesa annua di un ordinante

```
SELECT SUM(c.prezzo)/100 AS spesa_euro
FROM    RichiestaAcquisto r
JOIN    Partecipa p ON p.id_richiesta = r.id_richiesta
JOIN    ProdottoCandidato c ON c.id_richiesta = r.id_richiesta
WHERE   p.id_utente = :ordinante
        AND p.ruolo = 'ORDINANTE'
        AND YEAR(c.data_ordine) = :anno
        AND c.esito_revisione = 'APPROVATO'
        AND c.data_ordine IS NOT NULL;
```

---

## Funzionalità 11 - Tempo medio di evasione ordini

```
SELECT AVG(TIMESTAMPDIFF(DAY, r.data_inserimento, c.data_ordine)) AS giorni_medio
FROM RichiestaAcquisto r
JOIN ProdottoCandidato c USING (id_richiesta)
WHERE c.data_ordine IS NOT NULL;
```

---

## Funzionalità 12 - Chiusura di una richiesta alla consegna

```
UPDATE RichiestaAcquisto
SET data_chiusura = NOW(),
    esito_chiusura = :esito -- 'ACCETTATO' / 'NON_CONFORME' / 'NON_FUNZIONANTE'
WHERE id_richiesta = :richiesta
AND data_chiusura IS NULL;
```

**Nota:** nei file PHP `operations/opX_...php` del progetto gli stessi comandi appaiono come prepared-statement MySQLi, con binding parametri e gestione delle transazioni; lo script `database/market_queries.sql` raggruppa invece le view usate per Funzionalità 10 e 11. Questi frammenti consolidano in un unico punto la logica SQL, pronta per l'inclusione nella documentazione.

---

## Interfaccia verso il database

Il progetto include una **interfaccia web minimale** sviluppata in PHP 8.1 che consente di esercitare tutte le funzionalità definite nei capitoli precedenti senza dover ricorrere alla riga di comando MySQL. L'interfaccia è già presente nella cartella `web/` del repository consegnato.

### Tecnologie impiegate

- **PHP 8.1** con estensione **MySQLi** per la connessione al database e l'esecuzione di query tramite prepared-statement.
- **HTML 5** per la struttura delle pagine e dei form.
- **CSS 3** (foglio `style.css`) per la presentazione e il layout responsivo, senza dipendenze da framework esterni.

### Architettura e mapping delle funzionalità e mapping delle funzionalità

Funzionalità	Script PHP	Metodo / Endpoint	Descrizione sintetica
1 – Inserimento richiesta	<code>operations/op1_inserimento_richiesta.php</code>	POST /op1	Riceve categoria, JSON con caratteristiche, note; esegue transazione SQL (#1).
2 – Associa tecnico	<code>operations/op2Associazione_tecnico.php</code>	POST /op2	Parametri <code>id_richiesta</code> e <code>id_tecnico</code> ; aggiorna/crea record <i>Partecipa</i> .

Funzionalità	Script PHP	Metodo / Endpoint	Descrizione sintetica
3 – Approvazione e candidato	operations/op3_approvazione_prodotto.php	-	Vedi “Nota di progetto - Gestione del prodotto Candidato”
4 – Eliminazione richiesta	operations/op4_eliminazione_richiesta.php	POST /op4	Soft-delete con aggiornamento campi chiusura.
5 – Lista richieste ordinante	operations/op5_lista_richieste_ordinante.php	GET /op5?id_ordinante=...	Restituisce JSON con richieste in corso e candidato in attesa.
6 – Richieste non assegnate	operations/op6_lista_richieste_non_assegnate.php	GET /op6	Query su <i>RichiestaAcquisto</i> senza tecnico.
7 – Lista richieste tecnico	operations/op7_lista_richieste_tecnico.php	GET /op7?id_tecnico=...	Filtra candidati approvati non ordinati.
8 – Dettaglio richiesta	operations/op8_dettaglio_richiesta.php	GET /op8?id=...	Con più SELECT aggregate (funz. 8).
9 – Conteggio richieste tecnico	operations/op9_conteggio_richieste_tecnico.php	GET /op9?id_tecnico=...	COUNT(*).
10 – Calcolo spesa ordinante	operations/op10_calcolo_spesa_ordinante.php	GET /op10?id_ordinante=...&anno=YYYY	Aggregazione SUM().
11 – Tempo medio evasione	operations/op11_tempo_medio_evasione.php	GET /op11	AVG TIMESTAMPDIF (vista).
12 – Chiusura richiesta	operations/op12_chiusura_richiesta.php	POST /op12	Aggiorna esito e data chiusura.

Tutte le **rotte** sono incluse in web/index.php, che funge da *front-controller* e reindirizza alle singole operazioni dopo aver verificato la sessione utente.

## Nota di progetto - Gestione del prodotto candidato

Attualmente non esiste alcuno script predefinito per l'inserimento di record nella tabella **ProdottoCandidato**. Quando occorre svolgere l'Operazione 3 (presa in carico, approvazione e successivo ordine del prodotto candidato) il personale tecnico redige manualmente la relativa istruzione INSERT ed esegue il comando tramite *MySQL Workbench*.

## Modalità di esecuzione in locale

1. Copiare l'intera cartella web/ in htdocs/ (XAMPP) o public\_html/ (Apache + PHP).
2. Configurare in web/db\_connection.php le credenziali MySQL (host, utente, password).
3. Importare lo schema e i dati di esempio:

```
mysql -u root -p < database/market_schema.sql  
mysql -u root -p market < database/data.sql
```

4. Accedere via browser a <http://localhost/web/> con le credenziali di test (presenti in data.sql).

L'interfaccia permette l'inserimento e la consultazione delle richieste, nonché l'esecuzione di tutte le query previste tramite pulsanti o menu contestuali; le risposte AJAX sono visualizzate in tabelle HTML stilizzate con Bootstrap.

Il **codice sorgente** completo dell'interfaccia è già incluso nella cartella del progetto consegnato e deve essere considerato allegato alla presente relazione.