Piattaforma di Food Delivery

Nathan Luboz – classe 5C IT

19/12/2025

Contents

Piattaforma di Food Delivery Nathan Luboz classe 5C IT 19/12/25

Introduzione al progetto

Il progetto ha come obiettivo la realizzazione di un database relazionale per una piattaforma di food delivery, in grado di gestire in modo efficiente le informazioni relative a ristoranti, clienti, ordini e consegne. L'intento è quello di creare una struttura dati robusta e coerente, capace di supportare le principali funzionalità di un sistema di consegna di cibo a domicilio, come la registrazione degli utenti, il catalogo dei piatti offerti dai vari ristoranti, la gestione degli ordini e la tracciabilità delle consegne.

Nel corso dello sviluppo del progetto sono state realizzate le seguenti fasi principali:

-Analisi e progettazione concettuale: costruzione del diagramma ER (Entity-Relationship) per rappresentare in modo chiaro le entità coinvolte e le loro relazioni.

-Progettazione logica: trasformazione del modello concettuale nello schema logico, ottimizzato per l'implementazione in un sistema di gestione di basi di dati relazionali, ovvero MariaDB.

-Implementazione fisica: scrittura del codice DDL per la creazione delle tabelle, delle chiavi primarie e delle chiavi esterne.

Il risultato finale è un database completo, pensato per essere una base solida su cui costruire le componenti applicative della piattaforma di food delivery.

Analisi dei requisiti

I requizisiti necessari per la progettazione di questo database sono:

- Dei ristoranti partner, che comprendevano al loro interno i rispettivi menu e gli orari di apertura nei diversi giorni della settimana;
- I piatti che sono venduti nel ristorante a cui sono correlati, che comprendono gli ingredienti, i possibili allergeni contenuti, e il quantitativo di disponibilità di un piatto;
- I clienti, di cui vengono fornite come informazioni gli indirizzi di consegna, che può essere uno solo o multipli;
- Gli ordini, che possono assumere tre stati nella fase di preparazione e consegna del piatto: ricevuto, in preparazione, in consegna e consegnato;
- Gli ordini possono essere caratterizzati anche da dei dettagli, che esprimino i piatti ordinati, la loro quantità e se sono avvenute delle personalizzazioni del piatto;
- Ogni ristorante assume vari rider, ognuno che copre delle diverse zone della città in cui portare gli ordini ai clienti;
- Ad ogni rider appunto, sarà assegnato di volta in volta un ordine diverso, sempre appartenente alla sua zona di lavoro;
- Il pagamento di un ordine , che puo avvenire sia a livello fisico, quindi dando poi la somma di denaro necessaria al rider al suo arrivo, oppure digitalmente, pagando in app per esempio;
- Ogni cliente dopo aver effettuato l'ordine può lasciare una recensione sull'esperienza, dando anche un voto sia al ristorante che al piatto;
- A volte, può essere che un piatto abbia una promozione, o che al cliente venga applicato uno sconto su un ordine;

Schema logico

INGREDIENTE_ALLERGENE(FK_IDingrediente, FK_IDallergene)

RISTORANTE_GIORNO(FK_IDristorante, FK_IDgiorno)

ORDINE(IDordine, stato, FKEmail, FKIDindirizzo)

RIDER(CF, nome, cognome, iban)

ZONA(IDzona)

COMUNE(IDcomune, nome)

CLIENTE(Email, nTel, nome, cognome)

INDIRIZZO(IDindirizzo, FK_Email, FK_IDcomune)

METODO_PAGAMENTO(IDmetodo)

CARTA(FK_IDmetodo, pagaInApp, banca, Ncarta)

RECENSIONE(IDrecensione, descrizione, stelle, FK_IDordine)

PROMOZIONE(IDpromozione, dataInizio, dataFine, nome)

PIATTO(IDpiatto, prezzo, disponibilità, FK_IDristorante)

INGREDIENTE(IDingrediente, nome, quantita)

ALLERGENE(IDallergene, nome, personalizzazioni)

RISTORANTE(IDristorante, indirizzo)

GIORNO(IDgiorno, nome, orarioApertura, orarioChiusura)

ORDINE_RIDER(FK_IDordine, FK_CF)

RIDER_ZONA(FK_CF, FK_IDzona)

ZONA_COMUNE(FK_IDzona, FK_IDcomune)

ORDINE_PAGAMENTO(FK_IDordine, FK_IDmetodo)

ORDINE_PROMOZIONE(FK_IDordine, FK_IDpromozione)

PROMOZIONE_PIATTO(FK_IDpromozione, FK_IDpiatto)

PIATTO_ORDINE(FK_IDpiatto, FK_IDordine)

INGREDIENTE_PIATTO(FK_IDingrediente, FK_IDpiatto)

Codice DDL

```
/*
Vincoli usati:
CHECK (Email LIKE '%@%') → vincolo di dominio sulla struttura dell'email
Impedisce l'inserimento di email non valide.
*/
CREATE TABLE CLIENTE (

    Email VARCHAR(100) PRIMARY KEY,
    nTel VARCHAR(20) NOT NULL,
    nome VARCHAR(50) NOT NULL,
    cognome VARCHAR(50) NOT NULL,
```

```
    CHECK (Email LIKE '%@%')
);

/*
FOREIGN KEY verso CLIENTE con ON DELETE CASCADE

FOREIGN KEY verso COMUNE con ON DELETE RESTRICT

Un indirizzo deve appartenere a un cliente esistente.
Se un cliente viene eliminato, i suoi indirizzi non hanno senso; invece un comune non può es
*/
CREATE TABLE INDIRIZZO (
    IDindirizzo INT PRIMARY KEY,
    FK_Email VARCHAR(100),
    FK_IDcomune INT,
   FOREIGN KEY (FK_Email) REFERENCES CLIENTE(Email)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (FK_IDcomune) REFERENCES COMUNE(IDcomune)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

/*
CHECK (stato IN (...)) → vincolo di dominio

FOREIGN KEY verso CLIENTE (CASCADE)

FOREIGN KEY verso INDIRIZZO (RESTRICT)

Ogni ordine deve avere uno stato valido ed essere associato a un cliente e a un indirizzo es
*/
CREATE TABLE ORDINE (
    IDordine INT PRIMARY KEY,
    stato VARCHAR(30),
    FKEmail VARCHAR(100),
    FKIDindirizzo INT,
    CHECK (stato IN ('ricevuto', 'in preparazione', 'in consegna', 'consegnato')),
    FOREIGN KEY (FKEmail) REFERENCES CLIENTE(Email)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (FKIDindirizzo) REFERENCES INDIRIZZO(IDindirizzo)
        ON DELETE RESTRICT
        ON UPDATE CASCADE
);

/*
```

```
CHECK (LENGTH(CF) = 16)

La lunghezza del codice fiscale non può essere diversa da 16.
*/
CREATE TABLE RIDER (
    CF VARCHAR(16) PRIMARY KEY,
    nome VARCHAR(50),
    cognome VARCHAR(50),
    iban VARCHAR(34),
     CHECK (LENGTH(CF) = 16)
);

/*
CHECK (orarioApertura < orarioChiusura) → vincolo di tupla

Garantisce orari coerenti.
*/
CREATE TABLE GIORNO (
    IDgiorno INT PRIMARY KEY,
    nome VARCHAR(20),
    orarioApertura TIME,
    orarioChiusura TIME,
     CHECK (orarioApertura < orarioChiusura)
);

/*
CHECK (prezzo > 0)

FOREIGN KEY verso RISTORANTE (CASCADE)

Un piatto non può esistere senza ristorante e deve avere un prezzo valido.
*/
CREATE TABLE PIATTO (
    IDpiatto INT PRIMARY KEY,
    prezzo DECIMAL(6,2),
    disponibilita BOOLEAN,
    FK_IDristorante INT,
    CHECK (prezzo > 0),
    FOREIGN KEY (FK_IDristorante) REFERENCES RISTORANTE(IDristorante)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

/*
CHECK (LENGTH(Ncarta) BETWEEN 13 AND 20)

FOREIGN KEY verso METODO_PAGAMENTO (CASCADE)
```

```sql
Ogni carta è associata a un solo metodo di pagamento.
*/
CREATE TABLE CARTA (
    FK_IDmetodo INT PRIMARY KEY,
    pagaInApp BOOLEAN,
    banca VARCHAR(50),
    Ncarta VARCHAR(20) unique,
    CHECK (LENGTH(Ncarta) BETWEEN 13 AND 20),
    FOREIGN KEY (FK_IDmetodo) REFERENCES METODO_PAGAMENTO(IDmetodo)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

/*
CHECK (stelle BETWEEN 1 AND 5)

FOREIGN KEY verso ORDINE (CASCADE)

Un ordine può avere una sola recensione e la valutazione deve essere valida.
*/
CREATE TABLE RECENSIONE (
    IDrecensione INT PRIMARY KEY,
    descrizione VARCHAR(255),
    stelle INT CHECK (stelle BETWEEN 1 AND 5),
    FK_IDordine INT,
    CHECK (stelle BETWEEN 1 AND 5),
    FOREIGN KEY (FK_IDordine) REFERENCES ORDINE(IDordine)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

/*
RIMARY KEY composta

FOREIGN KEY con ON DELETE CASCADE

Impediscono duplicazioni della stessa relazione e garantiscono che le associazioni esistano

Questo è valido per tutte le tabelle ausiliarie che seguono.
*/
CREATE TABLE ORDINE_RIDER (
    FK_IDordine INT,
    FK_CF VARCHAR(16),
    PRIMARY KEY (FK_IDordine, FK_CF),
     FOREIGN KEY (FK_IDordine) REFERENCES ORDINE(IDordine)
        ON DELETE CASCADE
```

```
        ON UPDATE CASCADE,
    FOREIGN KEY (FK_CF) REFERENCES RIDER(CF)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE INGREDIENTE_ALLERGENE (
    FK_IDingrediente INT,
    FK_IDallergene INT,
    PRIMARY KEY (FK_IDingrediente, FK_IDallergene),
    FOREIGN KEY (FK_IDingrediente) REFERENCES INGREDIENTE(IDingrediente)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (FK_IDallergene) REFERENCES ALLERGENE(IDallergene)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);

CREATE TABLE RISTORANTE_GIORNO (
    FK_IDristorante INT,
    FK_IDgiorno INT,
    PRIMARY KEY (FK_IDristorante, FK_IDgiorno),
    FOREIGN KEY (FK_IDristorante) REFERENCES RISTORANTE(IDristorante)
        ON DELETE CASCADE
        ON UPDATE CASCADE,
    FOREIGN KEY (FK_IDgiorno) REFERENCES GIORNO(IDgiorno)
        ON DELETE CASCADE
        ON UPDATE CASCADE
);
```

Codice per il diagramma E-R

erDiagram

```
ORDINE{
    int IDordine PK
    string stato
}

RIDER{
    int CF PK
    string nome
    string cognome
    string iban
}

ZONA{
    int IDzona PK
```

```
}

COMUNE{
    int IDcomune PK
    string nome
}

CLIENTE{
    string Email PK
    int nTel "NULL"
    string nome
    string cognome
}

INDIRIZZO{
    string email FK
}

RECENSIONE{
    int IDrecensione PK
    int IDordine FK
    string descrizione
    double stelle
}

METODO_PAGAMENTO{
    int IDmetodo PK
}

CARTA{
    boolean pagaInApp
    string banca
    int Ncarta
}

METODO_PAGAMENTO ||--o{ CARTA : isa

ORDINE o|--o{ RIDER : portato_da
RIDER ||--o{ ZONA : opera_a
ZONA |{--o| COMUNE : include
COMUNE o{--|| INDIRIZZO : ind_del_com
INDIRIZZO ||--|{ CLIENTE : dove_abita
ORDINE o{--o| CLIENTE : ordine_di
INDIRIZZO o{--|| ORDINE : indirizzo_ordine
ORDINE ||--o{ METODO_PAGAMENTO : viene_pagato
RECENSIONE ||--o{ ORDINE : recensione_di

PROMOZIONE{
    int IDpromozione PK
    date dataInzio
```

```
    date dataFine
    string nome
}

PIATTO{
    int IDpiatto PK
    float prezzo
    int disponibilità
}

INGREDIENTE{
    int IDingrediente PK
    float quantita
    string nome
}

ALLERGENE{
    int IDallergene PK
    string nome
    string personalizzazioni
}

RISTORANTE{
    int IDristorante PK
    string indirizzo
}

GIORNO{
    int IDgiorno PK
    string nome
    time orarioApertura
    time orarioChiusura
}

ORDINE o{--o{ PROMOZIONE: prevede
PROMOZIONE o{--o{ PIATTO: riferita
PIATTO o{--|{ ORDINE: compongono
INGREDIENTE o{--o{ PIATTO: e_contenuto
INGREDIENTE  o{--o{ ALLERGENE: puo_contenere
RISTORANTE  o{--|| PIATTO : vende
RISTORANTE |{-- o{ GIORNO: apertura_in_che_giorno
```

Dizionario dei dati

CLIENTE

Email VARCHAR(100) → Email del cliente (chiave primaria)

nTel VARCHAR(20) → Numero di telefono del cliente

nome VARCHAR(50) → Nome del cliente

cognome VARCHAR(50) → Cognome del cliente

COMUNE

IDcomune INT → Identificativo univoco del comune

nome VARCHAR(50) → Nome del comune

INDIRIZZO

IDindirizzo INT → Identificativo dell'indirizzo

FK_Email VARCHAR(100) → Email del cliente associato

FK_IDcomune INT → Comune dell'indirizzo

ORDINE

IDordine INT → Identificativo univoco dell'ordine

stato VARCHAR(30) → Stato dell'ordine

FKEmail VARCHAR(100) → Cliente che effettua l'ordine

FKIDindirizzo INT → Indirizzo di consegna

RIDER

CF VARCHAR(16) → Codice fiscale del rider

nome VARCHAR(50) → Nome del rider

cognome VARCHAR(50) → Cognome del rider

iban VARCHAR(34) → IBAN del rider

ZONA

IDzona INT → Identificativo della zona di consegna

GIORNO

IDgiorno INT → Identificativo del giorno

nome VARCHAR(20) → Nome del giorno

orarioApertura TIME → Orario di apertura

orarioChiusura TIME → Orario di chiusura

RISTORANTE

IDristorante INT → Identificativo del ristorante

indirizzo VARCHAR(100) → Indirizzo del ristorante

PIATTO

IDpiatto INT → Identificativo del piatto

prezzo DECIMAL(6,2) → Prezzo del piatto

disponibilita BOOLEAN → Disponibilità del piatto

FK_IDristorante INT → Ristorante che offre il piatto

INGREDIENTE

IDingrediente INT → Identificativo dell'ingrediente

nome VARCHAR(50) → Nome dell'ingrediente

quantita VARCHAR(30) → Quantità dell'ingrediente

ALLERGENE

IDallergene INT → Identificativo dell'allergene

nome VARCHAR(50) → Nome dell'allergene

personalizzazioni VARCHAR(100) → Note o personalizzazioni

METODO_PAGAMENTO

IDmetodo INT → Identificativo del metodo di pagamento

CARTA

FK_IDmetodo INT → Metodo di pagamento associato

pagaInApp BOOLEAN → Pagamento effettuato tramite app

banca VARCHAR(50) → Banca della carta

Ncarta VARCHAR(20) → Numero della carta

RECENSIONE

IDrecensione INT → Identificativo della recensione

descrizione VARCHAR(255) → Testo della recensione

stelle INT → Valutazione da 1 a 5

FK_IDordine INT → Ordine recensitoù

PROMOZIONE

IDpromozione INT → Identificativo della promozione

dataInizio DATE → Data di inizio promozione

dataFine DATE → Data di fine promozione

nome VARCHAR(50) → Nome della promozione

ORDINE_RIDER

FK_IDordine INT → Ordine assegnato

FK_CF VARCHAR(16) → Rider assegnato

RIDER_ZONA

FK_CF VARCHAR(16) → Rider

FK_IDzona INT → Zona servita

ZONA_COMUNE

FK_IDzona INT → Zona

FK_IDcomune INT → Comune

ORDINE_PAGAMENTO

FK_IDordine INT → Ordine

FK_IDmetodo INT → Metodo di pagamento utilizzato

ORDINE_PROMOZIONE

FK_IDordine INT → Ordine

FK_IDpromozione INT → Promozione applicata

PROMOZIONE_PIATTO

FK_IDpromozione INT → Promozione

FK_IDpiatto INT → Piatto in promozione

PIATTO_ORDINE

FK_IDpiatto INT → Piatto ordinato

FK_IDordine INT → Ordine

INGREDIENTE_PIATTO

FK_IDingrediente INT → Ingrediente

FK_IDpiatto INT → Piatto

INGREDIENTE_ALLERGENE

FK_IDingrediente INT → Ingrediente

FK_IDallergene INT → Allergene

RISTORANTE_GIORNO

FK_IDristorante INT → Ristorante

FK_IDgiorno INT → Giorno di apertura

Conclusioni

La presente esercitazione ha esplorato in modo pratico i concetti chiave affrontati durante l'anno, mettendo in pratica le tecniche e gli strumenti appresi. Il progetto raggiunge gli obiettivi prefissati: analisi del problema, progettazione modulare, implementazione funzionale e verifica dei risultati. I risultati mostrano una soluzione stabile e documentata, facilmente estendibile per future migliorie. Tra i possibili sviluppi futuri si suggeriscono: estendere i casi di test, migliorare l'interfaccia utente e ottimizzare le performance là dove necessario.