UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI BASI DI DATI I

ANNO ACCADEMICO 2024/2025

Documentazione per

Progettazione Schema della Base di Dati

Ristrutturazione dello Schema secondo il modello relazionale

Schema Fisico

Autore:

Salvatore DE PASQUALE
MATRICOLA N86005746
salvat.depasquale@studenti.unina.it

Docente:

Prof.ssa Mara SANGIOVANNI

Indice

1 Introduzione

Questo documento descrive le fasi di progettazione della base di dati a supporto del sistema UninaFoodLab, una piattaforma per la gestione di corsi di cucina tematici. Vengono presentati lo Schema della Base di Dati, la ristrutturazione secondo il modello relazionale e lo Schema fisico. Questo documento fa parte del progetto BDD+OOP, riguardante i corsi di Basi Di Dati I e Programmazione Object-Oriented, a cura di Salvatore De Pasquale, studente del CdL in Informatica presso l'Università degli Studi di Napoli Federico II (gruppo OOBD84).

1.1 Traccia del Progetto

UninaFoodLab è un sistema per la gestione di corsi di cucina tematici. Gli chef possono registrare corsi su specifici argomenti (es. cucina asiatica, pasticceria, panificazione), specificando una data di inizio e una frequenza delle sessioni (es. settimanale, ogni due giorni). Ogni corso è articolato in più sessioni, che possono essere di due tipi: online, oppure in presenza, in cui gli utenti svolgono attività pratiche. Gli utenti possono iscriversi a più corsi e, nel caso delle sessioni pratiche, devono fornire una adesione esplicita per confermare la loro partecipazione. Ogni sessione pratica prevede la preparazione di una o più ricette, ciascuna delle quali richiede una specifica lista di ingredienti. Le adesioni vengono utilizzate per pianificare correttamente la quantità di ingredienti necessari, evitando cosi sprechi alimentari. Si utilizzino le proprie conoscenze del dominio per definire eventuali dettagli non specificati nella traccia.

1.2 Obiettivi

Gli obiettivi principali della base di dati sono:

- **Memorizzare** in modo strutturato tutte le informazioni relative a chef, utenti, corsi, sessioni, e ricette.
- **Definire** le relazioni tra le entità.
- **Garantire** integrità dei dati, attraverso l'uso di vincoli.
- **Supportare** le query necessarie al calcolo esatto delle quantità di ingredienti in merito al numero di partecipanti che confermano la presenza per ogni sessione pratica.

2 Progettazione schema della base di dati

Basandoci sulla traccia assegnata, definiamo le entità e le relazioni che le collegano. Questo ci porterà a definire lo schema del nostro database.

2.1 Entità Principali

- **Chef**: Chi tiene i corsi.
- **Utente**: Chi si iscrive ai corsi.
- **Corso**: L'argomento generale di studio (es. Pasticceria).
- **Sessione**: Una singola lezione di un corso.
- **Ricetta**: Una preparazione specifica insegnata in una sessione.
- **Ingrediente**: Un componente di una ricetta.

2.2 Relazioni e Cardinalità

- Uno Chef può creare molti Corsi. Un Corso è tenuto da un solo Chef (Relazione 1 a N).
- Un Utente può iscriversi a diversi Corsi, e un Corso può avere molti Utenti (Relazione N a N e conseguente entità derivata "Iscrizione").
- Un Corso si articola in più Sessioni. Ogni Sessione appartiene ad un solo Corso (Relazione 1 a N).
- Una Sessione è esattamente di un tipo: online o in presenza (Deduciamo che avrà una relazione 1 a 1 con una delle sotto-entità "SessioneOnline" e "SessioneInPresenza". Questa, ovviamente, risulta essere una *generalizzazione*, esclusiva e totale).
- Un Utente da l'adesione a più Sessioni in presenza, mentre una Sessione in presenza riceve l'adesione da più Utenti (Relazione N a N, deduciamo che abbiamo quindi l'entità derivata "Adesione").
- Una Sessione in presenza include la preparazione di molte Ricette, una Ricetta può far parte di più Sessioni (Relazione N a N, ne consegue quindi "ProgrammaSessione").
- Una Ricetta è composta da molti Ingredienti, un Ingrediente può essere usato in più Ricette (Relazione N a N, abbiamo quindi "ComposizioneRicetta" derivata).

2.3 Diagramma ER (Entità Relazione)

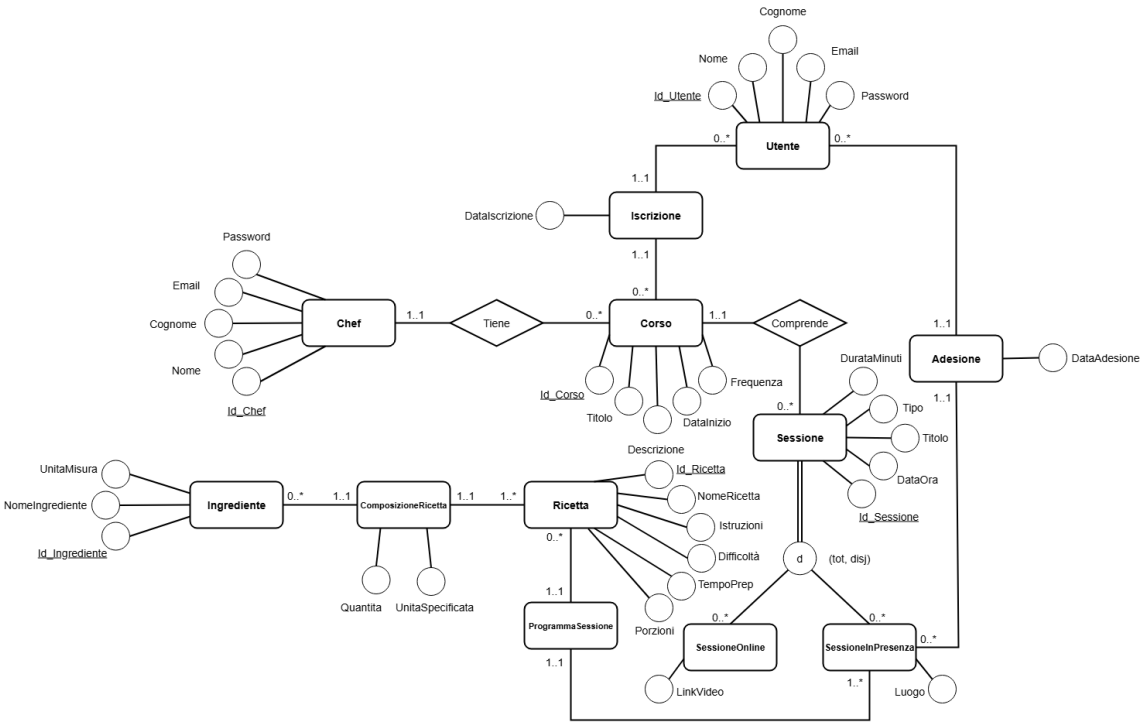

Figura 1: Diagramma Entità Relazione Esteso

Il diagramma mostrato in figura fornisce una visione astratta e formale delle informazioni, gestite dalle loro entità e relazioni che pongono le basi per la successiva progettazione logica e fisica della base di dati.

La modellazione ha permesso di identificare le seguenti entità principali:

- Entità anagrafiche, come Chef e Utente, che sono gli attori del sistema.
- Entità centrali, come Corso, Sessione, Ricetta, e Ingrediente.

Per descrivere in modo completo il dominio, sono state implementate le seguenti relazioni e gestioni di queste ultime:

- Relazioni Uno-a-Molti (1..N): Legami più diretti, come quelli, ad esempio, tra Chef e Corso, sono stati modellati specificando le relative cardinalità, secondo le regole comuni dei modelli ER.
- Relazioni Molti-a-Molti (N..M): Per gestire queste complesse relazioni sono state implementate le entità associative, come:
  - Iscrizione
  - ComposizioneRicetta
  - Adesione
  - ProgrammaSessione

  Ognuna con i propri eventuali attributi. Necessarie ai fini del corretto funzionamento logico del sistema.

- Generalizzazione/Specializzazione: L'entità Sessione è stata generalizzata, producendo due entità più specifiche: SessioneOnline e SessioneInPresenza. Questa scelta è dovuta, ovviamente, al fatto che una Sessione di un Corso può essere unicamente una delle due opzioni. La relazione è totale e disgiunta, significando che deve obbligatoriamente essere di uno e un solo tipo.

2.4 Diagramma Delle Classi UML

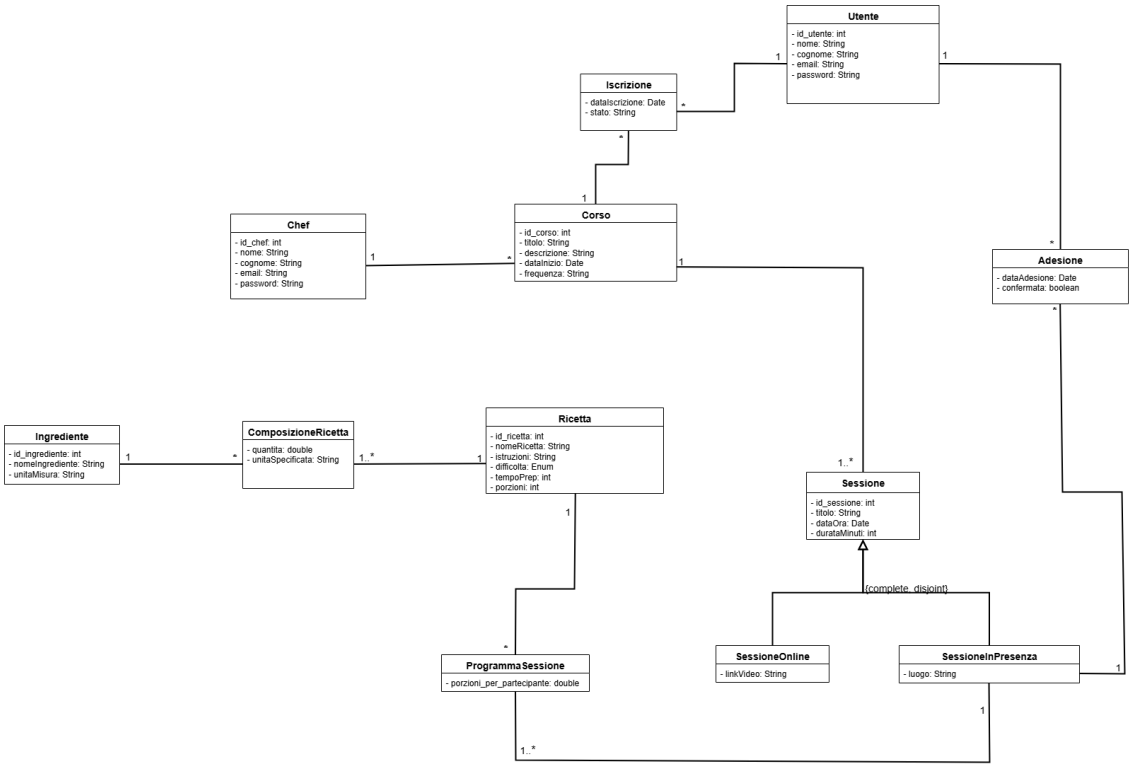

Figura 2: Diagramma delle Classi UML

Procediamo quindi adesso con la conversione del diagramma EER nel diagramma UML, una versione quindi meno vaga e più organizzata, che, una volta ristrutturato, sarà un ottimo blueprint per la progettazione vera e propria della base di dati.

Il modello EER è stato tradotto seguendo criteri specifici, privilegiando l'incapsulamento, l'ereditarietà, e una corretta rappresentazione delle associazioni tra gli oggetti.

Analizziamo quali regole e criteri sono stati seguiti per la corretta conversione:

- Entità e Classi: Anzitutto, ogni entità del modello è stata tradotta in classe. Le entità associative, inoltre, vengono rappresentate come classi vere e proprie, in quanto rappresentano concetti con propri attributi e comportamenti.

- Attributi e Incapsulamento: Gli attributi di ogni entità sono stati trasformati in attributi privati, con un tipo di dato specifico (es. String, int, Date). Questa scelta implementa il principio di incapsulamento, proteggendo i dati.

- Specializzazione e Ereditarietà: La generalizzazione dell'entità Sessione è stata tradotta correttamente utilizzando l'ereditarietà. La Classe "padre" Sessione nel dominio non esiste, una "sessione generica" non è una scelta possibile, ma deve piuttosto essere sempre online o in presenza.
  - Polimorfismo: Questo approccio ci consente di gestire le sessioni in maniera polimorfica. Ad esempio, la classe Corso potrebbe contenere una lista di sessioni e gestire gli oggetti al suo interno senza conoscere il loro tipo specifico, delegando la cosa alle sottoclassi.
  - Vincolo: il vincolo di generalizzazione scelto è "complete, disjoint", sempre per i motivi elencati sopra.

3 Ristrutturazione secondo il modello relazionale

Dopo aver definito il modello tramite il diagramma delle classi UML (versione concettuale), in questo capitolo si illustra il processo di ristrutturazione che porta alla versione UML ristrutturata, pronta per la traduzione nello schema logico relazionale.

L'obiettivo è di ridefinire il modello, perfezionandolo in fatto di compatibilità con il linguaggio SQL in modo tale da permetterci una conversione più semplice, veloce, e coerente.

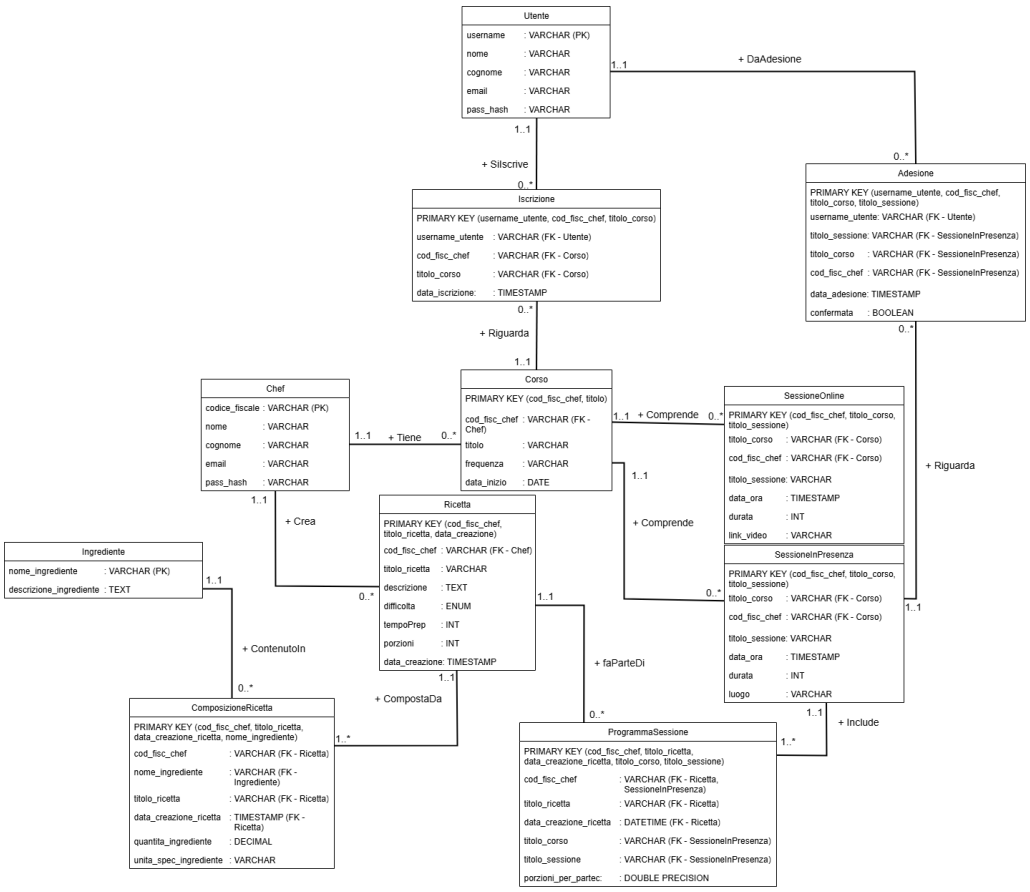

Figura 3: Diagramma Ristrutturato UML

3.1 Procedure effettuate sul diagramma

3.1.1 Conversione di tutte le classi in tabelle

La prima cosa fatta è stata adattare il diagramma in maniera tale che fosse compatibile con ciò che ci interessa, la costruzione della base di dati. Per questo motivo sono state convertite tutte le classi in tabelle con attributi SQL veri e propri.

3.1.2 Scelta di attributi coerenti con il linguaggio

Sono stati selezionati, invece che delle vaghe "String", come nel diagramma precedente, degli attributi coerenti con SQL: ogni tipo è stato selezionato in base a quale rispecchiasse meglio la natura del dato del caso.

3.1.3 Gestione delle relazioni 1:N

Le relazioni 1:N come `Chef-Corso` sono state gestite implementando delle chiavi esterne (FK).

3.1.4 Eliminazione della generalizzazione

Le generalizzazioni EER/UML sono costrutti non direttamente disponibili nel modello relazionale e quindi vanno sostituite con altre tecniche.

Per la gerarchia `Sessione {SessioneOnline, SessioneInPresenza}` si è scelta la strategia Concrete Table Inheritance: la superclasse `Sessione` è stata rimossa e gli attributi comuni sono stati duplicati nelle due tabelle concrete `SessioneOnline` e `SessioneInPresenza`. Questa scelta rende compatibile il diagramma con il linguaggio SQL e distingue i due tipi di sessioni, a costo della duplicazione controllata degli attributi comuni.

3.1.5 Scelta degli identificatori principali

Per ogni classe è stato definito un identificatore univoco (chiave primaria). Per alcune classi l'identificatore era singolo e naturale, per altre è composto da due o più attributi, sia appartenenti alla classe specifica, sia chiavi esterne che riconducono ad altre classi.

3.1.6 Creazione relazione Ricetta-Chef

Si è creata una nuova relazione "Crea" tra Ricetta e Chef, dato l'utilizzo in chiave primaria della chiave esterna del codice fiscale dello Chef da parte della classe Ricetta.

In conclusione, la ristrutturazione ha prodotto uno schema UML ristrutturato coerente con l'implementazione relazionale in MySQL: tutte le associazioni sono espresse tramite FK o tabelle ponte, le generalizzazioni sono state risolte tramite tabelle concrete e le chiavi primarie/esterne sono state definite in modo da preservare l'integrità referenziale e consentire operazioni di calcolo in modo efficiente.

3.2 Dizionario delle classi, delle associazioni, e dei vincoli

In questa sezione creeremo il dizionario delle classi: uno strumento utile a rendere chiaro, in maniera più verbale/testuale rispetto ai diagrammi visti finora, le classi della nostra base di dati, gli attributi che hanno, e le relazioni. Qui definiremo anche, per ogni classe, i vincoli di cui i loro attributi godono.

Classe Utente

Descrizione: Rappresenta un utente generico del sistema.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
username	VARCHAR	Identificativo univoco utente	PK
nome	VARCHAR	Nome dell'utente	Not null
cognome	VARCHAR	Cognome dell'utente	Not null
email	VARCHAR	Indirizzo email	Unique Not null
pass_hash	VARCHAR	Credenziali di accesso	Not null

Relazioni: - 0..* con *Iscrizione* - 0..* con *Adesione*

Classe Chef

Descrizione: Rappresenta un cuoco responsabile della preparazione delle ricette e della conduzione delle sessioni pratiche.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
codice_fiscale	VARCHAR	Identificativo univoco cuoco	PK
nome	VARCHAR	Nome del cuoco	Not null
cognome	VARCHAR	Cognome del cuoco	Not null
email	VARCHAR	Indirizzo email	Unique Not null
pass_hash	VARCHAR	Credenziali di accesso	Not null

Relazioni: - 0..* con *Ricetta* (uno Chef può creare più ricette) - 0..* con *Corso* (uno Chef può creare più corsi)

Classe Corso

Descrizione: Rappresenta un corso formativo creato da un determinato chef e composto da più sessioni.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
cod_fisc_chef	VARCHAR	Codice fiscale dello chef che tiene il corso	FK, Not null
titolo	VARCHAR	Titolo identificativo del corso	Not null
frequenza	VARCHAR	Frequenza delle lezioni (es. settimanale)	
data_inizio	DATE	Data di inizio del corso	Not null

Chiave primaria composta: (cod_fisc_chef, titolo)

Relazioni: - 1..1 con *Chef* (ogni corso è tenuto da un singolo chef) - 0..* con *SessioneOnline* e *SessioneInPresenza* (un corso comprende più sessioni) - 0..* con *Iscrizione* (gli utenti possono iscriversi a un corso)

Classe Ricetta

Descrizione: Rappresenta una ricetta creata da uno chef e legata a uno o più corsi tramite i programmi delle sessioni.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
cod_fisc_chef	VARCHAR	Codice fiscale chef	FK, Not null
titolo_ricetta	VARCHAR	Titolo della ricetta	Not null
data_creazione	TIMESTAMP	Data di creazione ricetta	Not null
descrizione	TEXT	descrizione della ricetta	
difficolta	ENUM	Livello di difficoltà	
tempoPrep	INT	Tempo di preparazione	
porzioni	INT	Numero di porzioni	

Chiave primaria composta: (cod_fisc_chef, titolo_ricetta, data_creazione)

Relazioni: - 1..1 con *Chef* (ogni ricetta è creata da un solo chef) - 1..* con *ComposizioneRicetta* (ingredienti che compongono la ricetta) - 0..* con *ProgrammaSessione* (la ricetta può essere inclusa in più sessioni)

Classe ComposizioneRicetta

Descrizione: Rappresenta l'associazione tra una ricetta e i suoi ingredienti, specificando quantità e unità di misura.

Attributo	Tipo	Descrizione	Vincoli
cod_fisc_chef	VARCHAR	Codice chef	FK, Not null
titolo_ricetta	VARCHAR	Titolo della ricetta	FK, Not null
data_creazione_ric	TIMESTAMP	Data di creazione ricetta	FK, Not null
nome_ingrediente	VARCHAR	Nome dell'ingrediente	FK, Not null
quantita	DECIMAL	Quantità dell'ingrediente	
unita_spec	VARCHAR	Unità di misura (es. g, ml, cucchiai)	

Chiave primaria composta: (cod_fisc_chef, titolo_ricetta, data_creazione_ricetta, nome_ingrediente)

Relazioni: - 1..1 con *Ricetta* (ogni composizione appartiene a una ricetta) - 1..1 con *Ingrediente* (ogni composizione fa riferimento a un ingrediente)

Classe Ingrediente

Descrizione: Rappresenta un singolo ingrediente utilizzabile nelle ricette, con relativa descrizione.

Attributo	Tipo	Descrizione	Vincoli
nome_ingrediente	VARCHAR	Nome univoco dell'ingrediente	PK
descrizione_ingrediente	TEXT	Descrizione dettagliata dell'ingrediente	

Relazioni: - 0..* con *ComposizioneRicetta* (un ingrediente può comparire in più ricette)

Classe ProgrammaSessione

Descrizione: Rappresenta l'associazione tra una sessione di un corso e le ricette trattate al suo interno, specificando le porzioni per partecipante.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
cod_fisc_chef	VARCHAR	Codice fiscale chef	FK, Not null
titolo_ricetta	VARCHAR	Titolo della ricetta	FK, Not null
data_creazione_ricetta	TIMESTAMP	Data di creazione ricetta	FK, Not null
titolo_corso	VARCHAR	Titolo del corso	FK, Not null
titolo_sessione	VARCHAR	Titolo della sessione	FK, Not null
porzioni_per_partec	DOUBLE PRECISION	Numero di porzioni per ogni partecipante	

Chiave primaria composta: (cod_fisc_chef, titolo_ricetta, data_creazione_ricetta, titolo_corso, titolo_sessione)

Relazioni: - 1..1 con *Ricetta* (ogni record associa una ricetta a una sessione) - 1..1 con *SessioneInPresenza* (ogni record appartiene a una specifica sessione in presenza)

Classe SessioneInPresenza

Descrizione: Rappresenta una singola sessione di un corso svolta in presenza, con indicazione di data, orario e luogo.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
titolo_corso	VARCHAR	Titolo del corso	FK, Not null
titolo_sessione	VARCHAR	Titolo della sessione	Not null
cod_fisc_chef	VARCHAR	Identificativo Chef	FK, Not null
data_ora	TIMESTAMP	Data e orario sessione	Not null
durata	INT	Durata sessione	
luogo	VARCHAR	Luogo della sessione	

Chiave primaria composta: (cod_fisc_chef, titolo_corso, titolo_sessione)

Relazioni: - 1..1 con *Corso* (ogni sessione appartiene a un corso) - 1..* con *ProgrammaSessione* (una sessione può includere più ricette) - 0..* con *Adesione* (una sessione può avere più partecipanti iscritti)

Classe SessioneOnline

Descrizione: Rappresenta una sessione di corso svolta online, con indicazione della piattaforma e delle credenziali di accesso.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
cod_fisc_chef	VARCHAR	Identificativo Chef	FK, Not null
titolo_corso	VARCHAR	Titolo del corso	FK, Not null
titolo_sessione	VARCHAR	Titolo della sessione	Not null
data_ora	TIMESTAMP	Data e orario della sessione	Not null
durata	INT	Durata sessione	
link_video	VARCHAR	Link del video sessione	

Chiave primaria composta: (cod_fisc_chef, titolo_corso, titolo_sessione)

Relazioni: - 1..1 con *Corso* (ogni sessione online appartiene a un corso)

Classe Iscrizione

Descrizione: Rappresenta l'iscrizione di un utente a un corso, con la relativa data di iscrizione.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
username_utente	VARCHAR	Identificativo dell'utente iscritto	FK, Not null
cod_fisc_chef	VARCHAR	Codice fiscale dello chef	FK, Not null
titolo_corso	VARCHAR	Titolo del corso a cui ci si iscrive	FK, Not null
data_iscrizione	TIMESTAMP	Data in cui l'utente si è iscritto	Not null

Chiave primaria composta: (cod_fisc_chef, username_utente, titolo_corso)

Relazioni: - 1..1 con *Utente* (ogni iscrizione è riferita a un utente) - 1..1 con *Corso* (ogni iscrizione è riferita a un corso)

Classe Adesione

Descrizione: Rappresenta la partecipazione effettiva di un utente a una specifica sessione (in presenza o online) di un corso a cui è iscritto.

Attributo	**Tipo**	**Descrizione**	**Vincoli**
username_utente	VARCHAR	Identificativo dell'utente	FK, Not null
titolo_corso	VARCHAR	Titolo del corso della sessione	FK, Not null
cod_fisc_chef	VARCHAR	Codice fiscale dello chef	FK, Not null
titolo_sessione	VARCHAR	Titolo della sessione	FK, Not null
data_adesione	TIMESTAMP	Data e orario dell'adesione	Not null
confermata	BOOLEAN	Conferma dell'adesione	

Chiave primaria composta: (username_utente, titolo_corso, cod_fisc_chef, titolo_sessione)

Relazioni: - 1..1 con *Utente* (ogni adesione è riferita a un utente) - 1..1 con *SessioneInPresenza* / *SessioneOnline* (ogni adesione è riferita a una specifica sessione di corso)

4 Schema Fisico

Entriamo quindi nelle ultime fasi della creazione di questo database, lo **schema fisico**. Per schema fisico intendiamo la base di dati completamente implementata in linguaggio SQL utilizzando un DBMS di nostra scelta. Inoltre, lo schema fisico comprende anche i vari *trigger*, *procedure*, e il *popolamento del database* stesso.

In questa sezione andremo quindi ad analizzare ogni fase di questo processo, dalla scelta del DBMS, la scrittura della base di dati, fino ad arrivare al suo popolamento e, di conseguenza, completamento, in modo da renderlo perfettamente compatibile e comunicante con l'applicativo Java. In questa maniera arriveremo a completare il progetto.

4.1 Scelta del DBMS e installazione

La scelta del DBMS da utilizzare per questa base di dati è ricaduta su **PostgreSQL**, la motivazione è da trovarsi nella sua semplicità e fedeltà agli standard SQL, soprattutto quando combinato con Java tramite driver JDBC.

In questo contesto ci limiteremo ad installare PostgreSQL senza andare poi ad utilizzare programmi come pgAdmin per gestirlo ed utilizzarlo, dato che per progettare tutta la base di dati utilizzeremo un mix tra *terminale CMD (Windows)*, e *Visual Studio Code*.

- Scarichiamo e installiamo PostgreSQL dal sito ufficiale.
- Scegliamo una password per il profilo utente che andremo ad utilizzare (postgres).
- Scegliamo la porta (default: 5432).

4.2 Creazione e connessione al database

Adesso è il momento di creare il vero e proprio database all'interno del DBMS. Lo facciamo molto semplicemente aprendo il CMD di Windows, collegandoci al server di PostgreSQL tramite path e credenziali, e utilizzando quindi SQL per creare la base di dati (possiamo conservare la creazione del db, tabelle, popolamento e molto altro in file ".sql" appositi):

```
1    -- ====================================================
2    -- File: create_database.sql
3    -- Descrizione: Crea il database del progetto
4    -- Autore: Salvatore De Pasquale
5    -- ====================================================
6
7    -- Crea il database
8    CREATE DATABASE UninaFoodLab;
```

Fatto ciò, ci connetteremo al database. Possiamo farlo tramite comando "\c UninaFoodLab".

4.3 Creazione dello schema del database

Andremo ora a creare lo schema della nostra base di dati, ossia tutte le classi, gli attributi, eventuali ENUM, e tutto ciò che riguarda la struttura del db. In sostanza questo è il momento di andare a definire in SQL tutto ciò che abbiamo progettato fino ad ora.

```
1   -- ====================================================
2   -- File: create_tables.sql
3   -- Descrizione: Definisce tutta la struttura della base di
        dati
4   -- Autore: Salvatore De Pasquale
5   -- ====================================================
6
7   BEGIN;
```

Il file inizia così, con una transazione. Ciò ci garantisce coerenza durante la creazione della base di dati, poiché una transazione definisce una serie di azioni che verranno eseguite **tutte**, oppure **nessuna**.

Andremo adesso a definire, nello stesso file, tutto ciò che ci serve.

4.3.1 Definizione di Chef

```
1       -- Definizione tabella Chef
2
3       CREATE TABLE Chef(
4           codice_fiscale VARCHAR (16) PRIMARY KEY,
5           nome VARCHAR(50) NOT NULL,
6           cognome VARCHAR(50) NOT NULL,
7           email VARCHAR(254) UNIQUE NOT NULL,
8           pass_hash VARCHAR(50) NOT NULL
9       );
```

4.3.2 Definizione di Corso

```
1       -- Definizione tabella Corso
2
3       CREATE TABLE Corso(
4           cod_fisc_chef VARCHAR(16) NOT NULL REFERENCES Chef(
                codice_fiscale) ON DELETE CASCADE,
5           titolo VARCHAR(50) NOT NULL,
6           frequenza VARCHAR(50),
7           data_inizio DATE NOT NULL,
8           PRIMARY KEY(cod_fisc_chef, titolo)
9       );
```

4.3.3 Definizione dell'ENUM stadi_difficolta

```
1         -- Creiamo l'ENUM per la tabella Ricetta
2
3         CREATE TYPE stadi_difficolta AS ENUM ('facile', 'medio',
              'difficile');
```

4.3.4 Definizione di Ricetta

```
1         -- Definizione tabella Ricetta
2
3         CREATE TABLE Ricetta(
4             cod_fisc_chef VARCHAR(16) NOT NULL REFERENCES Chef(
                  codice_fiscale) ON DELETE CASCADE,
5             titolo_ricetta VARCHAR(50) NOT NULL,
6             descrizione TEXT,
7             difficolta stadi_difficolta,
8             tempo_prep INT,
9             porzioni INT,
10            data_creazione TIMESTAMP NOT NULL DEFAULT
                  CURRENT_TIMESTAMP,
11            PRIMARY KEY (cod_fisc_chef, titolo_ricetta,
                  data_creazione)
12        );
```

4.3.5 Definizione di Utente

```
1         -- Definizione tabella Utente
2
3         CREATE TABLE Utente(
4             username VARCHAR(30) PRIMARY KEY,
5             nome VARCHAR(50) NOT NULL,
6             cognome VARCHAR(50) NOT NULL,
7             email VARCHAR(254) UNIQUE NOT NULL,
8             pass_hash VARCHAR(50) NOT NULL
9         );
```

4.3.6 Definizione di Ingrediente

```
1         -- Definizione tabella Ingrediente
2
3         CREATE TABLE Ingrediente(
4             nome_ingrediente VARCHAR(25) PRIMARY KEY,
5             descrizione_ingrediente TEXT
6         );
```

4.3.7 Definizione di SessioneOnline

```
1         -- Definizione SessioneOnline
2
3         CREATE TABLE SessioneOnline (
4             cod_fisc_chef VARCHAR(16) NOT NULL,
5             titolo_corso VARCHAR(50) NOT NULL,
6             titolo_sessione VARCHAR(50) NOT NULL,
7             data_ora TIMESTAMP NOT NULL,
8             durata INT,
9             link_video VARCHAR(2083),
10            PRIMARY KEY (cod_fisc_chef, titolo_corso,
                  titolo_sessione),
11            FOREIGN KEY (cod_fisc_chef, titolo_corso) REFERENCES
                  Corso(cod_fisc_chef, titolo) ON DELETE CASCADE
12        );
```

4.3.8 Definizione di SessioneInPresenza

```
1         -- Definizione SessioneInPresenza
2
3         CREATE TABLE SessioneInPresenza(
4             cod_fisc_chef VARCHAR(16) NOT NULL,
5             titolo_corso VARCHAR(50) NOT NULL,
6             titolo_sessione VARCHAR(50) NOT NULL,
7             data_ora TIMESTAMP NOT NULL,
8             durata INT,
9             luogo VARCHAR(100),
10            posti_totali INT,
11            posti_disponibili INT,
12            PRIMARY KEY (cod_fisc_chef, titolo_corso,
                  titolo_sessione),
13            FOREIGN KEY (cod_fisc_chef, titolo_corso) REFERENCES
                  Corso(cod_fisc_chef, titolo) ON DELETE CASCADE
14        );
```

4.3.9 Definizione di Iscrizione

```
1         -- Definizione Iscrizione
2
3         CREATE TABLE Iscrizione(
4             username_utente VARCHAR(30) NOT NULL,
5             cod_fisc_chef VARCHAR(16) NOT NULL,
6             titolo_corso VARCHAR(50) NOT NULL,
7             data_iscrizione TIMESTAMP NOT NULL DEFAULT
                  CURRENT_TIMESTAMP,
8             PRIMARY KEY (username_utente, cod_fisc_chef,
                  titolo_corso),
9             FOREIGN KEY (cod_fisc_chef, titolo_corso) REFERENCES
                  Corso(cod_fisc_chef, titolo) ON DELETE CASCADE,
10            FOREIGN KEY (username_utente) REFERENCES Utente(
                  username) ON DELETE CASCADE
11        );
```

4.3.10 Definizione di Adesione

```
1         -- Definizione Adesione
2
3         CREATE TABLE Adesione(
4             username_utente VARCHAR(30) NOT NULL,
5             titolo_corso VARCHAR(50) NOT NULL,
6             titolo_sessione VARCHAR(50) NOT NULL,
7             cod_fisc_chef VARCHAR(16) NOT NULL,
8             data_adesione TIMESTAMP NOT NULL DEFAULT
                  CURRENT_TIMESTAMP,
9             confermata BOOLEAN,
10            PRIMARY KEY (username_utente, cod_fisc_chef,
                  titolo_corso, titolo_sessione),
11            FOREIGN KEY (username_utente) REFERENCES Utente(
                  username) ON DELETE CASCADE,
12            FOREIGN KEY (cod_fisc_chef, titolo_corso,
                  titolo_sessione) REFERENCES SessioneInPresenza(
                  cod_fisc_chef, titolo_corso, titolo_sessione) ON
                  DELETE CASCADE
13        );
```

4.3.11 Definizione di ProgrammaSessione

```
1         -- Definizione ProgrammaSessione
2
3         CREATE TABLE ProgrammaSessione(
4             cod_fisc_chef VARCHAR(16) NOT NULL,
5             titolo_ricetta VARCHAR(50) NOT NULL,
6             data_creazione_ricetta TIMESTAMP NOT NULL,
7             titolo_corso VARCHAR(50) NOT NULL,
8             titolo_sessione VARCHAR(50) NOT NULL,
9             porzioni_per_partec DOUBLE PRECISION,
10            PRIMARY KEY (cod_fisc_chef, titolo_ricetta,
                  data_creazione_ricetta, titolo_corso,
                  titolo_sessione),
11            FOREIGN KEY (cod_fisc_chef, titolo_ricetta,
                  data_creazione_ricetta) REFERENCES Ricetta(
                  cod_fisc_chef, titolo_ricetta, data_creazione) ON
                  DELETE CASCADE,
12            FOREIGN KEY (cod_fisc_chef, titolo_corso,
                  titolo_sessione) REFERENCES SessioneInPresenza(
                  cod_fisc_chef, titolo_corso, titolo_sessione) ON
                  DELETE CASCADE
13        );
```

4.3.12 Definizione di ComposizioneRicetta

```
1         -- Definizione di ComposizioneRicetta
2
3         CREATE TABLE ComposizioneRicetta(
4             cod_fisc_chef VARCHAR(16) NOT NULL,
5             nome_ingrediente VARCHAR(25) NOT NULL,
6             titolo_ricetta VARCHAR(50) NOT NULL,
7             data_creazione_ricetta TIMESTAMP NOT NULL,
8             quantita_ingrediente DECIMAL(6,2),
9             unita_spec_ingrediente VARCHAR(10),
10            PRIMARY KEY (cod_fisc_chef, titolo_ricetta,
                 data_creazione_ricetta, nome_ingrediente),
11            FOREIGN KEY (cod_fisc_chef, titolo_ricetta,
                 data_creazione_ricetta) REFERENCES Ricetta(
                 cod_fisc_chef, titolo_ricetta, data_creazione) ON
                 DELETE CASCADE,
12            FOREIGN KEY (nome_ingrediente) REFERENCES Ingrediente
                 (nome_ingrediente) ON DELETE RESTRICT
13        );
```

Chiudiamo ora il file con:

```
1
2         COMMIT;
```

In modo da completare la preparazione della transazione.

4.4 Popolamento del database

Andremo adesso a popolare il database con dei dati che utilizzeremo per testare l'applicativo e le varie funzionalità che SQL ci fornisce per manipolare i dati.

Iniziamo il file "populate_database.sql" così:

```
1     -- ================================================
2     -- File: populate_database.sql
3     -- Descrizione: Popola il database UninaFoodLab con dati
4     -- Autore: Salvatore De Pasquale
5     -- ================================================
6
7     BEGIN;
```

4.4.1 Inserimento Chef

```
1         - -------------------------
2         -- TABELLA CHEF
3         -- -------------------------
4         INSERT INTO Chef (codice_fiscale, nome, cognome, email,
              pass_hash)
5         VALUES
6         ('RSSMRA85M01H501U', 'Mario', 'Rossi', 'mario.
              rossi@example.com', 'hashpassword123'),
7         ('BNCLRA90L45F839K', 'Lara', 'Bianchi', 'lara.
              bianchi@example.com', 'hashpassword456');
```

4.4.2 Inserimento Corso

```
1         -- -------------------------
2         -- TABELLA CORSO
3         -- -------------------------
4         INSERT INTO Corso (cod_fisc_chef, titolo, frequenza,
              data_inizio)
5         VALUES
6         ('RSSMRA85M01H501U', 'Cucina Italiana', 'settimanale', '
              2025-10-20'),
7         ('BNCLRA90L45F839K', 'Dolci al Cucchiaio', 'mensile', '
              2025-11-05');
```

4.4.3 Inserimento Utente

```
1         -- -------------------------
2         -- TABELLA UTENTE
3         -- -------------------------
4         INSERT INTO Utente (username, nome, cognome, email,
              pass_hash)
5         VALUES
6         ('giulia123', 'Giulia', 'Bianchi', 'giulia.
              bianchi@example.com', 'hashpass456'),
7         ('andrea88', 'Andrea', 'Verdi', 'andrea.verdi@example.com
              ', 'hashpass789');
```

4.4.4 Inserimento Ingrediente

```
1         -- -------------------------
2         -- TABELLA INGREDIENTE
3         -- -------------------------
4         INSERT INTO Ingrediente (nome_ingrediente,
              descrizione_ingrediente)
5         VALUES
6         ('Pomodoro', 'Pomodoro fresco maturo'),
7         ('Basilico', 'Foglie di basilico fresco'),
8         ('Zucchero', 'Zucchero semolato'),
9         ('Latte', 'Latte intero fresco');
```

4.4.5 Inserimento Ricetta

```
-- -------------------------
-- TABELLA RICETTA
-- -------------------------
INSERT INTO Ricetta (cod_fisc_chef, titolo_ricetta,
    descrizione, difficolta, tempo_prep, porzioni,
    data_creazione)
VALUES
('RSSMRA85M01H501U', 'Pasta al Pomodoro', 'Ricetta
    tradizionale italiana', 'facile', 30, 4, '2025-10-15
    18:00:00'),
('BNCLRA90L45F839K', 'Panna Cotta', 'Dolce al cucchiaio
    con panna e zucchero', 'medio', 45, 6, '2025-10-14
    10:30:00');
```

4.4.6 Inserimento SessioneInPresenza

```
-- -------------------------
-- TABELLA SESSIONE IN PRESENZA
-- -------------------------
INSERT INTO SessioneInPresenza (cod_fisc_chef,
    titolo_corso, titolo_sessione, data_ora, durata, luogo
    )
VALUES
('RSSMRA85M01H501U', 'Cucina Italiana', 'Lezione 1', '
    2025-10-21 15:30:00', 120, 'Napoli'),
('BNCLRA90L45F839K', 'Dolci al Cucchiaio', 'Lezione 1', '
    2025-11-07 17:00:00', 90, 'Milano');
```

4.4.7 Inserimento SessioneOnline

```
-- -------------------------
-- TABELLA SESSIONE ONLINE
-- -------------------------
INSERT INTO SessioneOnline (cod_fisc_chef, titolo_corso,
    titolo_sessione, data_ora, durata, link_video)
VALUES
('RSSMRA85M01H501U', 'Cucina Italiana', 'Lezione Online 1
    ', '2025-10-23 18:00:00', 90, 'https://zoom.example.
    com/mario'),
('BNCLRA90L45F839K', 'Dolci al Cucchiaio', 'Lezione
    Online 1', '2025-11-09 20:30:00', 60, 'https://zoom.
    example.com/lara');
```

4.4.8 Inserimento Iscrizione

```
1         -- -------------------------
2         -- TABELLA ISCRIZIONE
3         -- -------------------------
4         INSERT INTO Iscrizione (username_utente, cod_fisc_chef,
              titolo_corso, data_iscrizione)
5         VALUES
6         ('giulia123', 'RSSMRA85M01H501U', 'Cucina Italiana', '
              2025-10-15 18:10:00'),
7         ('andrea88', 'BNCLRA90L45F839K', 'Dolci al Cucchiaio', '
              2025-10-16 09:45:00');
```

4.4.9 Inserimento Adesione

```
1         -- -------------------------
2         -- TABELLA ADESIONE
3         -- -------------------------
4         INSERT INTO Adesione (username_utente, titolo_corso,
              titolo_sessione, cod_fisc_chef, data_adesione,
              confermata)
5         VALUES
6         ('giulia123', 'Cucina Italiana', 'Lezione 1', '
              RSSMRA85M01H501U', '2025-10-17 10:00:00', TRUE),
7         ('andrea88', 'Dolci al Cucchiaio', 'Lezione 1', '
              BNCLRA90L45F839K', '2025-10-18 11:00:00', FALSE);
```

4.4.10 Inserimento ComposizioneRicetta

```
1         -- -------------------------
2         -- TABELLA COMPOSIZIONE RICETTA
3         -- -------------------------
4         INSERT INTO ComposizioneRicetta (cod_fisc_chef,
              nome_ingrediente, titolo_ricetta,
              data_creazione_ricetta, quantita_ingrediente,
              unita_spec_ingrediente)
5         VALUES
6         ('RSSMRA85M01H501U', 'Pomodoro', 'Pasta al Pomodoro', '
              2025-10-15 18:00:00', 200.0, 'g'),
7         ('RSSMRA85M01H501U', 'Basilico', 'Pasta al Pomodoro', '
              2025-10-15 18:00:00', 5.0, 'foglie'),
8         ('BNCLRA90L45F839K', 'Latte', 'Panna Cotta', '2025-10-14
              10:30:00', 500.0, 'ml'),
9         ('BNCLRA90L45F839K', 'Zucchero', 'Panna Cotta', '
              2025-10-14 10:30:00', 100.0, 'g');
```

4.4.11 Inserimento ProgrammaSessione

```
1         -- -------------------------
2         -- TABELLA PROGRAMMA SESSIONE
3         -- -------------------------
4         INSERT INTO ProgrammaSessione (cod_fisc_chef,
              titolo_ricetta, data_creazione_ricetta, titolo_corso,
              titolo_sessione, porzioni_per_partec)
5         VALUES
6         ('RSSMRA85M01H501U', 'Pasta al Pomodoro', '2025-10-15
              18:00:00', 'Cucina Italiana', 'Lezione 1', 2.0),
7         ('BNCLRA90L45F839K', 'Panna Cotta', '2025-10-14 10:30:00'
              , 'Dolci al Cucchiaio', 'Lezione 1', 1.5);
8
9         COMMIT;
```

4.5 Creazione Trigger e Funzioni

Adesso andremo a creare funzioni e trigger comode per il funzionamento corretto e scorrevole dell'applicativo.

4.5.1 Funzione aggiorna_posti_disponibili()

La funzione serve ad aggiornare automaticamente il numero di posti disponibili in una SessioneInPresenza, ogni volta che cambia qualcosa nelle adesioni (ad esempio, quando un utente conferma o annulla la sua partecipazione).

```
1         -- Funzione che aggiorna i posti disponibili
2         CREATE OR REPLACE FUNCTION aggiorna_posti_disponibili()
3         RETURNS TRIGGER AS $$
4         DECLARE
5             -- Variabile per memorizzare i posti totali della
                  sessione
6             v_posti_totali INT;
7         BEGIN
8
9             -- Caso 1: Aggiornamento da (FALSE a TRUE) -> L'
                  utente conferma
10            -- --> DECREMENTA posti disponibili
11            IF (TG_OP = 'UPDATE' AND OLD.confermata = FALSE AND
                  NEW.confermata = TRUE) THEN
12
13                UPDATE SessioneInPresenza
14                SET posti_disponibili = posti_disponibili - 1
15                WHERE cod_fisc_chef = NEW.cod_fisc_chef
16                  AND titolo_corso = NEW.titolo_corso
17                  AND titolo_sessione = NEW.titolo_sessione
18                  AND posti_disponibili > 0; -- Sicurezza: non
                        scendere sotto 0
19
20            -- Caso 2: Aggiornamento da (TRUE a FALSE) -> L'
                  utente annulla la conferma
21            -- --> INCREMENTA posti disponibili
22            ELSIF (TG_OP = 'UPDATE' AND OLD.confermata = TRUE AND
                      NEW.confermata = FALSE) THEN
```

```
23
24                  -- Recupero i posti totali per il controllo di
                        sicurezza
25                  SELECT posti_totali INTO v_posti_totali
26                  FROM SessioneInPresenza
27                  WHERE cod_fisc_chef = OLD.cod_fisc_chef
28                    AND titolo_corso = OLD.titolo_corso
29                    AND titolo_sessione = OLD.titolo_sessione;
30
31                  UPDATE SessioneInPresenza
32                  SET posti_disponibili = posti_disponibili + 1
33                  WHERE cod_fisc_chef = OLD.cod_fisc_chef
34                    AND titolo_corso = OLD.titolo_corso
35                    AND titolo_sessione = OLD.titolo_sessione
36                    AND posti_disponibili < v_posti_totali; --
                          Sicurezza: non superare i posti totali
37
38              END IF;
39
40              RETURN NEW;
41
42          END;
43          $$ LANGUAGE plpgsql;
```

4.5.2 Trigger trigger_aggiorna_posti

Il trigger serve a far rispettare una regola di business automatica: prima di inserire una nuova adesione, controlla (tramite la funzione) se il numero massimo di partecipanti alla sessione è stato raggiunto.

```
1           -- Trigger su inserimento e aggiornamento nella tabella
               Adesione
2           CREATE TRIGGER trigger_aggiorna_posti
3           AFTER INSERT OR UPDATE OF confermata ON Adesione
4           FOR EACH ROW
5           EXECUTE FUNCTION aggiorna_posti_disponibili();
```