# Relazione Applicazioni Web: Linguaggi e Architetture

Componenti del gruppo:

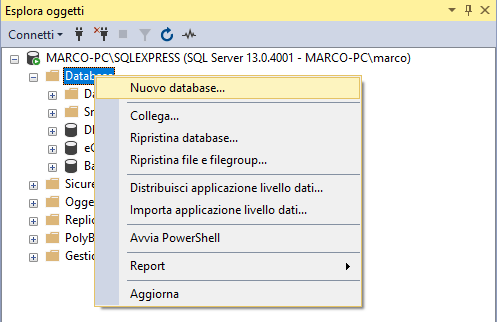
-Uejdis Dushi: 20010884;

-Luca Franciscone: 20010057;

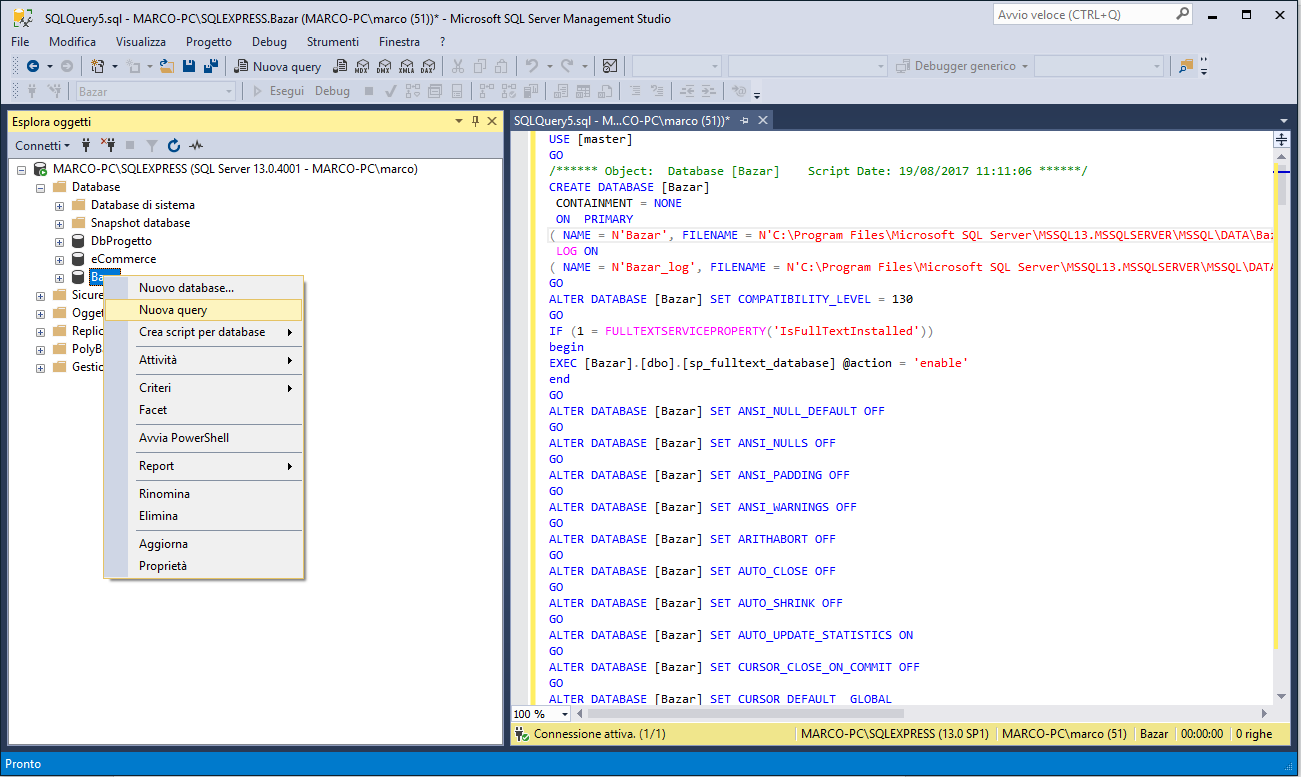
-Marco Viola: 20010208.

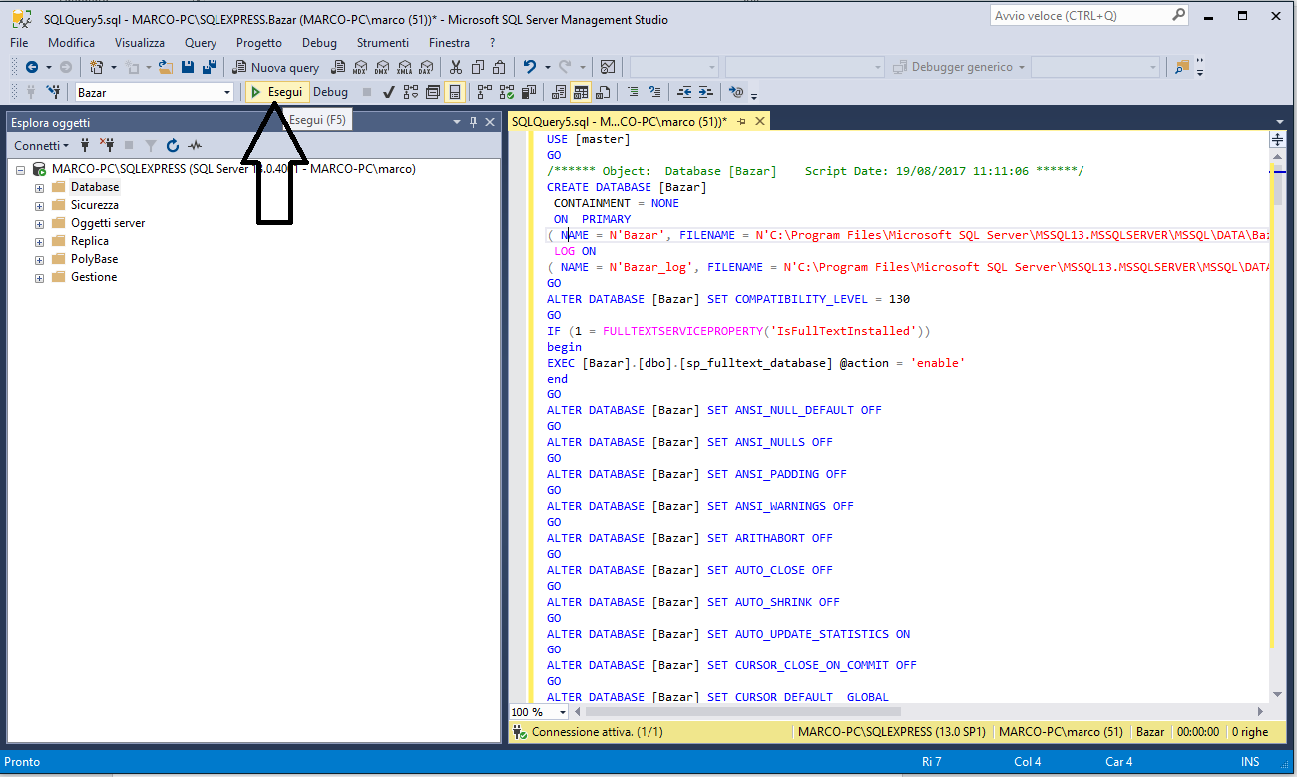
# Istruzioni per l’avvio

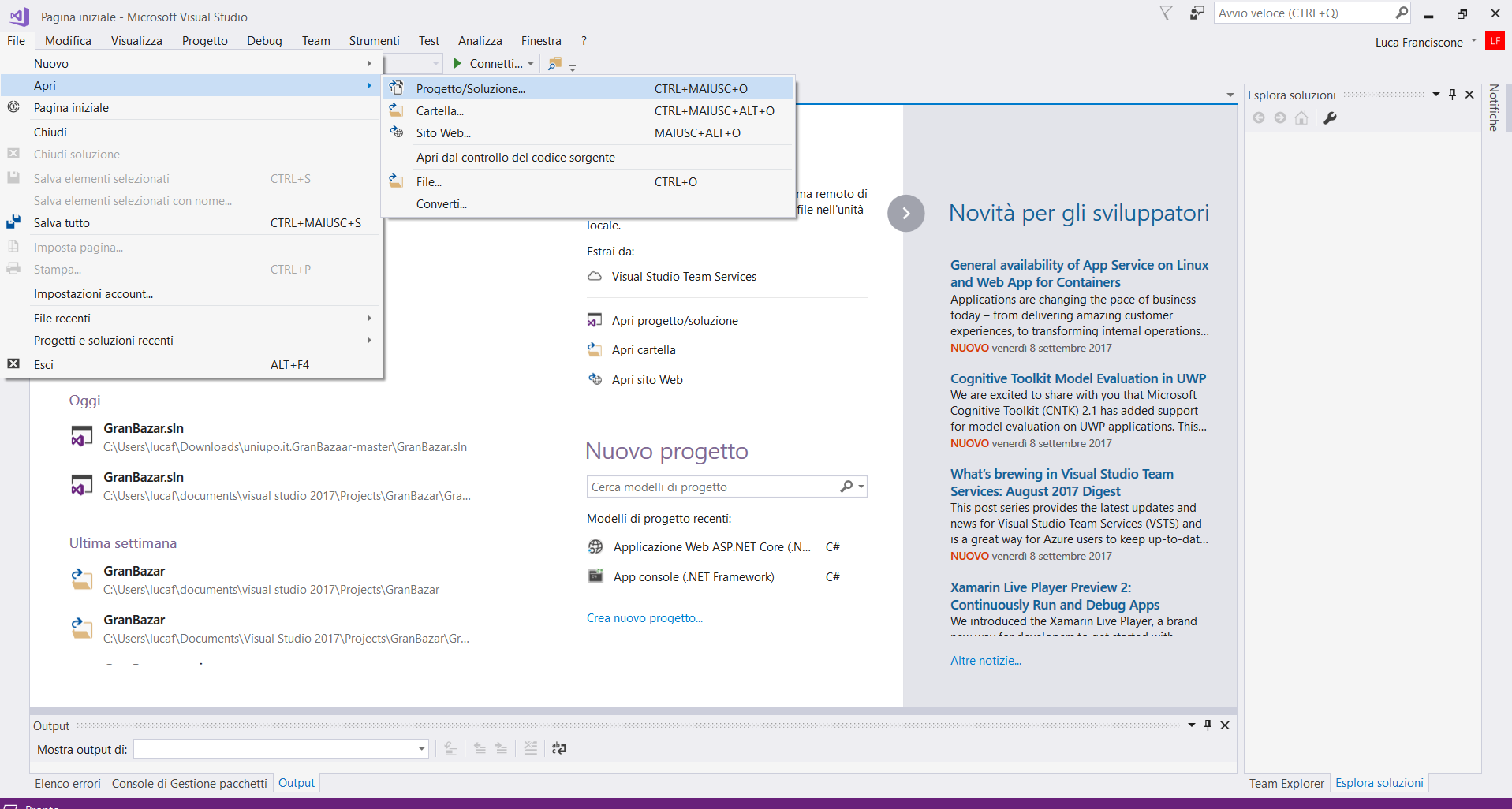
1. Scaricare il progetto;
2. Scompattarlo;
3. Aprire Microsoft SQL Server Manager Studio;
4. Creare nuovo database con nome “Bazar” -> Tasto destro su “Database” poi “Nuovo database”



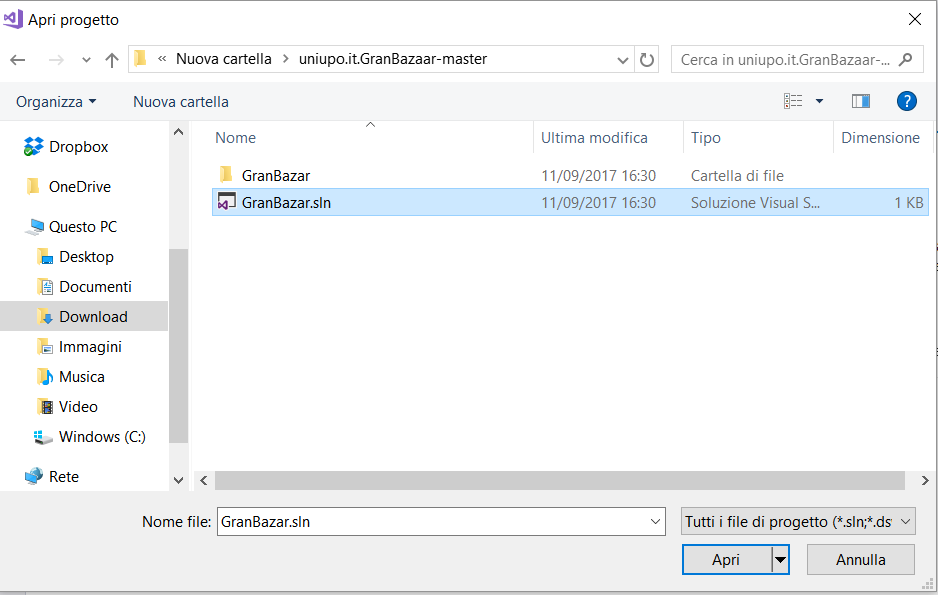
1. Aprire con Notepad++ il file “Bazar\uniupo.it.GranBazaar\GranBazar\Sql\DDL-DML” poi selezionare tutto e fare copia;
2. Eseguire query di creazione su database bazar -> tasto destro sul database “Bazar” poi “Nuova query”



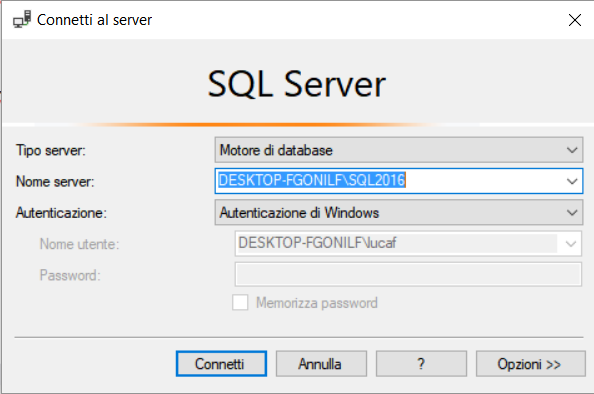
1. Incollare nella finestra appena aperta il testo prima copiato;
2. Premere il tasto Esegui;
3. Aprire Visual Studio, File🡪Apri🡪Progetto/Soluzione

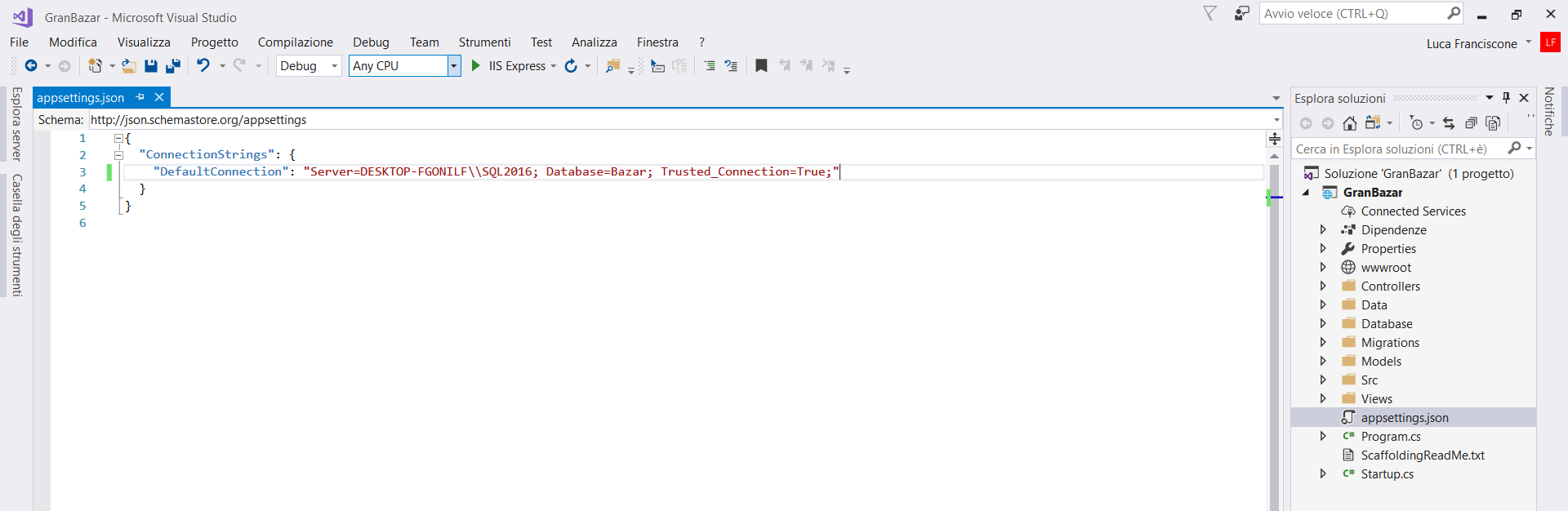


1. Selezionare il file GranBazar.sln e cliccare su “Apri”



1. All’interno del file “appsetting.json” in “Connection String” settare il nome del server e nome del database da utilizzare, nel nostro caso “Bazar” e la connection string recuperata da “Sql Server Management Studio”





Per il testare il progetto nel DB sono state inserite delle utenze di prova con le seguenti credenziali:

* Amministratore:
  + Email: [admin@email.com](mailto:admin@email.com)
  + Password: Admin2017.
* Utente:
  + Email: [user1@email.com](mailto:user1@email.com)
  + Password: User2017.

# Scelte implementative

* Si è deciso di utilizzare AspNetAuthentication per permettere di proteggere l’accessibilità a una o più action esposte dal controller mediante l’utilizzo del tag [Authorize]. Allo stesso tempo abbiamo optato di utilizzare una tabella utenti per avere un maggiore controllo sui dati inseriti. Le tabelle hanno copia speculare del contenuto delle utenze;
* Si è deciso di mantenere il carrello in sessione senza creare tabelle temporanee all’interno del database; il carrello ha una validità di 10 minuti limitata alla sessione corrente, trascorso il tempo viene cancellato il suo contenuto automaticamente.
* È stato implementato l’extension methods “SessionExtensions” che permette di aggiungere funzionalità avanzate per il passaggio di oggetti complessi ai metodi Get, Set alla classe HttpContext.Session;
* Per la realizzazione di tabelle, griglie e contenuto filtrato abbiamo scelto di utilizzare le funzionalità offerte da Bootstrap per una maggiore personalizzazione;
* È stato creato un modello VistaProdottoOrdine che raggruppa tutti i campi d’interesse per il passaggio dei contenuti degli ordini alle viste interessate;
* All’interno della tabella Prodotto sono state create 3 colonne con la funzionalità di memorizzare la locazione nella memoria delle immagini per i prodotti;
* Si è scelto di reindirizzare l’utente dopo il login alla medesima pagina in cui si trovava in precedenza, in caso di errore di autenticazione viene restituito un messaggio di errore;
* Si è scelto di privare l’account amministratore dalla possibilità di effettuare ordini;
* Si è scelto di creare una pagina di errore che viene restituita dall’handler globale quando si tenta di accedere ad una pagina inesistente per evitare codice duplicato.

# Descrizione Modello a Oggetti

# Folder Controllers

All’interno della cartella Controller sono stati inseriti tutti i controller utilizzati per l’implementazione del nostro progetto:

* **AccountController**: All’interno della classe sono contenuti tutti i metodi utilizzabili per la gestione delle utenze, quali la creazione di una nuova utenza, metodi per la gestione del login e del logout, metodi adibiti all’area di amministrazione e all’area personale dell’utente;
* **CarrelloController**: All’interno della classe sono presenti tutti i metodi adibiti all’utilizzo del carrello, come stampa del contenuto del carrello e i metodi per inserimento, aggiornamento e modifica del contenuto del medesimo;
* **CheckoutController**: Gestisce l’inserimento di un nuovo ordine all’interno del database partendo dal carrello in sessione;
* **HomeController**: Gestisce la Top10 dei prodotti acquistati negli ultimi 30 giorni e mostra il catalogo completo dei prodotti;
* **ProdottiController**: Gestisce la restituzione della scheda prodotto a partire dall’id prodotto passato come parametro;
* **RicercaController**: Permette di cercare contenuto per titolo e descrizione prodotto;

# Folder Models

Le classi **Ordine, Prodotto, Utente, Ordine\_Prodotto** rappresentano i Model del contesto. È stato creato anche un Model **VistaProdottoOrdine** che raggruppa tutti i campi d’interesse per il passaggio dei contenuti degli ordini alle viste interessate.

# Folder Src

Contiene la classe SessionExtensions che permette di aggiungere funzionalità avanzate per il passaggio di oggetti complessi ai metodi Get, Set alla classe HttpContext.Session. Gli oggetti complessi vengono serializzati e deserializzati in modo da poterli salvare in sessione, il formato utilizzato è il Json e per effettuare la serializzazione e serializzazione è stata utilizzata la libreria esterna Newtonsoft.Json.

Per eventuali dubbi si faccia riferimento ai commenti nel codice, in quanto le parti più complesse sono state commentate.

# Schema E-R

Il database è composto da solo 3 entità: Utente, Ordine e Prodotto. I Models presenti nel progetto sono 4, oltre alle 3 entità appena citate la tabella Ordine\_Prodotto è il risultato di una relazione molti a molti tra l’entità Ordine e l’entità Prodotto. Vedere lo schema sottostante.

