Appunti POMERIGGIO 11-02-26

Nel contesto di un'applicazione per la gestione di ristoranti, la modellazione delle relazioni tra entità rappresenta un passaggio fondamentale per garantire coerenza dei dati e prestazioni ottimali. La relazione tra città e ristorante è di tipo uno a molti bidirezionale, dove una città può contenere numerosi ristoranti ma ogni ristorante appartiene necessariamente a una sola città. Questa bidirezionalità consente di navigare la relazione in entrambe le direzioni, permettendo di accedere ai ristoranti di una città e di risalire dalla singola attività alla città che la ospita.

La strategia di caricamento dei dati associati può essere configurata come eager o lazy, scelta che dipende strettamente dal caso d'uso specifico. Considerando uno scenario reale in cui un utente apre l'applicazione e si trova nella propria città, il sistema deve essere in grado di visualizzare immediatamente i ristoranti disponibili nella zona. Ogni ristorante è caratterizzato da attributi essenziali quali nome, email, password, indirizzo e il riferimento alla città di appartenenza. La struttura relazionale si estende ulteriormente considerando che ogni ristorante può servire molteplici piatti, mentre ogni piatto è associato esclusivamente a un singolo ristorante. Anche in questo caso la bidirezionalità permette di accedere ai piatti di un ristorante e di risalire dal piatto al ristorante che lo offre.

L'architettura del sistema prevede l'utilizzo di oggetti di trasferimento dati, i DTO, che rappresentano una versione semplificata delle entità destinate alla comunicazione tra strati applicativi. L'obiettivo principale è ridurre il numero di richieste al database, poiché la migliore ottimizzazione consiste proprio nel minimizzare le chiamate remote. Per questo motivo il DTO della città non rappresenta semplicemente una copia dell'entità originale, ma viene arricchito per includere anche i ristoranti e i relativi piatti, aggregando così tutte le informazioni necessarie in un'unica struttura.

La definizione fisica dell'entità Restaurant nel progetto richiede l'utilizzo di annotazioni JPA specifiche. L'annotazione JoinColumn specifica quale colonna della tabella corrente rappresenta il collegamento alla chiave primaria dell'entità correlata, permettendo al framework di gestire automaticamente le relazioni durante le operazioni di persistenza. Questa annotazione viene impiegata in

combinazione con altre come ManyToOne, OneToMany, OneToOne e ManyToMany per definire la chiave esterna che realizza il join tra le tabelle.

Dopo aver definito l'entità è necessario creare il repository corrispondente, componente fondamentale per la permanenza dei dati. Il RestaurantRepository estende JpaRepository, ereditando così tutte le operazioni CRUD di base senza necessità di implementazione manuale. La progettazione delle API rappresenta invece l'ultimo passo, poiché costituisce il punto di contatto con il mondo esterno.

Il RestaurantDTO viene progettato includendo gli attributi identificativi del ristorante e un riferimento all'oggetto CityDTO. Questa scelta evita caricamenti circolari e mantiene la struttura leggera, includendo solo le informazioni essenziali della città senza ricaricare l'intero grafo di oggetti. Il mapper associato si occupa della conversione bidirezionale tra entità e DTO. Nella conversione da DTO a entità, se è presente un identificativo di città valido, il sistema interroga il repository per recuperare l'entità completa e associarla al ristorante. È possibile estendere questa logica per cercare la città anche tramite nome qualora l'identificativo non sia disponibile. Nella direzione opposta, dalla entità al DTO, vengono estratti solo gli attributi necessari della città, popolando un nuovo oggetto CityDTO con identificativo, nome e provincia.

La gestione della sicurezza richiede particolare attenzione nel trattamento delle password. Durante la conversione verso il DTO la password non viene mai inclusa, garantendo che informazioni sensibili non vengano esposte attraverso le API. Al contrario, durante la ricezione di dati dal client, la password viene acquisita ma immediatamente sottoposta a hashing prima della persistenza. Per questo scopo è necessario aggiungere al file di configurazione Maven la dipendenza commons-codec, libreria che fornisce implementazioni di algoritmi di hashing. Il file pom.xml, scritto in XML che è un metalinguaggio utilizzato per definire altri linguaggi simili a HTML, permette di gestire le dipendenze del progetto che vengono automaticamente scaricate da Maven.

L'implementazione dell'API REST per la creazione di nuovi ristoranti prevede l'utilizzo dell'annotazione RestController e la definizione del percorso base tramite RequestMapping. Il metodo dedicato alla registrazione riceve un DTO tramite RequestBody, lo converte nell'entità corrispondente utilizzando il mapper, procede con l'hashing della password attraverso un componente dedicato chiamato MD5Hasher, salva l'entità nel database e infine restituisce il DTO aggiornato al

client. Questo flusso garantisce che la password non venga mai memorizzata in chiaro e che il client riceva una rappresentazione sicura dell'oggetto appena creato. Rimane da implementare un endpoint GET per recuperare un ristorante specifico tramite il suo identificativo, completando così le operazioni CRUD di base per questa entità.