Documentazione recupero singolo ordine

Consumer

OrdineController

GetOrdineDettaglio

ManageOrdineBusiness

GetOrdineDettaglioAsync

ManageOrdineData

GetOrdineDettaglioAsync

RepositoryOrdine

GetOrdineDettaglioAsync

IRepositoryOrdine

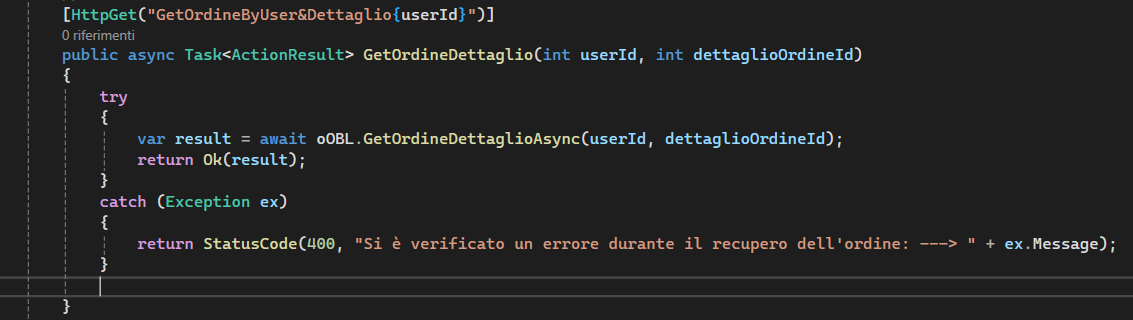
GetOrdineDettaglioAsync

AcademyShopDB

-Ordine; -Prodotto;

-Stato\_Ordine; -Dettagli\_Ordine;

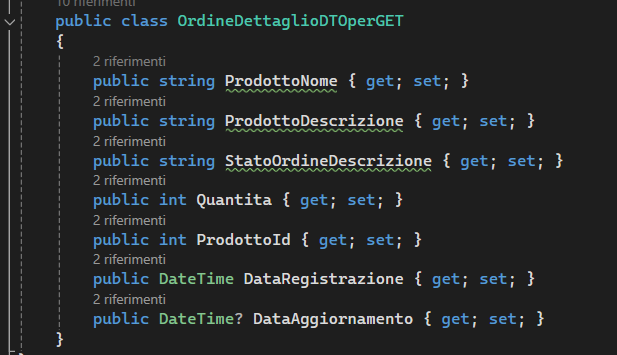
Api Controller: (oOBL = object Ordini Business Layer)



L'endpoint è un metodo HTTP GET che accetta due parametri nella sua URL: "userId" e "dettaglioOrdineId". Quando viene chiamato, il metodo cerca di recuperare un ordine dettagliato associato all'ID dell'utente e all'ID del dettaglio dell'ordine forniti.

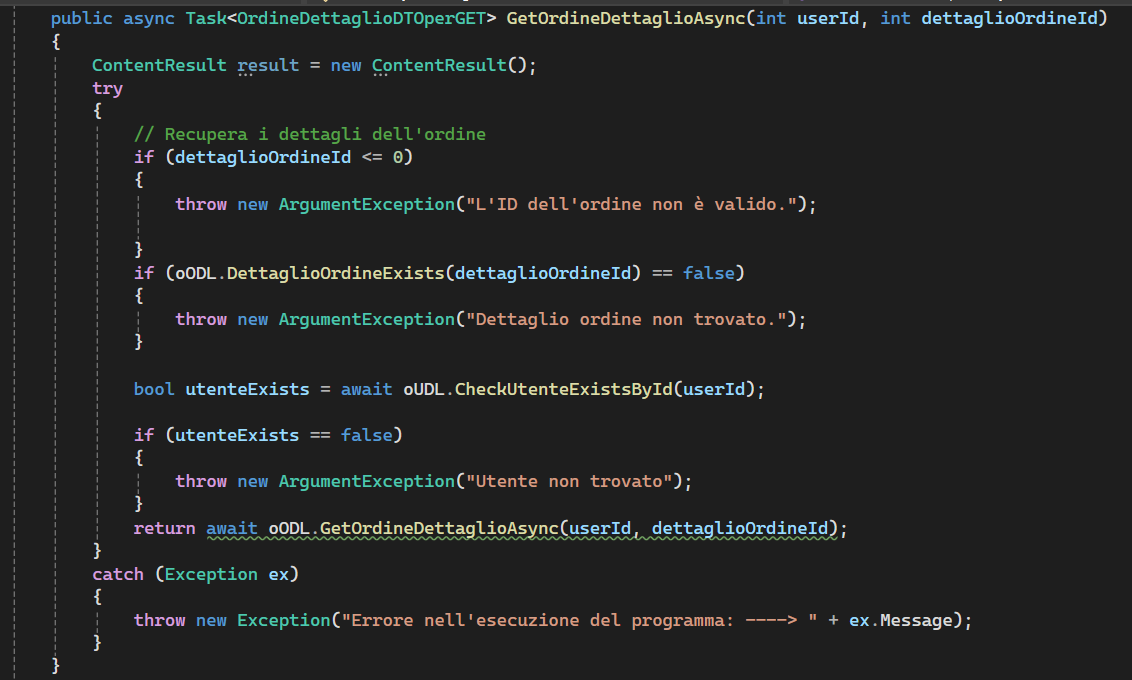
Se il recupero ha successo, restituisce un oggetto di OrdineDettaglioDTOperGET rappresentante l'ordine dettagliato con uno stato HTTP 200 (OK). Se si verifica un errore durante il recupero, restituisce uno stato HTTP 400 (Bad Request) insieme a un messaggio di errore che indica cosa è andato storto durante l'operazione di recupero.

DTO usato per mappare l’oggetto:



Dto usato per mappare un oggetto contenente informazioni specifiche di un ordine, come nome e descrizione del prodotto, descrizione dello stato dell'ordine, quantità, ID del prodotto, data di registrazione e data di aggiornamento.

BusinessLayer – Controlli di business



In questo metodo GetOrdineDettaglioAsync, stiamo gestendo la logica per recuperare i dettagli di un ordine in modo asincrono.

Controlli eseguiti:

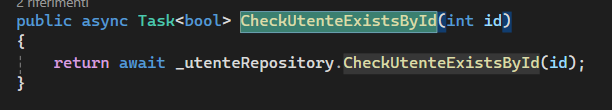
Controllo dell'ID dell'ordine: Verifichiamo che l'ID del dettaglio dell'ordine fornito sia valido. Se l'ID è minore o uguale a zero, lanciamo un'eccezione ArgumentException con un messaggio indicando che l'ID dell'ordine non è valido.

Verifica dell'esistenza del dettaglio dell'ordine(DettaglioOrdineExists): Controlliamo se il dettaglio dell'ordine esiste nel database. Se non esiste, lanciamo un'eccezione ArgumentException indicando che il dettaglio dell'ordine non è stato trovato.

A screen shot of a computer

Description automatically generated

Verifica dell'esistenza dell'utente(CheckUtenteExistsById): Utilizziamo un altro livello di controllo di business per verificare se l'utente associato all'ordine esiste nel sistema. Se l'utente non esiste, lanciamo un'eccezione ArgumentException indicando che l'utente non è stato trovato.

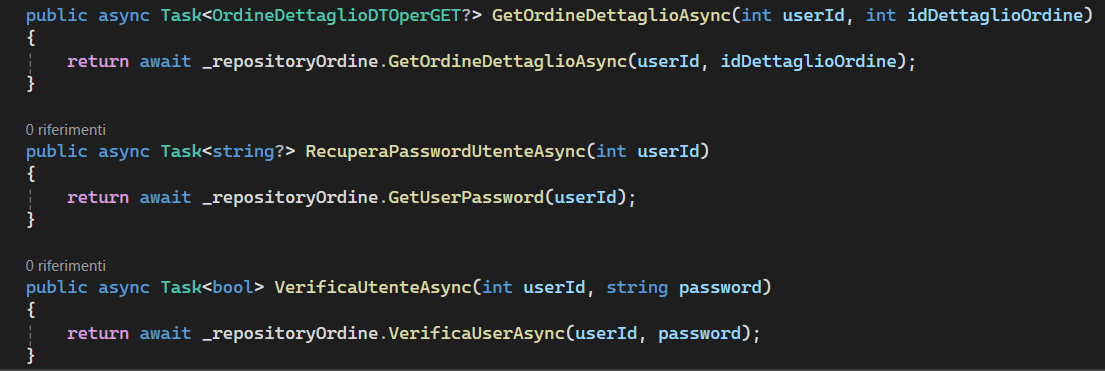


Recupero dei dettagli dell'ordine: Se tutti i controlli di business passano con successo, chiamiamo il metodo GetOrdineDettaglioAsync del livello dati per recuperare effettivamente i dettagli dell'ordine associati all'ID dell'utente e all'ID del dettaglio dell'ordine forniti.

Gestione delle eccezioni: Se si verifica un errore durante il processo di recupero, catturiamo l'eccezione, aggiungiamo un messaggio personalizzato per identificare l'errore, e lanciamo una nuova eccezione Exception con questo messaggio.

Data Layer (IRepositoryOrdine – RepositoryOrdine – ManageOrdineData)

ManageOrdineData.cs

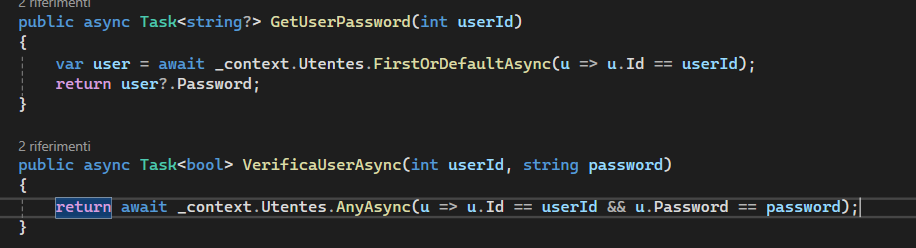


In queste tre funzioni stiamo eseguendo operazioni asincrone per interagire con il repository degli ordini:

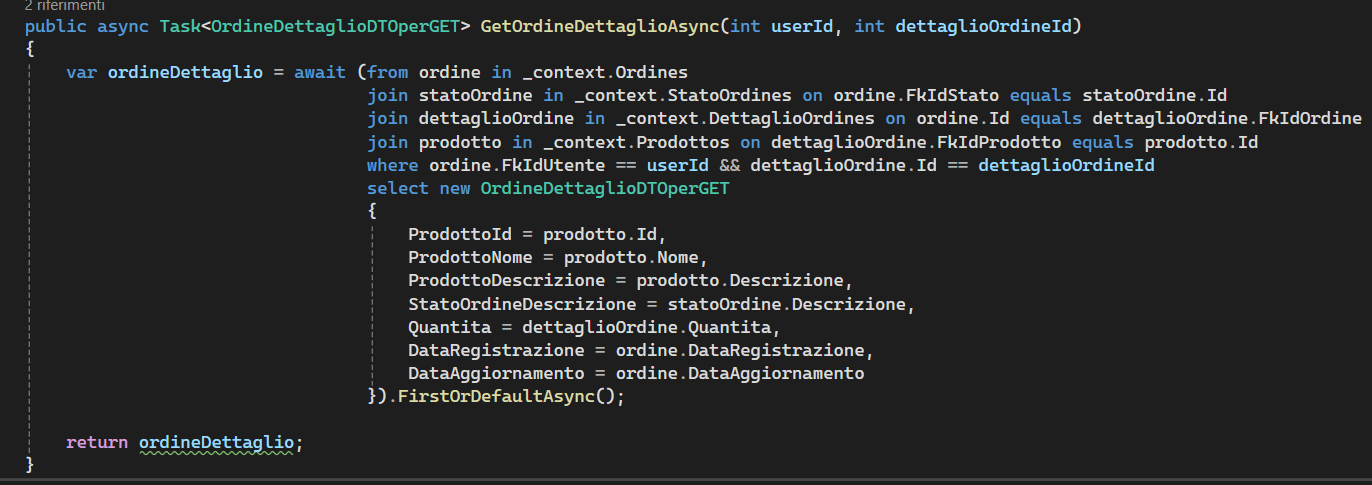
GetOrdineDettaglioAsync: Recuperiamo i dettagli di un ordine associati a un determinato utente e ID di dettaglio dell'ordine.

RecuperaPasswordUtenteAsync: Recuperiamo la password di un utente dato il suo ID.

VerificaUtenteAsync: Verifichiamo se un utente, dato il suo ID e la sua password, è valido nel sistema.



Repository Ordine



Analisi quey LINQ.

from ordine in \_context.Ordines: Inizia una query LINQ per selezionare i dati dalla tabella Ordines.

join statoOrdine in \_context.StatoOrdines on ordine.FkIdStato equals statoOrdine.Id: Esegue un join con la tabella StatoOrdines utilizzando la chiave esterna FkIdStato della tabella Ordines e l'ID della tabella StatoOrdines.

join dettaglioOrdine in \_context.DettaglioOrdines on ordine.Id equals dettaglioOrdine.FkIdOrdine: Esegue un secondo join con la tabella DettaglioOrdines utilizzando l'ID dell'ordine come chiave esterna.

join prodotto in \_context.Prodottos on dettaglioOrdine.FkIdProdotto equals prodotto.Id: Esegue un terzo join con la tabella Prodottos utilizzando l'ID del prodotto come chiave esterna.

where ordine.FkIdUtente == userId && dettaglioOrdine.Id == dettaglioOrdineId: Applica dei filtri per selezionare solo gli ordini associati a un determinato utente (userId) e un particolare ID del dettaglio dell'ordine (dettaglioOrdineId).

Spiegazione metodo GetOrdineDettaglioAsync

In questa funzione GetOrdineDettaglioAsync, stiamo recuperando i dettagli di un ordine associati a un determinato utente e ID del dettaglio dell'ordine. Ecco cosa stiamo facendo:

Join delle tabelle: Utilizziamo linq per eseguire una query che unisce diverse tabelle del database. Le tabelle coinvolte sono Ordines, StatoOrdines, DettaglioOrdines, e Prodottos.

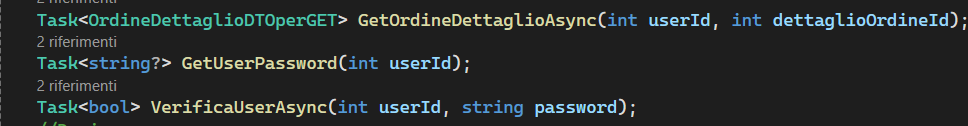
Filtro dei risultati: Applichiamo dei filtri per recuperare solo i dettagli dell'ordine desiderati. In particolare, filtriamo gli ordini in base all'ID dell'utente (userId) e all'ID del dettaglio dell'ordine (dettaglioOrdineId).

Selezione dei dati: Selezioniamo i dati rilevanti da queste tabelle per costruire un oggetto OrdineDettaglioDTOperGET, che rappresenta i dettagli dell'ordine nel formato desiderato. I dati selezionati includono l'ID e la descrizione del prodotto, la descrizione dello stato dell'ordine, la quantità, e le date di registrazione e aggiornamento dell'ordine.

Recupero dei dati: Utilizziamo FirstOrDefaultAsync() per eseguire la query in modo asincrono e recuperare il primo risultato corrispondente, oppure null se non è stato trovato alcun risultato.

Ritorno dei risultati: Restituiamo l'oggetto OrdineDettaglioDTOperGET contenente i dettagli dell'ordine recuperati.

IRepositoryOrdine



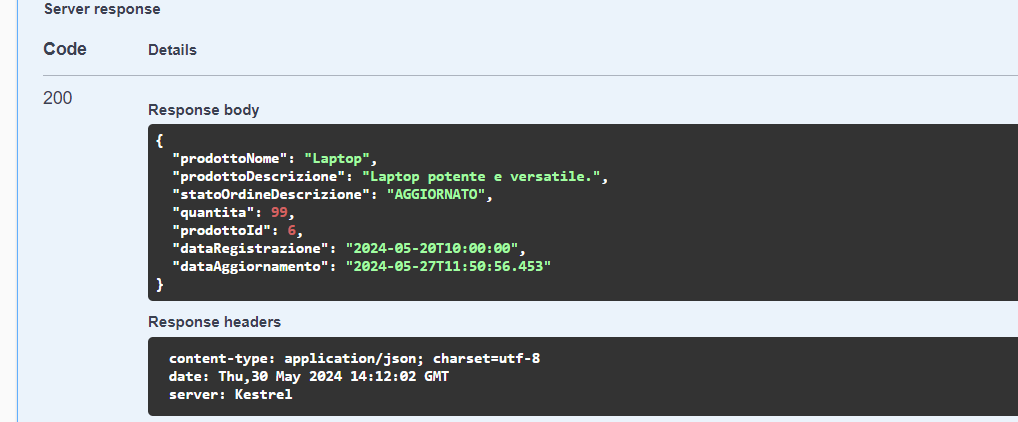
Queste tre funzioni sono metodi asincroni che interagiscono con il repository degli ordini:

GetOrdineDettaglioAsync: Recupera i dettagli di un ordine associato a un utente specifico e a un determinato ID di dettaglio dell'ordine.

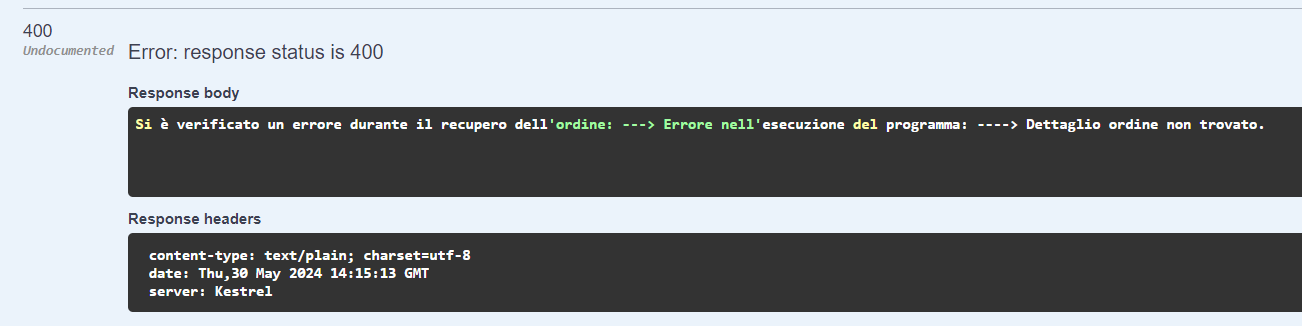
GetUserPassword: Recupera la password di un utente dato il suo ID.

VerificaUserAsync: Verifica se un utente, dato il suo ID e la sua password, è valido nel sistema.

Esecuzione del programma



Questo comando CURL esegue una richiesta GET all'endpoint https://localhost:7031/api/Ordine/GetOrdineByUser&Dettaglio1 con il parametro dettaglioOrdineId impostato su 10. La risposta del server ha uno status code 200, indicando che la richiesta è stata completata con successo.



Questa richiesta CURL ha restituito uno status code 400, indicando un errore nella richiesta. Nel corpo della risposta, viene restituito un messaggio di errore che indica che si è verificato un problema durante il recupero dell'ordine. Il messaggio specifico dell'errore è "Dettaglio ordine non trovato".

Questo significa che la richiesta ha fallito perché il dettaglio dell'ordine con ID 11 non è stato trovato nel sistema. Potrebbe essere dovuto a un'errata specifica dell'ID dell'ordine o a un problema nel sistema che impedisce il recupero del dettaglio dell'ordine richiesto.