# Architettura applicativa funzionalità

OrdineController

PutOrdine

­

BusinessLayer

ManageOrdineBusiness

­

DataLayer

ManageOrdineData

DataLayer.Repository

RepositoryOrdine

f

IRepositoryOrdine

f

AcademyShopDB

Ordine

DettaglioOrdine

Prodotto

## Api Layer

In questo strato applicativo è presente l’ **OrdineController** che implementa il metodo **PutOrdine** il quale espone un’operazione Rest che restituisce un JSON contenente i dati dell’Ordine modificato.

Immagine che contiene testo, schermata, Carattere

Descrizione generata automaticamente

Il metodo **PutOrdine** consente di aggiornare un ordine esistente. Accetta l'ID dell'utente, l'ID del dettaglio dell'ordine e la quantità da aggiornare. Chiama il metodo **ModificaOrdineCompletaAsync** nel Business Layer per effettuare l'aggiornamento. Se l'operazione ha successo, restituisce un codice di stato HTTP 200 con i dettagli dell'ordine modificato. In caso di fallimento, restituisce un codice di stato HTTP appropriato con un messaggio di errore.

## Business Layer

In questo strato applicativo è presente la classe **ManageOrdineBusiness** la quale implementa il metodo **ModificaOrdineCompletaAsync.**

Il metodo **ModificaOrdineCompletaAsync** consente di modificare un ordine esistente identificato dall'ID utente e dall'ID del dettaglio dell'ordine, aggiornando la quantità richiesta. Verifica l'esistenza dell'ordine, lo stato e la disponibilità dei prodotti. Se l'operazione ha successo, restituisce i dettagli dell'ordine modificato; altrimenti, restituisce un messaggio di errore e un codice di stato HTTP appropriato.

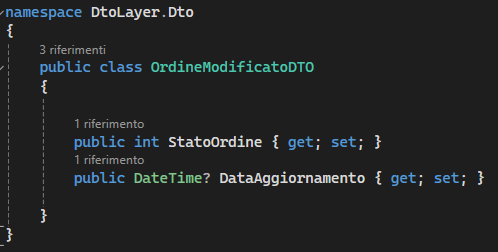
Le varie operazioni di CRUD ognuna divisa atomicamente in un metodo specifico vengono demandate a un altro strato applicativo, il Dao Layer.

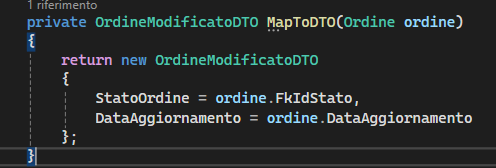
Immagine che contiene testo, schermata, software, Sistema operativo

Descrizione generata automaticamente

Nel Business Layer come si può notare dal tipo di ritorno del metodo ho fatto uso del **pattern DTO** poiché mi serve mappare l’oggetto Ordine da restituire al controller con solo alcuni campi e non tutti quelli presente nel database.

## Oggetto Ordine mappato nel DTO Layer





## Dao Layer

In questo strato applicativo la classe **ManageOrdineData** implementa i vari metodi legati alle operazioni di CRUD e li demanda a sua volta alla classe **RepositoryOrdine** presente sempre nel Dao Layer che implementa i metodi dell’interfaccia **IRepositoryOrdine** rispettando le linee guida del pattern **Loose Coupling.**

#### ManageOrdineData

Immagine che contiene testo, schermata

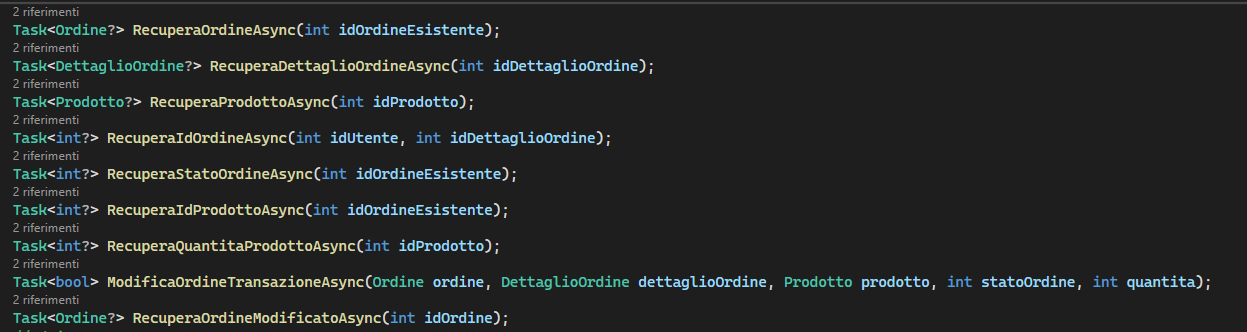
Descrizione generata automaticamente

Immagine che contiene schermata, testo

Descrizione generata automaticamente

Il corpo di questi metodi e quindi la loro esecuzione come detto in precedenza avviene all’interno della classe **RepositoryOrdine,** la quale implementa i metodi dell’interfaccia **IRepositoryOrdine.**

#### IRepositoryOrdine

****

#### RepositoryOrdine

Immagine che contiene testo, schermata

Descrizione generata automaticamente

Immagine che contiene testo, schermata, software

Descrizione generata automaticamente

## Spiegazione dei singoli Metodi

* **RecuperaOrdineAsync(int idOrdineEsistente)** 🡪 Recupera un ordine dal database utilizzando l’ID fornito.
* **RecuperaDettaglioOrdineAsync(int idDettaglioOrdine)** 🡪 Recupera il dettaglio di un ordine dal database utilizzando l’ID del dettaglio dell’ordine.
* **RecuperaProdottoAsync(int idProdotto)** 🡪 Recupera un prodotto dal database utilizzando l’ID del prodotto.
* **RecuperaIdOrdineAsync(int idUtente, int idDettaglioOrdine)** 🡪 Recupera l’ID di un ordine associato a un utente e un dettaglio dell’ordine specifico.
* **RecuperaStatoOrdineAsync(int idOrdineEsistente)** 🡪 Recupera lo stato di un ordine associato a un ordine specifico.
* **RecuperaIdProdottoAsync(int idOrdineEsistente)** 🡪 Recupera l’ID di un prodotto associato a un ordine specifico.
* **RecuperaQuantitaProdottoAsync(int idProdotto)** 🡪 Recupera la quantità disponibile di un prodotto utilizzando l’ID del prodotto.
* **ModificaOrdineTransazioneAsync(Ordine ordine, DettaglioOrdine dettaglioOrdine, Prodotto prodotto, int statoOrdine, int quantita)** 🡪 Esegue una transazione per aggiornare nella tabella Ordine la data di aggiornamento e lo stato dell’ordine e per aggiornare la quantità nelle tabelle Prodotto e DettaglioOrdine. In caso di non successo della transazione tutte le modifiche vengono annullate.
* **RecuperaOrdineModificatoAsync(int idOrdine)** 🡪 Recupera l’ordine modificato dal database, includendo i dettagli dell’ordine modificato.

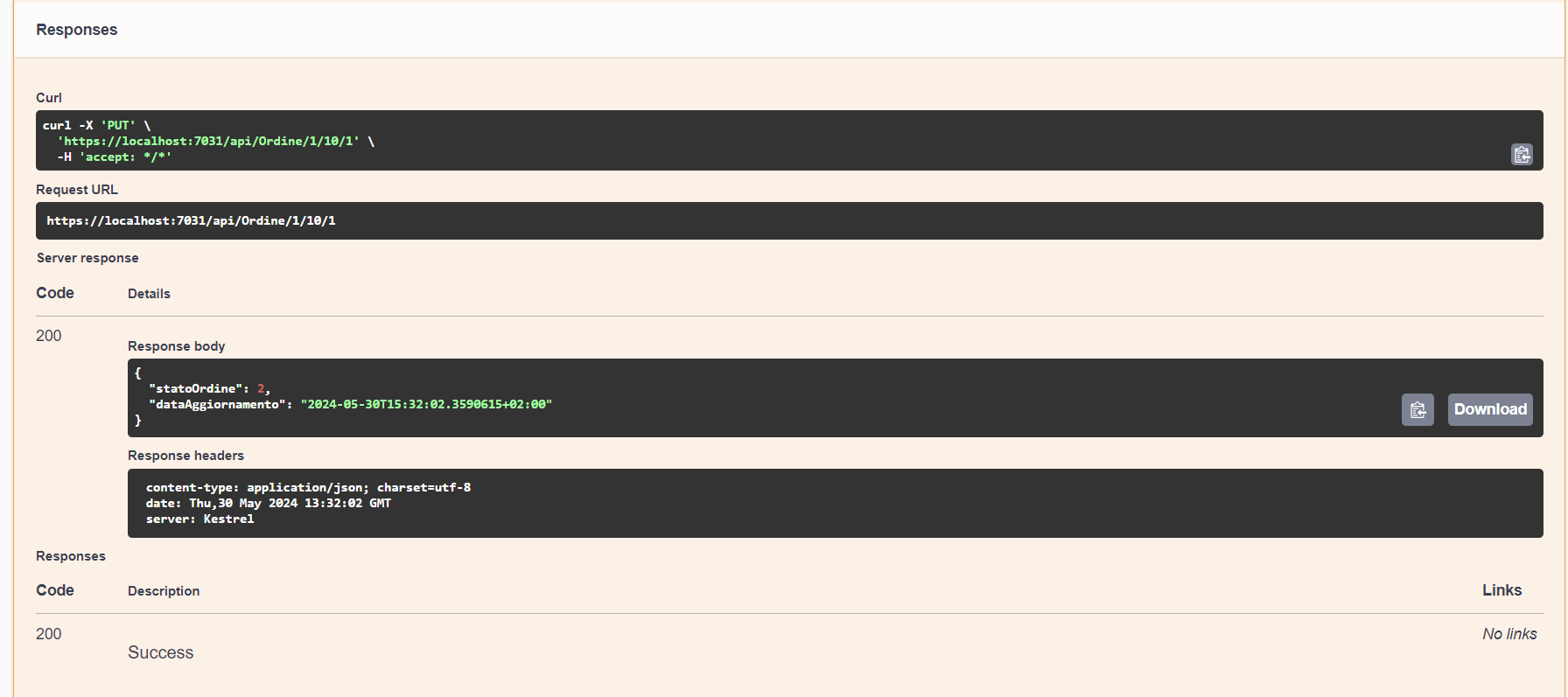
## Esempio di test Swagger con successo

Inserisco i parametri richiesti dal metodo.

Immagine che contiene testo, schermata, software, numero

Descrizione generata automaticamente

Eseguo il test con Swagger.



Il risultato come da previsione è la risposta dello Status Code 200 con i dati dell’ordine modificato:

* Stato dell’ordine: corrisponde a 2 perché l’ordine dallo stato di aperto corrispondente ad 1 passa a quello di aggiornato (2).
* Data aggiornamento: data in cui avviene l’aggiornamento dell’ordine (Datetime)

## Esempi di test Swagger con errori

#### Ordine non presente nel database

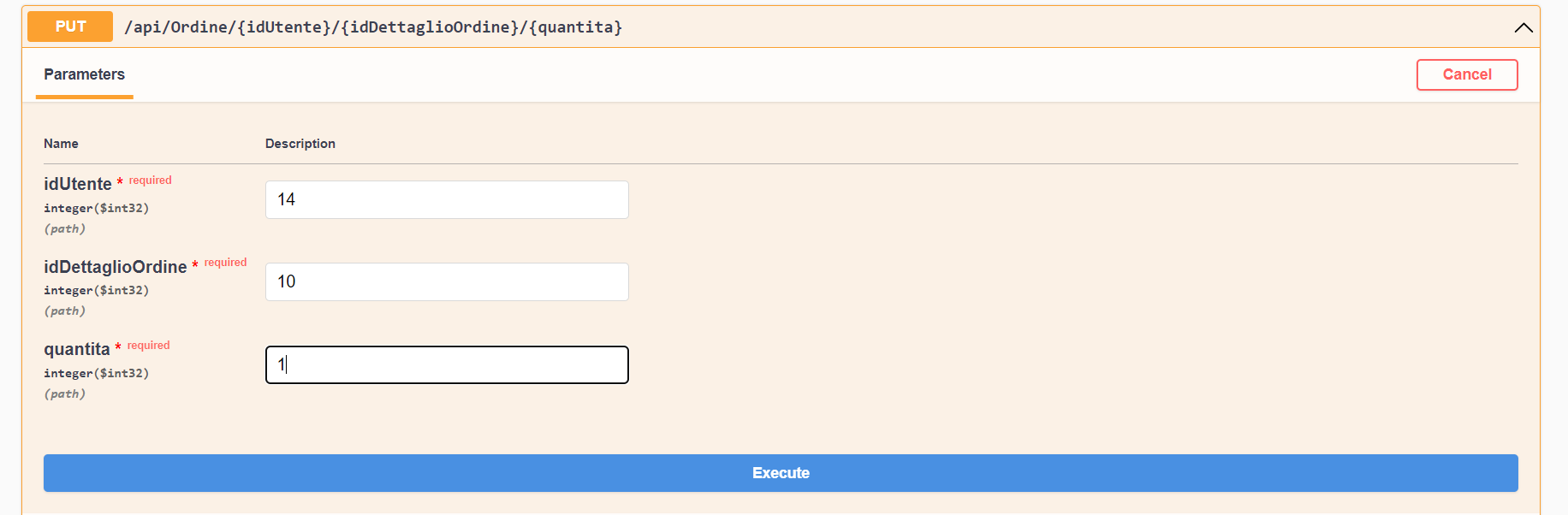


Immagine che contiene schermata, testo, software

Descrizione generata automaticamente

L’ ordine non esiste perché non è stato trovato nessun ordine con i parametri inseriti e avrò quindi uno Status Code 404 con il messaggio “L’ordine non esiste”.

#### Stato ordine chiuso

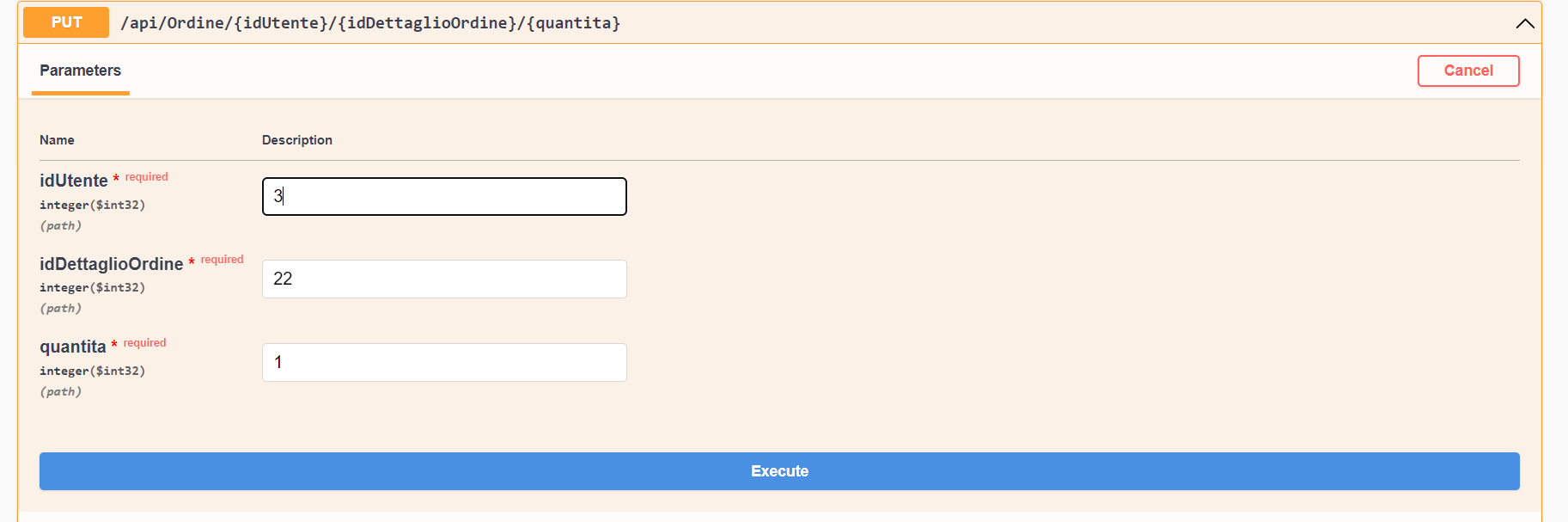


Immagine che contiene schermata, testo, linea, Carattere

Descrizione generata automaticamente

Dai parametri che ho inserito è stato trovato un ordine nel database con FK\_ID\_Stato uguale a 3 e quindi il risultato sarà uno Status Code 400 con il messaggio “L’ordine è chiuso”.

#### Quantità prodotto disponibile non sufficiente

Immagine che contiene testo, software, numero, schermata

Descrizione generata automaticamente

Immagine che contiene schermata, testo, Carattere, software

Descrizione generata automaticamente

Inserendo come parametro una quantità troppo alta rispetto a quella presenta nel database il risultato sarà uno Status Code 400 con il messaggio “La quantità disponibile non è sufficiente”.