La soluzione comprende 9 progetti così divisi:

* Photosi.api

l’api che fa da gateway verso gli altri 4 microservizi.

Si può interagire tramite l’interfaccia fatta con Swagger UI.

* Photosi.orders

Il microservizio che si occupa di gestire gli ordini.

Si appoggia ad un database sql server chiamato orders\_db.

Ha il suo progetto con i test chiamato Test.Photosi.orders.

* Photosi.users

Il microservizio che si occupa di gestire gli utenti.

Si appoggia ad un database sql server chiamato users\_db.

Ha il suo progetto con i test chiamato Test.Photosi.users.

* Photosi.catalog

Il microservizio che si occupa di gestire gli articoli.

Si appoggia ad un database sql server chiamato catalog\_db.

Ha il suo progetto con i test chiamato Test.Photosi.catalog.

* Photosi.pickuppoints

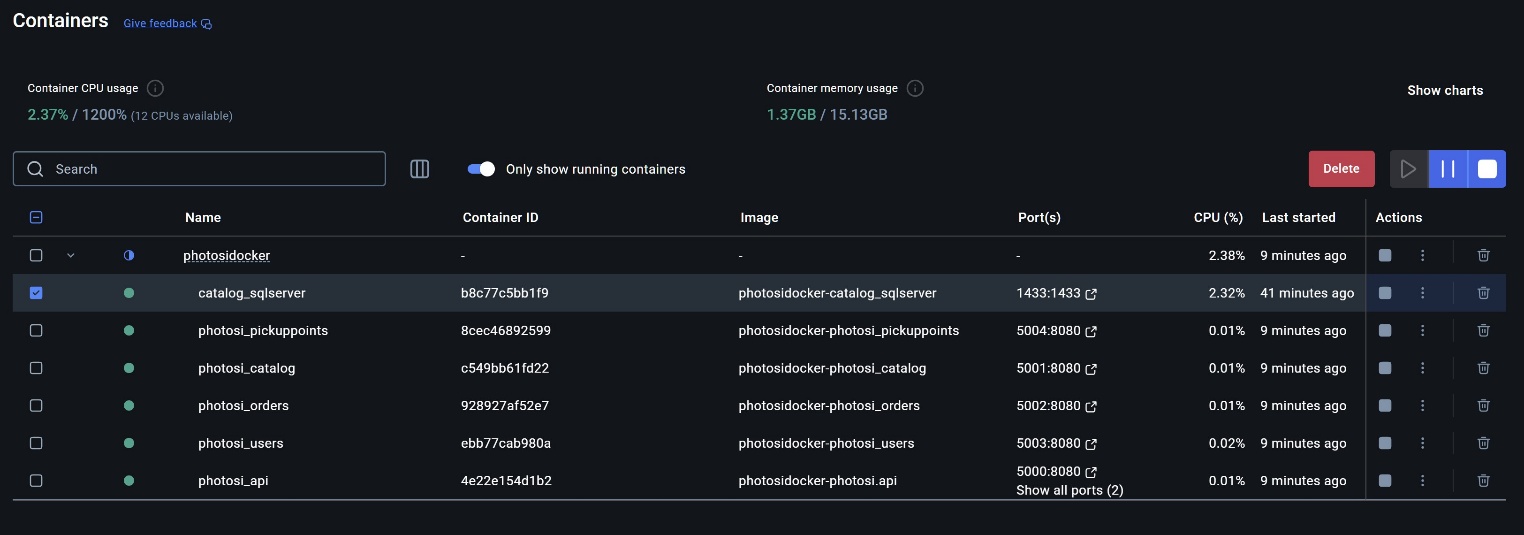
Il microservizio che si occupa di gestire gli indirizzi dei punti di ritiro.

Si appoggia ad un database sql server chiamato pickuppoints\_db.

Ha il suo progetto con i test chiamato Test.Photosi.pickuppoints.

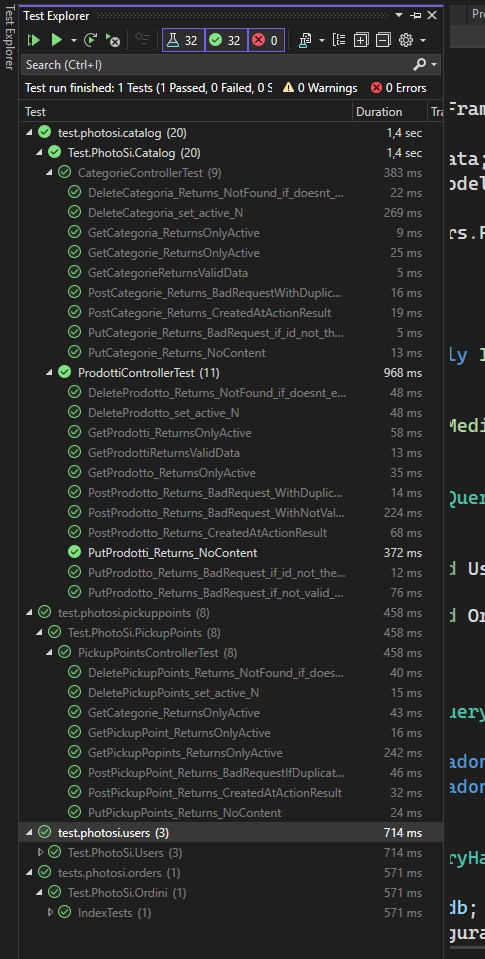
La soluzione comprende il file docker-compose per creare i contenitori dei microservizi e dell’istanza di Sql Server Express.

Dovrebbe bastare eseguire un “docker-compose up –build”.



Inizialmente avevo optato per un contenitore separato con la propria istanza di Sql Server per ogni database ma si è rivelato troppo pesante da eseguire sulla mia macchina; alla fine tutti i database sono creati nella stessa istanza contenuta nel contenitore catalog\_sqlserver.

Tutti i test sono passati con successo.  
Prima di eseguire i test è necessario eseguire il container contenente i database (catalog\_sqlserver).



Tra i testi si trovano sia unit test sia integration test.

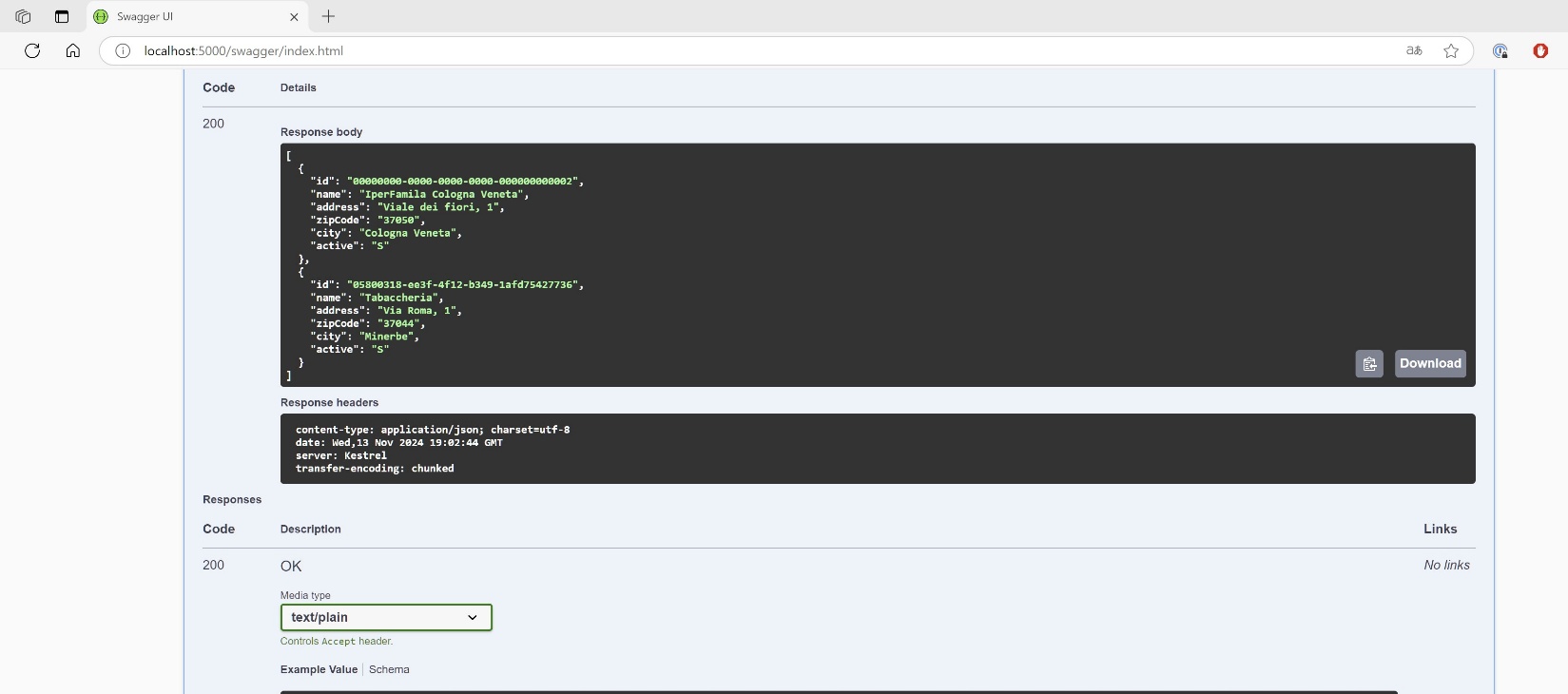
In alcuni casi è stato usato Respawn per eseguire i test all’interno di una transazione direttamente sul database reale e ripulire i dati alla fine di ogni test.

I test utilizzano le connectionstring contenute nei file appsettings.json e si collegano ai db presenti nell’istanza di Sql Server raggiungibile in locale all’indirizzo localhost:1433.

I microservizi eseguiti tramite docker invece utilizzano i file di configurazione appsettings.production.json e si collegano all’istanza di Sql server all’indirizzo “catalog\_sqlserver, 1433”.

L’api REST che fa da gateway è raggiungibile all’indirizzo <http://localhost:500/swagger/index.html>.

Anche per quest’ultima i percorsi dei microservizi sono contenuti nel file appsettings.json per l’esecuzione in debug e appsettings.production.json per l’esecuzione nei contenitori docker.



Nei vari progetti ho voluto adottare approcci diversi.

In alcuni progetti i controller hanno accesso direttamente al dbcontext mentre in altri ho adottato l’approccio vertical slicing e CQSR.

In questo caso l’accesso ai dati è demandato alla classe che rappresenta la query o il comando da eseguire e ognuna potrebbe adottare un approccio differente. Il controller invia il comando da eseguire alla classe opportuna tramite Mediatr, Respawn, Shouldly e xunit.