Università degli Studi di Salerno

Corso di Ingegneria Del Software

Eat & Reorder

Unit Test Report

**Docente: Studenti:**

Prof. Andrea De Lucia Abate Francesco

Dello Buono Marco

De Martino Vincenzo

Gagliardi Rosario

Sommario

[**1. Introduzione 2**](#_Toc30449956)

[**2. Relazione con altri documenti 2**](#_Toc30449957)

[2.1 Relazione con RAD 2](#_Toc30449958)

[2.2 Relazione con SDD 2](#_Toc30449959)

[2.3 Relazione con ODD 2](#_Toc30449960)

[**3. Approccio di Unit Testing 2**](#_Toc30449961)

[**4. Testing di unità 2**](#_Toc30449962)

[4.1 Test delle classi Servlet 2](#_Toc30449963)

[4.2 Test delle classi DAO 2](#_Toc30449964)

1. Introduzione

Il testing di unità rappresenta una delle fasi di testing più importanti, in quanto consiste nella verifica e il collaudo di singole unità software di un sistema. Tuttavia, questa fase è anche molto delicata e può essere causa di imprevisti e slittamenti dei tempi a causa di errori e malfunzionamento del sistema.

Questo **Unit Test Plan** scrive quindi l'approccio di test ed il framework generale che guiderà i test della piattaforma Eat&Reorder. Ha come scopo l’assicurarsi che i difetti critici vengano rimossi prima che possano iniziare i prossimi livelli di test.

Il documento introduce quindi:

• approccio di unit testing, ovvero le regole su cui basare il test e la descrizione del processo per impostare un test valido;

• Pass\fail criteria gestione dei test: processo per gestire la logistica del test e tutti gli eventi che si verificano durante l'esecuzione.

# Relazione con altri documenti

Essendo chiaramente in relazione con altri documenti, ne derivano vari criteri di accettazione del test. I documenti di test case specification devono essere disponibili prima dell'inizio della fase di progettazione del test.

Per individuare correttamente i test case si terrà conto dei documenti prodotti precedentemente. Infatti, la fase di testing è strettamene legata alle fasi precedenti, in quanto saranno un punto di partenza indispensabile per poter effettuare un testing corretto ed adeguato e per verificare che il sistema desiderato sia simile a quello proposto.

* Eat&Reorder – Test Case Specification Document
* Eat&Reoder – Test Plan Document

## Relazione con RAD

La relazione tra Unit Test Plan e RAD riguarda in particolare i requisiti funzionali e non funzionali del sistema poiché i test che saranno eseguiti su ogni funzionalità terranno conto delle specifiche espresse in tale documento.

## Relazione con SDD

Nel System Design Document abbiamo suddiviso il nostro sistema in sottosistemi e l’architettura in tre livelli: Presentation Layer, Application Layer e Storage Layer. Il test dei vari componenti deve rimanere fedele a queste suddivisioni il più possibile.

## Relazione con ODD

Il test d’integrazione farà quanto più riferimento possibile alle interfacce delle classi e i package definiti nell’ODD.

Inoltre, lo sviluppo deve essere completato ed i risultati condivisi tra il team di test per evitare i difetti duplicati. Infine, l’ambiente di test deve essere installato, configurato e pronto per l'uso dell'applicazione.

1. Approccio di Unit Testing

Il testing si compone di tre fasi. Inizialmente verranno eseguiti i test di unità dei singoli componenti, in modo da testare nello specifico la correttezza di ciascuna unità andando a constatare il corretto funzionamento di tutte le singole unità di codice.

I metodi delle classi dao sono stati testati verificando il corretto funzionamento e la consistenza delle query effettuate a database. Per fare ciò è stato creato un database di testing così da poter effettuare i test in integrazione tra i dao e il database.

Per il testing delle servlet abbiamo utilizzato delle classi driver (simulate tramite il framework **Junit**) e delle classi stub (simulate tramite il framework **Mockito**).

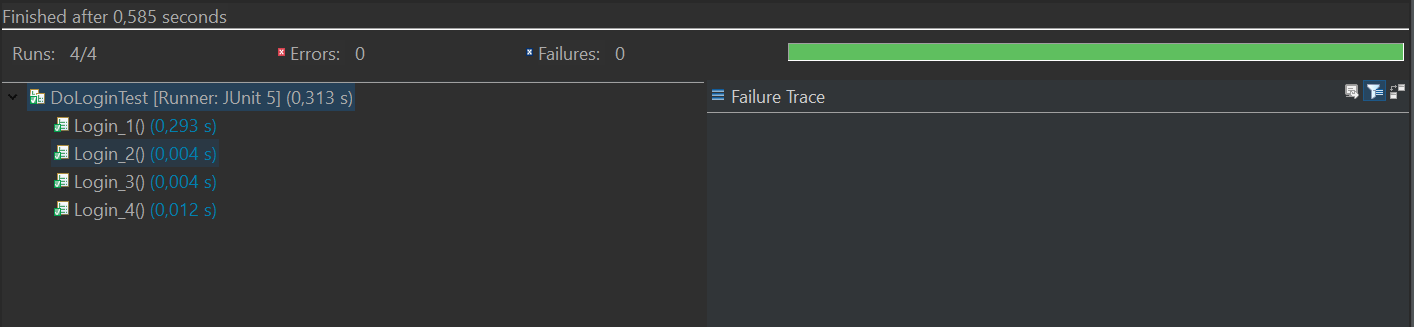
Per il testing delle classi dao abbiamo utilizzato degli stub simulati tramite il framework **Mockito**.

1. Testing di unità

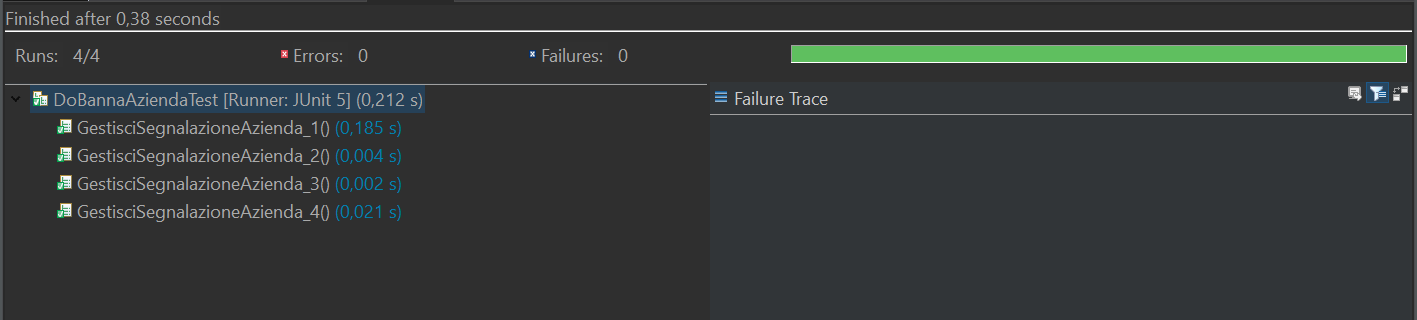
Per effettuare il testing di ogni singola componente del sistema verrà utilizzata la tecnica “Black-Box testing” attraverso il framework **JUnit**. In questa fase saranno analizzate le funzionalità dell’applicazione ed il comportamento delle singole componenti senza tener conto della loro struttura interna. I risultati del testing verranno analizzati e usati per correggere gli errori che causano il fallimento del sistema. Se si verifica un errore con dei risultati inattesi si interviene in maniera tempestiva sulla componente in modo da renderla correttamente funzionante e procedere con le fasi di testing successive. Di seguito sono riportati tutti i test effettuati:

* 1. **Test delle classi Servlet**

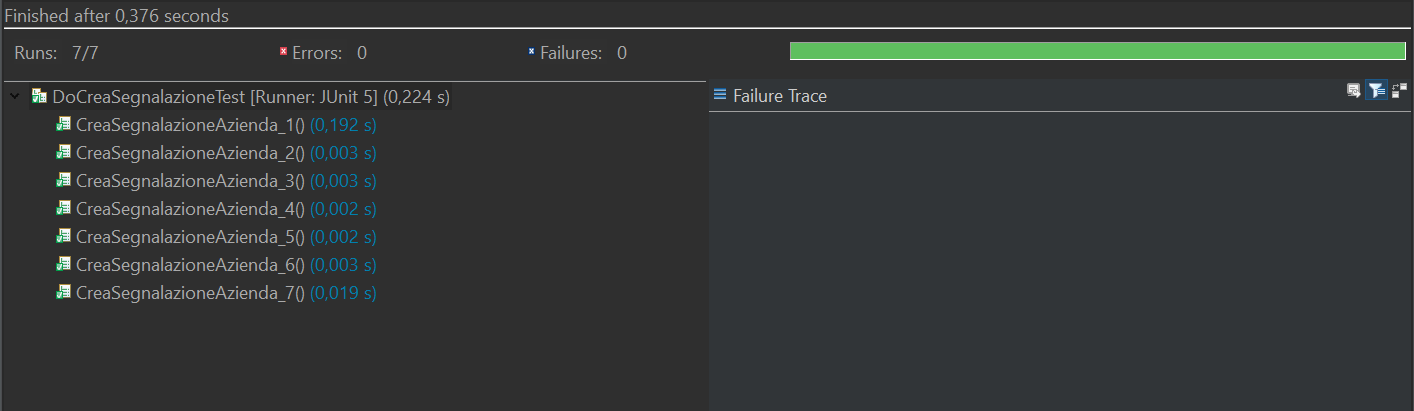
**DoLoginTest**

****

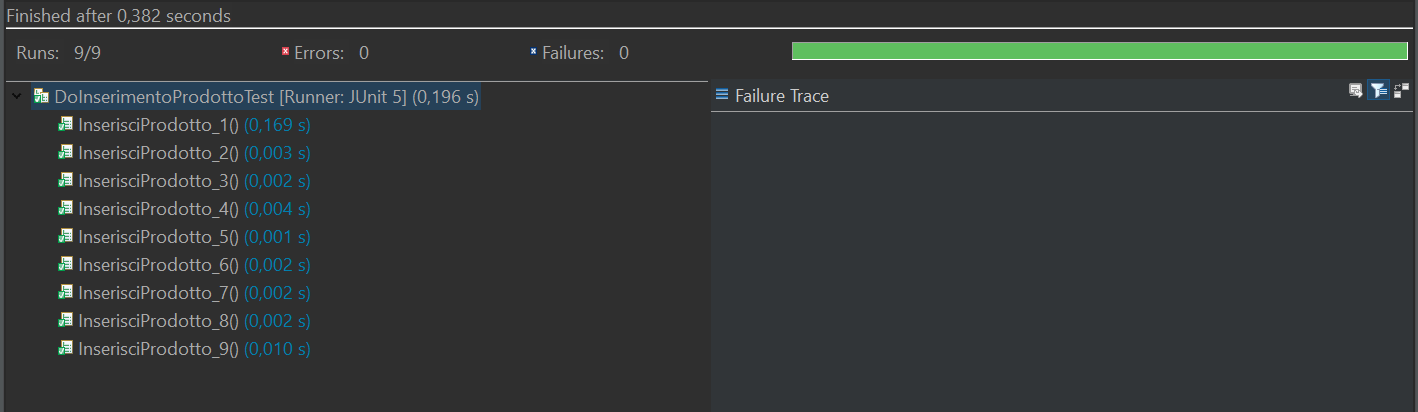
**DoBannaAziendaTest**

****

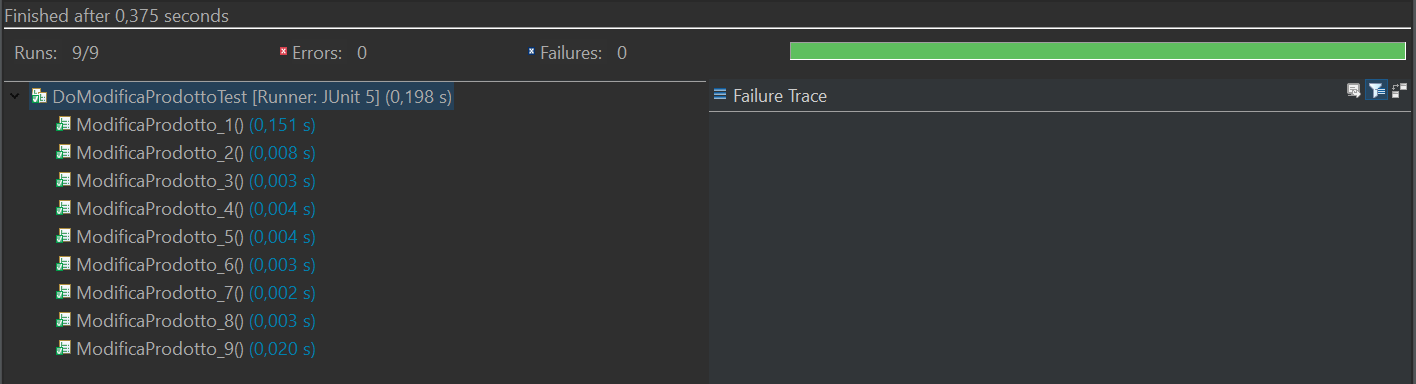
**DoCreaSegnalazioneTest**

****

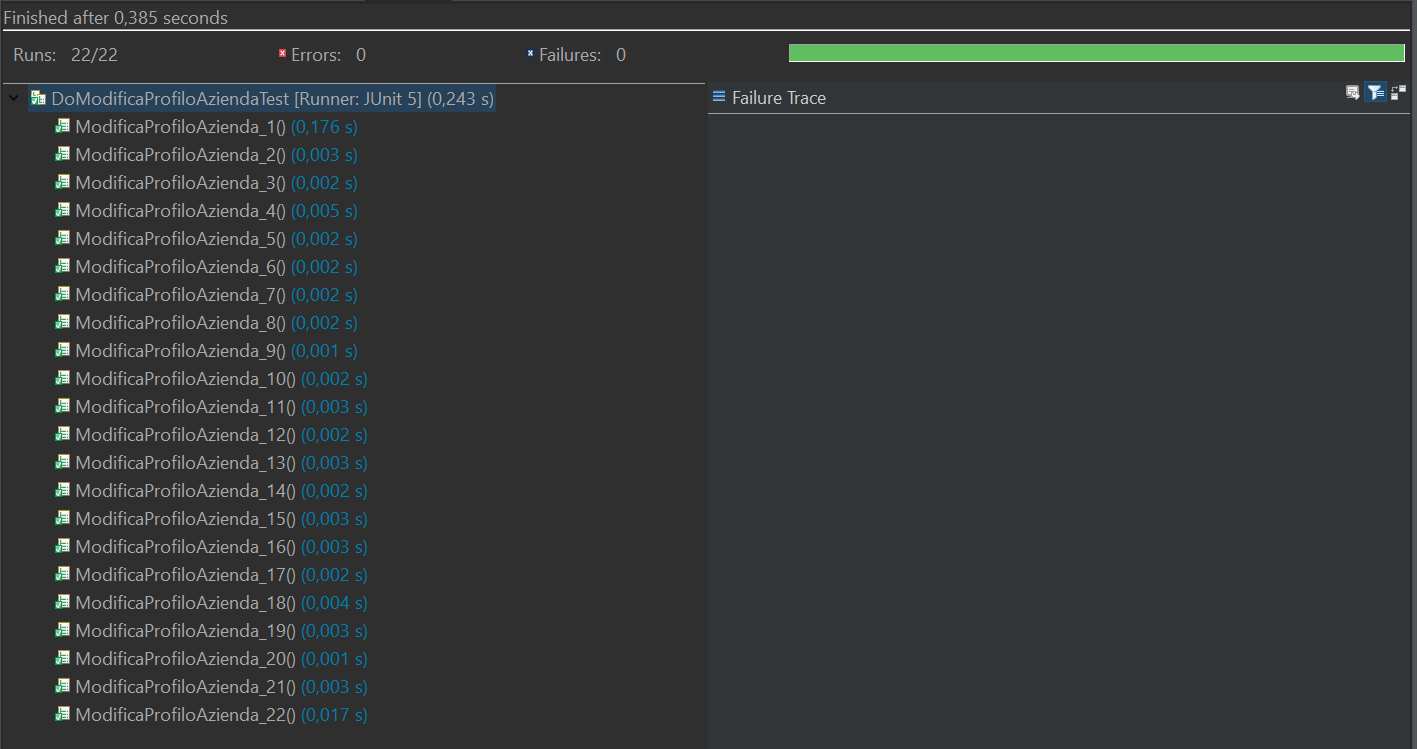
**DoInserimentoProdottoTest**

****

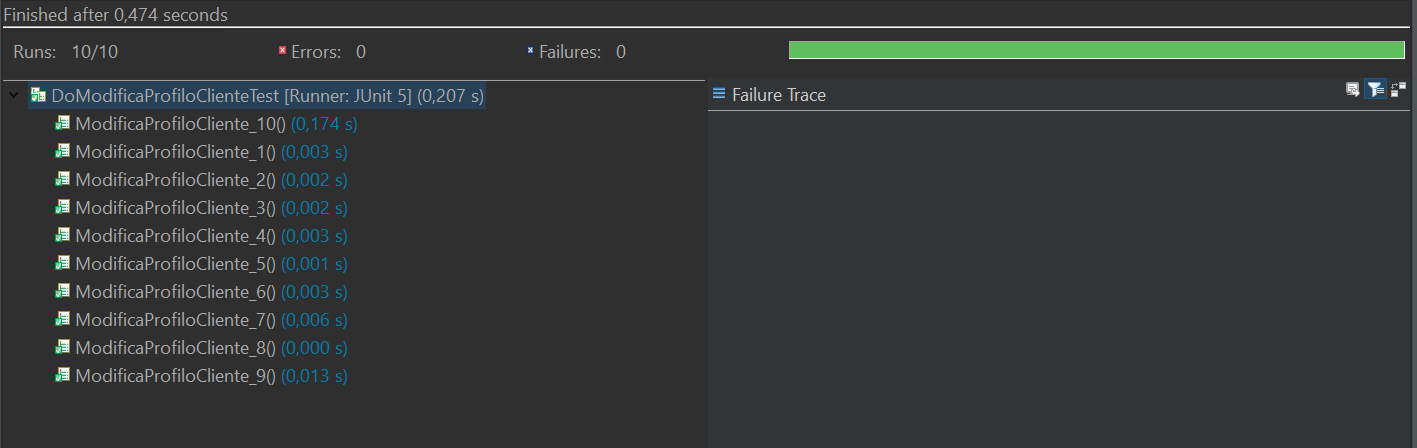
**DoModificaProdottoTest**

****

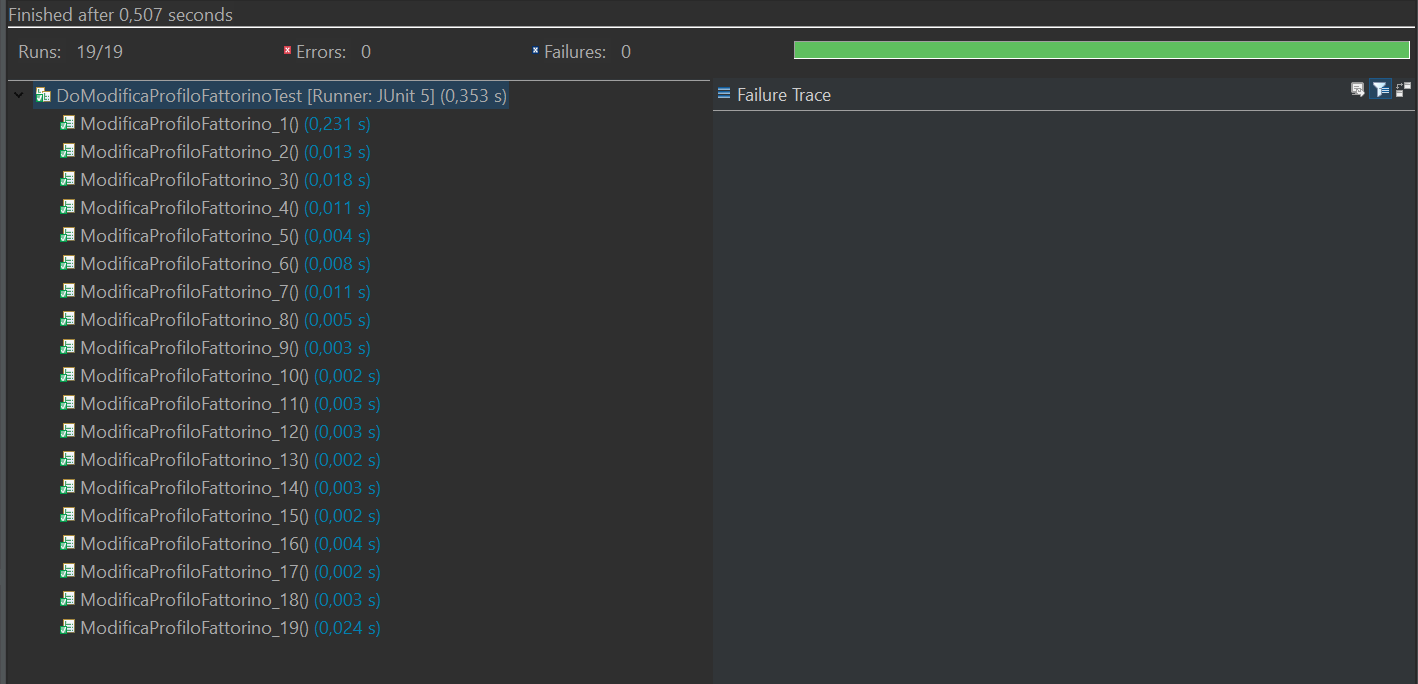
**DoModificaProfiloAziendaTest**

****

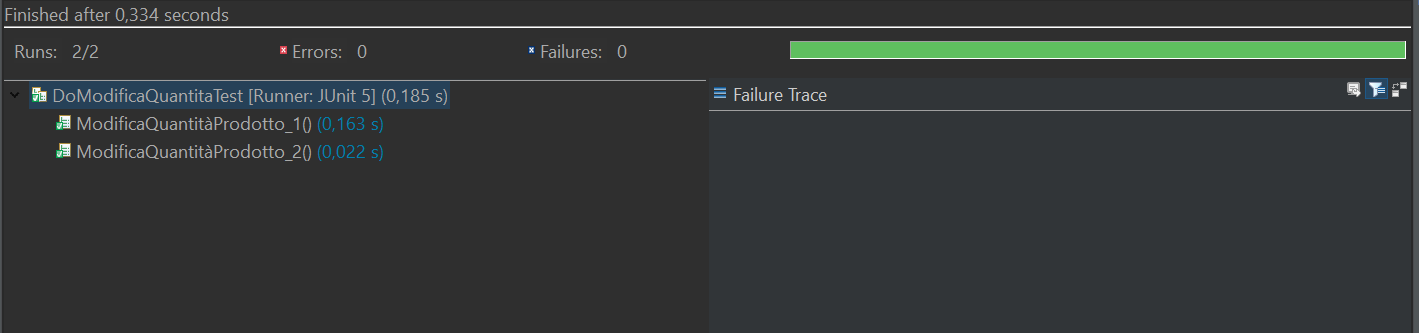
**DoModificaProfiloClienteTest**

****

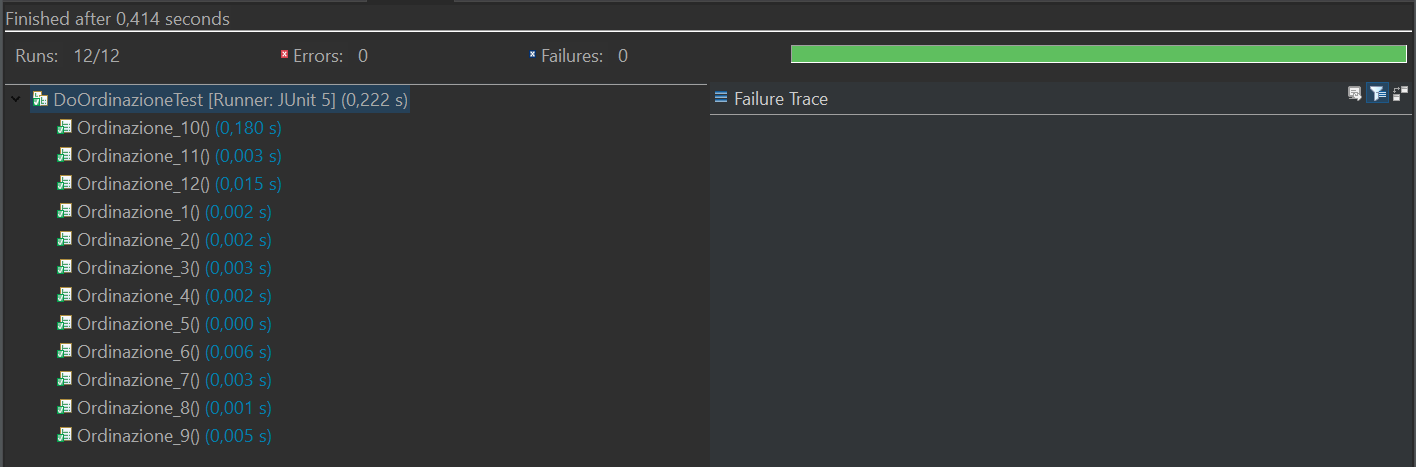
**DoModificaProfiloFattorinoTest**

****

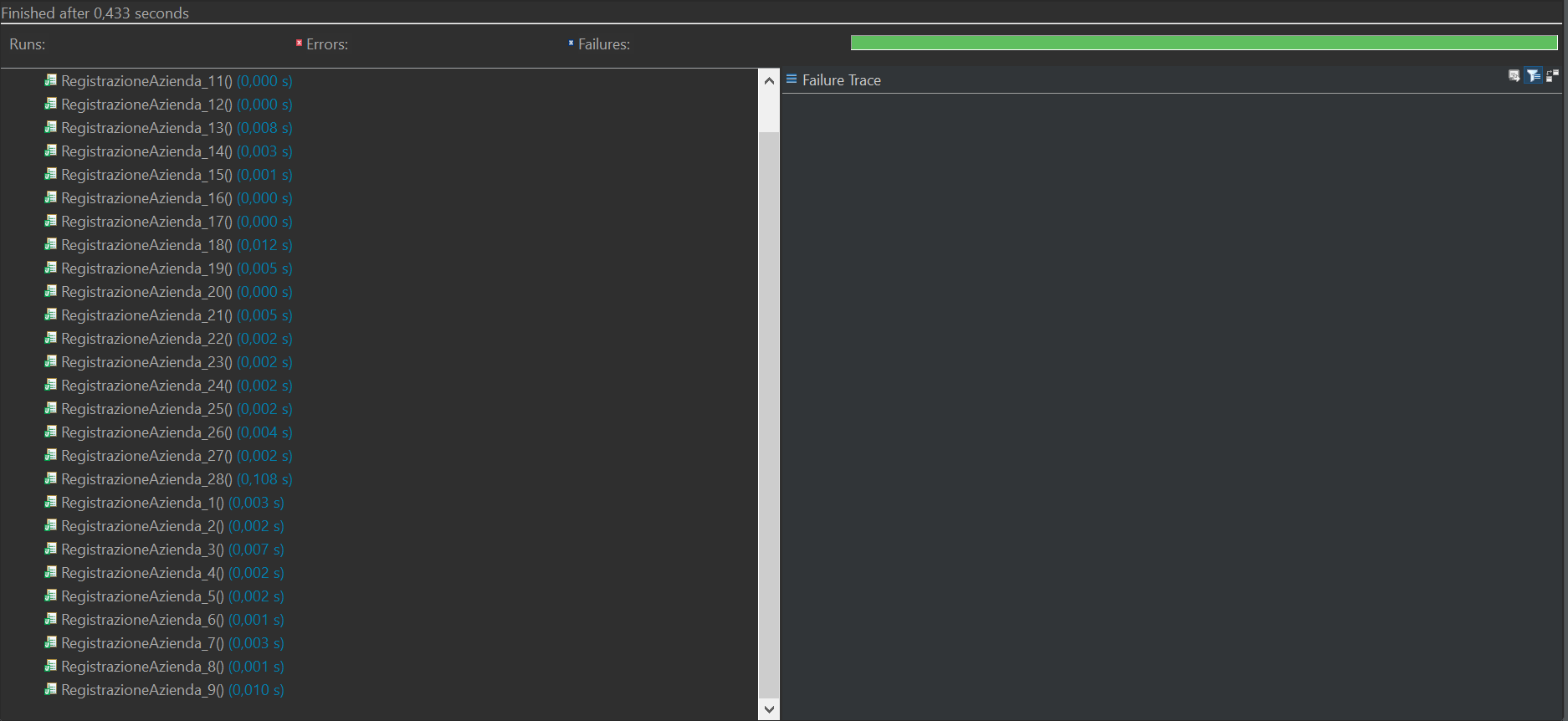
**DoModificaQuantitaTest**

****

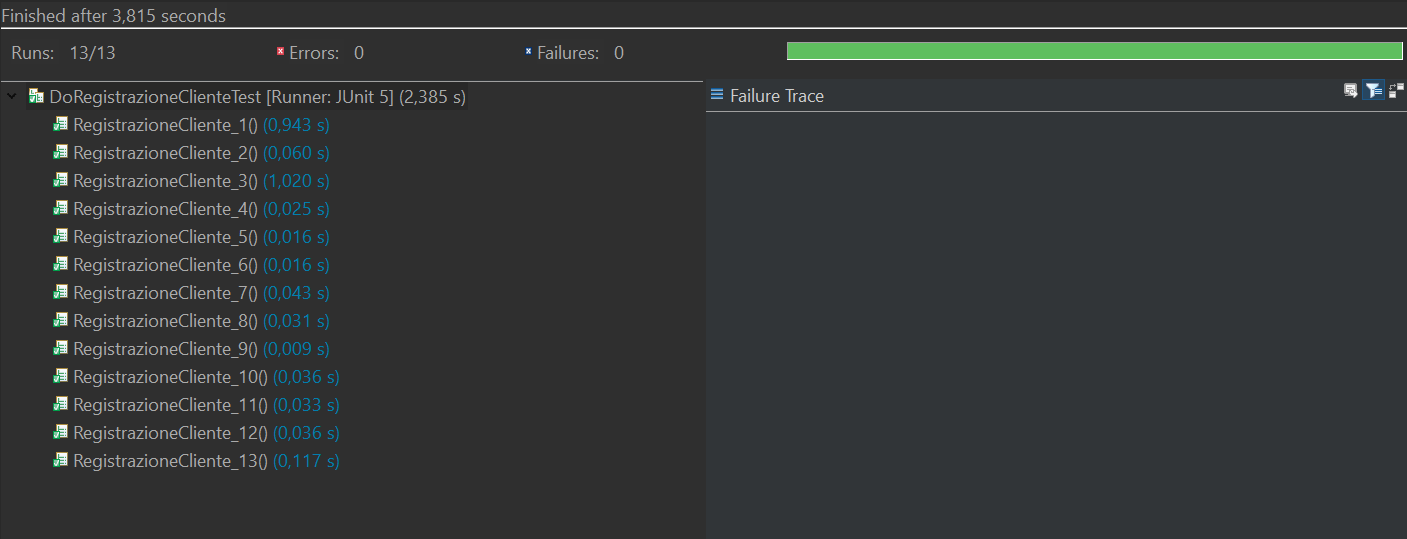
**DoOrdinazioneTest**

****

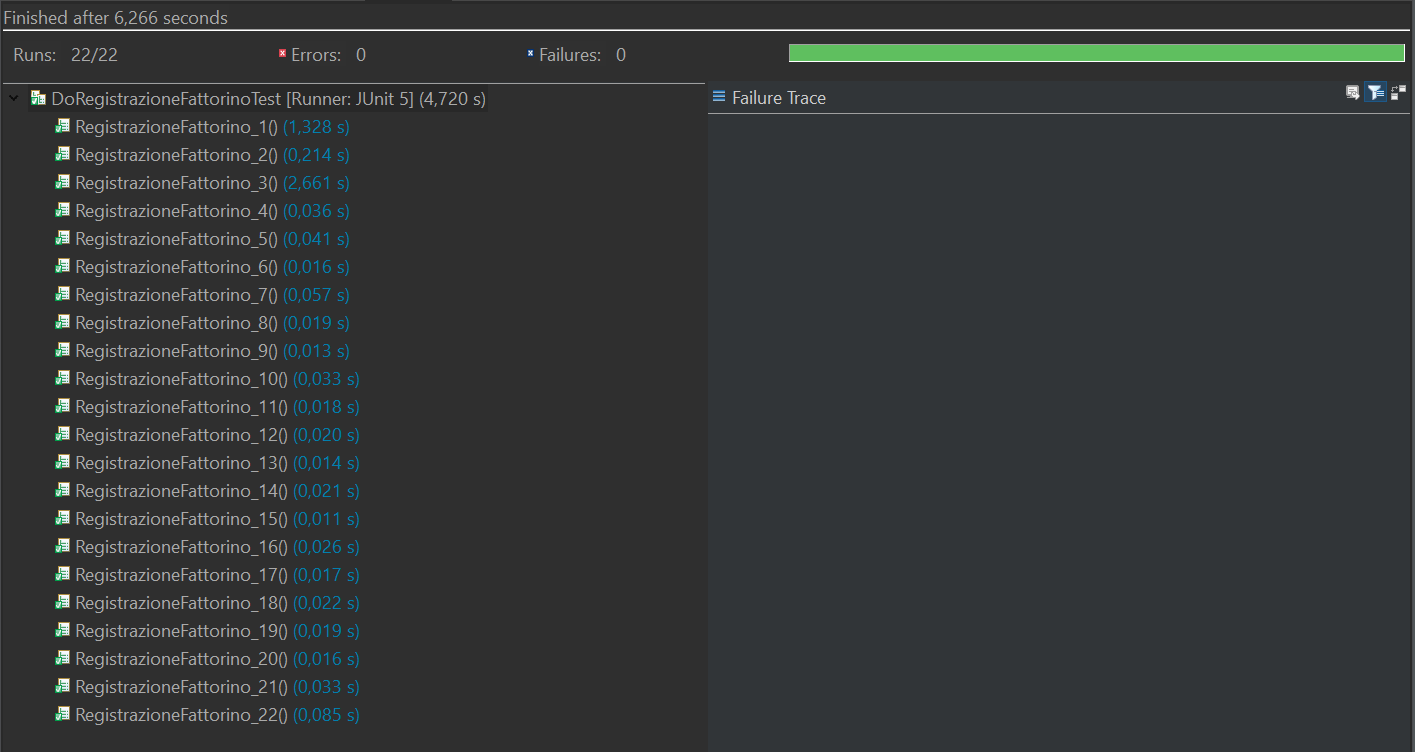
**DoRegistrazioneAziendaTest**

****

**DoRegistrazioneClienteTest**

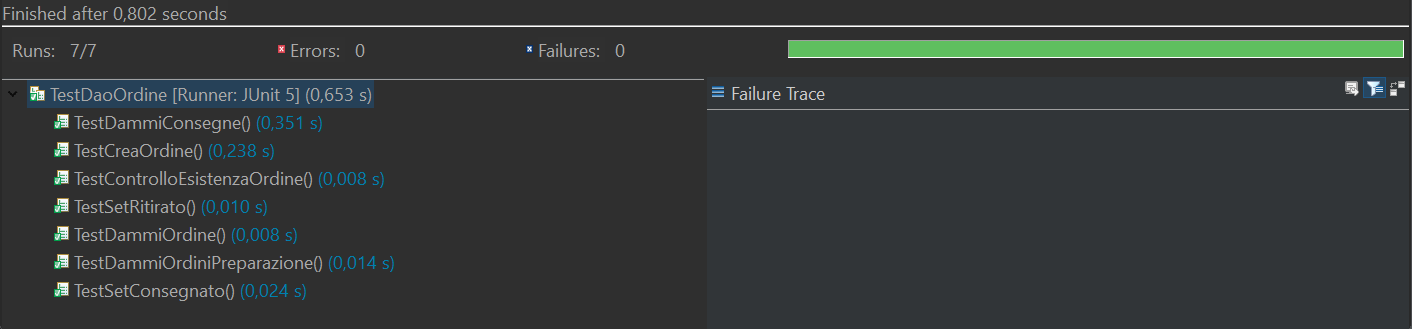
****

**DoRegistrazioneFattorinoTest**

****

* 1. **Test delle classi DAO**

**TestDaoOrdine**



**TestDaoUtente**

