



# TSR: Test Summary Report

Transport Efficiency Manager

Riferimento	
Versione	1.3
Data	04/12/2020
Destinatario	Prof.ssa F. Ferrucci
Presentato da	Team NC08
Approvato da	

## Revision History

---

Data	Versione	Descrizione	Autori
10/03/2021	1.1	Prima stesura	Francesca Moschella
06/06/2021	1.2	Aggiunta	Federica Attianese, Federica Pica
12/06/2021	1.3	Aggiornamenti	Francesca Moschella, Federica Attianese, Federica Pica
16/06/2021	1.4	Revisione e ultimazione	Federica Attianese, Francesca Moschella, Federica Pica

## Indice dei contenuti

1.Introduzione.....	4
1.1 Acronimi ed Abbreviazioni .....	4
1.2 Riferimenti.....	4
1.3 Panoramica.....	4
2. Relazione con altri documenti di testing .....	5
2.1 Relazione con il Test Plan (TP) .....	5
2.2 Relazione con il Test Case Document (TCP).....	5
2.3 Relazione con il Test Incident Report (TIR).....	5
3.Analisi dei test cases .....	5
4.Risultati con JUnit per il controller .....	5
4.1 Features testate.....	5
4.2 Panoramica dei risultati del test delle classi .....	6
4.2.1 AccountController .....	6
4.2.2 CorsaController .....	6
4.2.3 DatiCorsaController .....	6
4.2.4 ProgrammaCorseController .....	6
4.2.5 RisorseController.....	7
5.Risultati con JUnit per il model .....	7
5.1 Features testate.....	7
5.2 Panoramica dei risultati del test delle classi .....	7
5.2.1 AccountService .....	7
5.2.2 CorsaService .....	8
5.2.3 ProgrammaCorseService .....	8
5.2.4 DatiCorsaService .....	8
5.2.5 RisorseService .....	8
5.2.6 ProgrammaAutomaticoMaker .....	9
5.2.6.1 ProgrammaAutomaticoMakerConducenteTest .....	9
5.2.6.2 ProgrammaAutomaticoMakerMezzoTest.....	9
5.2.6.3 ProgrammaAutomaticoMakerOtherTest .....	9
6.Coverage .....	9
7.Riepilogo del testing .....	10
8.Glossario .....	11

## 1. Introduzione

Dopo aver puntato l'attenzione alla specifica dei diversi test case descritti nel Test Case Document, si passa alla rendicontazione dei risultati ottenuti dalle effettive attività di testing effettuate su di essi. Come già specificato nel Test Integration Document, per la fase di testing sono stati utilizzati gli strumenti JUnit e Mockito, successivamente sarà invece usato il tool Katalon Studio per la fase di testing di sistema.

Una prima analisi sarà effettuata sui difetti riscontrati nell'applicazione, ovvero un insieme di valutazioni rispetto ai fallimenti o gli errori riscontrati nei test, che necessitano di ulteriori approfondimenti.

Un report significativo delle attività di testing svolte è necessario, per poter portare a conoscenza di chi usufruirà della piattaforma, di eventuali failure o bug presenti nel sistema e permettere agli sviluppatori di poter giungere a soluzioni a tali limiti riscontrati nei test.

### 1.1 Acronimi ed Abbreviazioni

**TP:** Test Plan

**TCD:** Test Case Document

**TID:** Test Integration Document

**TSR:** Test Summary Report

### 1.2 Riferimenti

- RAD\_v2.1.2.docx
- SDD\_v1.1.docx
- ODD\_v1.3.docx
- TCD\_v.1.1.docx
- B.Bruegge, A.H. Dutoit, Object Oriented Software Engineering – Using UML, Patterns and Java, Prentice Hall.
- Slides del corso, presenti sulla piattaforma e-learning

### 1.3 Panoramica

Nella sezione successiva (2. Relazione con altri documenti di testing) sarà mostrato come questo documento si relaziona con il resto della documentazione prodotta durante le attività di testing, evidenziando come questo ne utilizza i risultati ottenuti.

Nella sezione 3 vengono definiti alcuni dati relativamente ai test case pianificati, implementati ed eseguiti.

Nelle sezioni 4 e 5 vengono riportati e specificati i risultati ottenuti, gli eventuali bug riscontrati durante l'esecuzione dei test ed elencate tutte le features testate rispettivamente per il model ed il controller.

## 2. Relazione con altri documenti di testing

In questa sezione sono riportate le relazioni che il seguente documento ha con altri documenti prodotti durante la fase di testing.

### 2.1 Relazione con il Test Plan (TP)

Il Test Plan sarà utilizzato per ricavare le componenti e le funzionalità che devono essere testate.

### 2.2 Relazione con il Test Case Document (TCP)

Il Test Case Document verrà utilizzato per ricavare (dalla sezione Test Case Specification) le specifiche dei test cases che saranno usate per testare le funzionalità del sistema.

### 2.3 Relazione con il Test Incident Report (TIR)

Nel Test Incident Report saranno riportate tutte le anomalie riscontrate durante il testing delle funzionalità effettuato in questa fase.

## 3. Analisi dei test cases

Le funzionalità del sistema prese in esame per la fase di testing sono state elencate nella sezione 4 del Test Plan; all'interno del Test Case Document, invece, sono state definite le varie combinazioni per i possibili input all'interno del sistema attraverso la tecnica del category partition. Successivamente, sono stati specificati in dettaglio i vari test case con il relativo comportamento atteso (oracolo).

Per il Black-box testing erano stati pianificati esattamente 80 e la quasi totalità di questi sono stati implementati. Inoltre, di questi 80 casi di test, quelli relativi alle operazioni di inserimento, sono stati riutilizzati per testare le rispettive operazioni di modifica, incrementando il numero di test effettuati.

Il team ha deciso di testare con il metodo White-box i metodi che, al loro interno, non prevedono l'utilizzo di form. Quindi, questi ulteriori test effettuati, non prevedono casi di test all'interno del Test Case Document.

Per quanto riguarda le classi <Entity>, queste non sono comprese tra le features testate del package model, in quanto contengono esclusivamente basilari metodi getter/setter.

## 4. Risultati con JUnit per il controller

Di seguito sono riportati i risultati dei test di unità per le classi del package controller.

### 4.1 Features testate

Sono state testate:

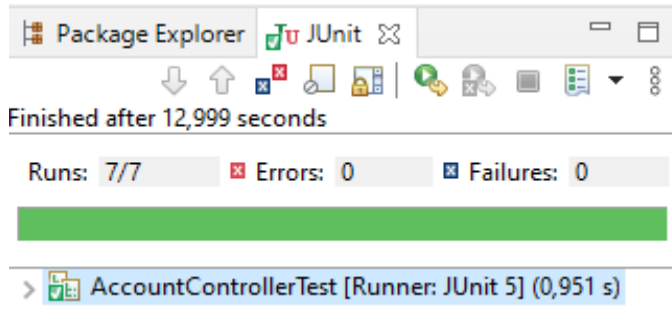
- AccountController
- CorsaController

- DatiCorsaController
- ProgrammaCorseController
- RisorseController

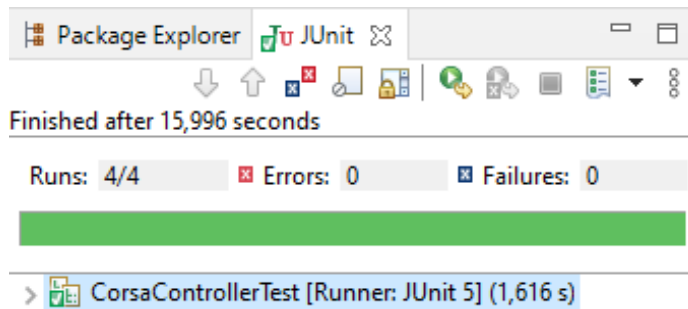
## 4.2 Panoramica dei risultati del test delle classi

Di seguito sono riportati i risultati dell'esecuzione di ogni classe.

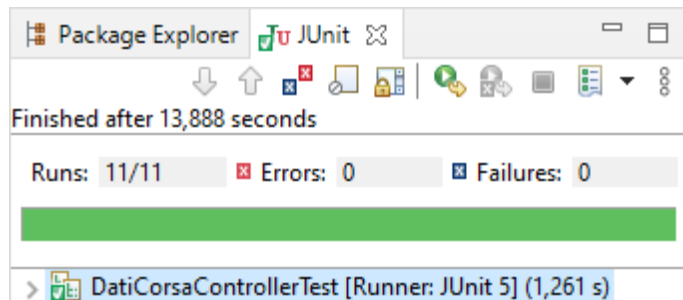
### 4.2.1 AccountController



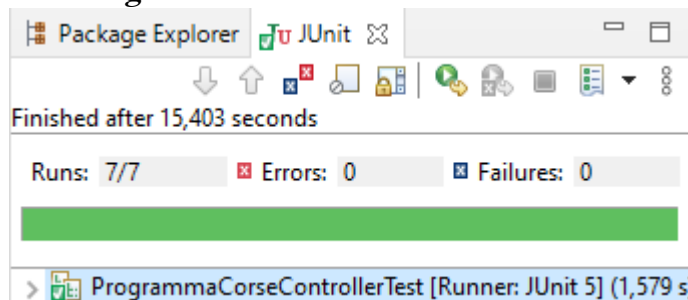
### 4.2.2 CorsaController



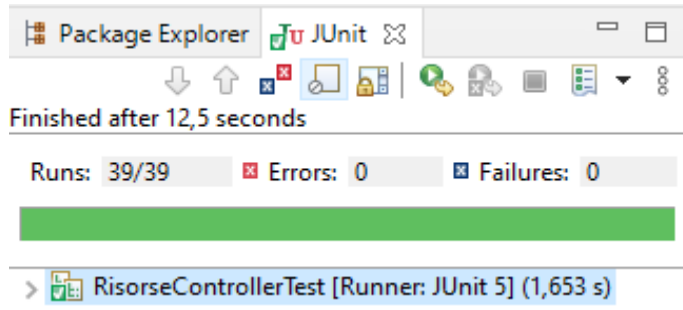
### 4.2.3 DatiCorsaController



### 4.2.4 ProgrammaCorseController



#### 4.2.5 RisorseController



## 5. Risultati con JUnit per il model

Di seguito sono riportati i risultati dei test di unità per le classi del package model.

### 5.1 Features testate

Il testing di unità si propone di effettuare il test delle classi presenti all'interno del Package model.

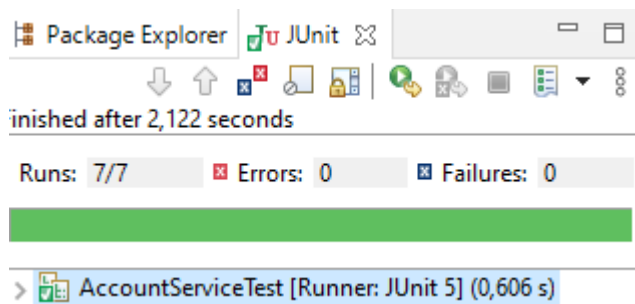
In particolare, sono state eseguite:

- AccountService
- CorsaService
- ProgrammaCorseService
- DatiCorseService
- RisorseService
- ProgrammaAutomaticoMaker (dato l'elevato numero di test, il team ha deciso di suddividerli nelle seguenti classi di test: ProgrammaAutomaticoMakerConducenteTest, ProgrammaAutomaticoMakerMezzoTest, ProgrammaAutomaticoMakerOtherTest)

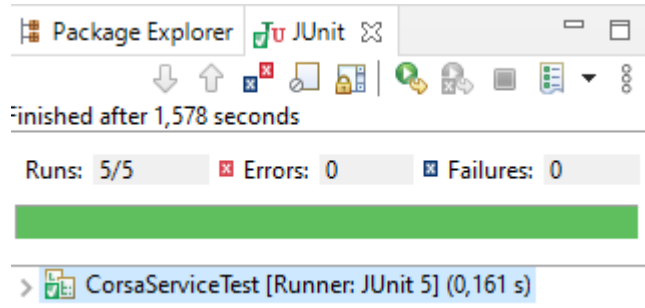
### 5.2 Panoramica dei risultati del test delle classi

Di seguito sono riportati i risultati dell'esecuzione di ogni classe.

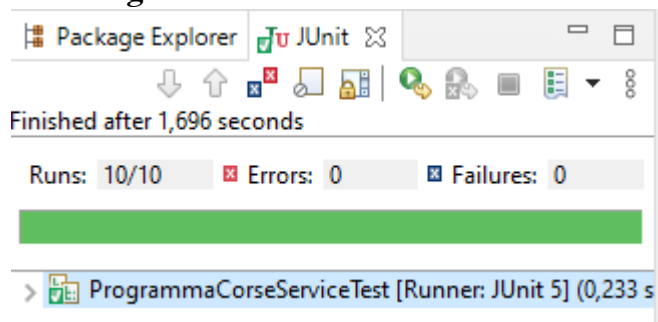
#### 5.2.1 AccountService



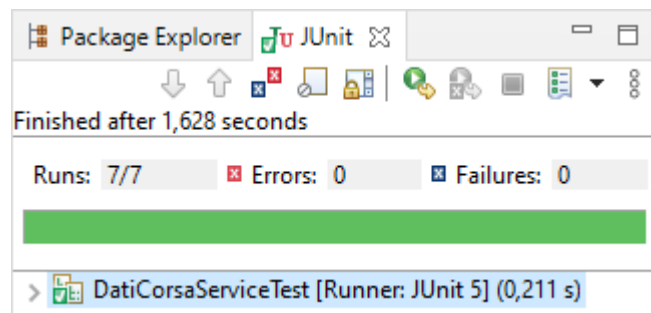
### 5.2.2 CorsaService



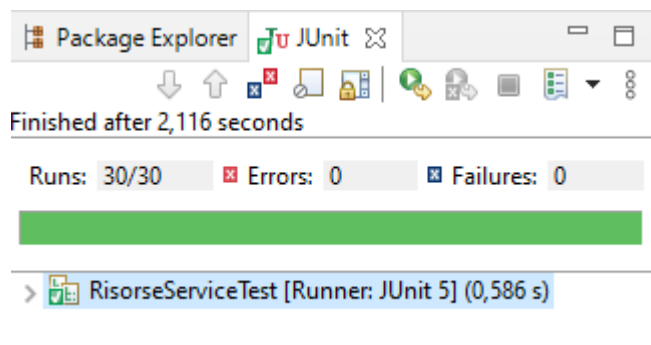
### 5.2.3 ProgrammaCorseService



### 5.2.4 DatiCorsaService



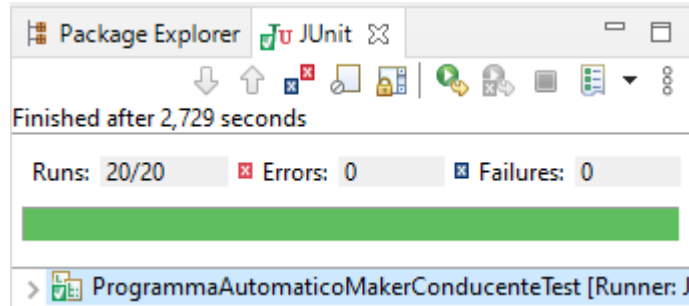
### 5.2.5 RisorseService



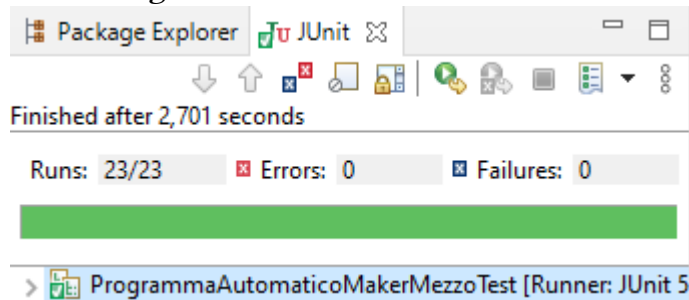


### 5.2.6 *ProgrammaAutomaticoMaker*

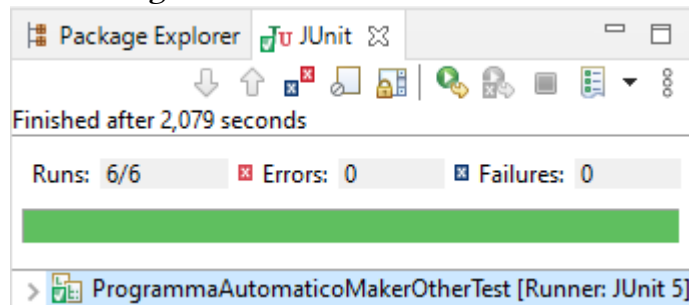
- 5.2.6.1 *ProgrammaAutomaticoMakerConducenteTest*



- 5.2.6.2 *ProgrammaAutomaticoMakerMezzoTest*






- 5.2.6.3 *ProgrammaAutomaticoMakerOtherTest*























## 6. Coverage

Di seguito riportati i dati riferiti alla percentuale di coverage raggiunta.







tem (16 giu 2021 12:19:13)				
Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼ tem	 91,6 %	14.111	1.296	15.407
> src/main/java	 71,6 %	2.835	1.124	3.959
> src/test/java	 98,5 %	11.276	172	11.448

tem (16 giu 2021 12:19:13)

Element	Coverage	Covered Instructio...	Missed Instructions	Total Instructions
▼ tem	 91,6 %	14.111	1.296	15.407
▼ src/main/java	 71,6 %	2.835	1.124	3.959
> com.java.tem	 90,4 %	113	12	125
> com.java.tem.aimodule	 70,0 %	749	321	1.070
> com.java.tem.controller	 58,4 %	925	658	1.583
> com.java.tem.exceptions	 87,5 %	28	4	32
> com.java.tem.model.accountservice.entity	 76,1 %	239	75	314
> com.java.tem.model.programmacorsesev	 88,5 %	208	27	235
> com.java.tem.model.programmacorsesev	 100,0 %	122	0	122
> com.java.tem.model.programmacorsesev	 97,6 %	403	10	413
> com.java.tem.model.programmacorsesev	 73,8 %	48	17	65
▼ src/test/java	 98,5 %	11.276	172	11.448
> com.java.tem	 100,0 %	4	0	4
> com.java.tem.aimodule	 100,0 %	4.628	0	4.628
> com.java.tem.controller	 100,0 %	5.316	0	5.316
> com.java.tem.katalon	 0,0 %	0	166	166
> com.java.tem.model.accountservice.entity	 97,7 %	259	6	265
> com.java.tem.model.programmacorsesev	 100,0 %	366	0	366
> com.java.tem.model.programmacorsesev	 100,0 %	139	0	139
> com.java.tem.model.programmacorsesev	 100,0 %	564	0	564

## Coverage

Session: tem (16 giu 2021 12:19:13)

Counter	Coverage	Covered	Missed	Total
Instructions	 71,6 %	2.835	1.124	3.959
Branches	 82,5 %	179	38	217
Lines	 74,1 %	742	260	1.002
Methods	 75,1 %	260	86	346
Types	 92,5 %	37	3	40
Complexity	 76,0 %	346	109	455

## 7. Riepilogo del testing

	Numero di componenti testate	Numero di errori trovati	Numero di errori corretti	Numero di componenti non testate
controller	5	2	2	2
model	8	3	3	2

## 8. Glossario

**Test Case Document:** Il Test Case Document descrive quali scenario saranno testati, come verranno effettuati i test e quanto spesso.

**Test Integration Document:** Il Test Integration Document specifica la strategia utilizzata e le componenti testate durante la fase di test di integrazione.

**Test Incident Report:** documento di testing che riporta la descrizione di un incidente osservato durante la fase di testing.

**Test Summary Report:** Il Test Summary Report è il documento che contiene il resoconto delle attività di testing ed i relativi risultati finali fornendo una valutazione relativamente all'esecuzione dei test.

**JUnit:** framework per Java utilizzato al fine di automatizzare il testing.

**Mockito:** framework per Java utilizzato per emulare delle componenti di una classe al fine di eseguire il testing.

**Katalon:** estensione web che permette di registrare le azioni da effettuare al fine di automatizzare il testing di sistema.

**RAD** (Requirement Analysis Document): Documento di Raccolta e analisi dei Requisiti; contiene l'elenco dei requisiti funzionali e non funzionali individuati in fase di individuazione degli stessi e la loro analisi sotto forma di scenari e casi d'uso. I mock-up mostrano una possibile implementazione dell'interfaccia del sistema.

**SDD** (System Design Document): Documento all'interno del quale viene riportata la progettazione del sistema come risultato di una prima fase di modellazione: contiene una suddivisione ad alto livello del sistema nei sottosistemi che lo comporranno.

**ODD** (Object Design Document): Documento che riporta e analizza gli oggetti che compongono il sistema analizzando le componenti a più basso livello, riportandole così come saranno implementate.