



Test Cases Document - HeavyRoute

Versione 1.0

Informazioni Generali

Corso di Laurea:	Informatica
Università:	Università degli Studi di Salerno (UNISA)
Docente:	Chiar.mo Prof. DE LUCIA Andrea
Data:	22/12/2025
Anno Accademico:	2025/2026

Membri Del Gruppo

MANFREDINI Umberto	Matricola 0512119797
MANZO Ugo	Matricola 0512119071 (<i>Coordinatore</i>)
ROMANO Pino Fiorello	Matricola 0512120259

Revision History

Data	Versione	Descrizione	Autore
22/12/2025	1.0	Creazione del Test Case Doc.	Pino Fiorello Romano
—	—	—	—
—	—	—	—

Indice

1	Analisi Funzionale: Creazione Richiesta di Trasporto	4
1.1	Unità Funzionale	4
1.2	Definizione delle Categorie e Scelte	4
2	Test Case 1: Creazione Richiesta Valida	5
2.1	1. Test case specification identifier	5
2.1.1	2. Test items	5
2.2	3. Input specifications	5
2.3	4. Output specifications (Oracle)	5
2.4	5. Environmental needs	6
2.5	6. Special procedural requirements	6
2.6	7. Intercase dependencies	6
3	Test Case 2: Validazione Input (Error Cases)	7
3.1	1. Test case specification identifier	7
3.2	2. Test items	7
3.3	3. Input specifications	7
3.4	4. Output specifications (Oracle)	7
3.5	5. Environmental needs	8
3.6	6. Special procedural requirements	8
3.7	7. Intercase dependencies	8
4	Analisi Funzionale: Approvazione Richiesta	9
4.1	Unità Funzionale	9
4.2	Definizione delle Categorie	9
5	Test Case 3: Approvazione Valida	9
5.1	1. Test case specification identifier	9
5.2	2. Test items	9
5.3	3. Input specifications	9
5.4	4. Output specifications (Oracle)	10
5.5	5. Environmental needs	10
5.6	6. Special procedural requirements	10
5.7	7. Intercase dependencies	10
6	Analisi Funzionale: Assegnazione Risorse	11
6.1	Unità Funzionale	11
6.2	Definizione delle Categorie	11

7	Test Case 4: Assegnazione Valida	11
7.1	1. Test case specification identifier	11
7.2	2. Test items	11
7.3	3. Input specifications	11
7.4	4. Output specifications (Oracle)	12
7.5	5. Environmental needs	12
7.6	6. Special procedural requirements	12
7.7	7. Intercase dependencies	12
8	Test Case 5: Gestione Conflitto Risorse	12
8.1	1. Test case specification identifier	12
8.2	2. Test items	12
8.3	3. Input specifications	12
8.4	4. Output specifications (Oracle)	13
8.5	5. Environmental needs	13
8.6	6. Special procedural requirements	13
8.7	7. Intercase dependencies	13
9	Analisi Funzionale: Controllo Accessi (Approvazione)	14
9.1	Unità Funzionale	14
9.2	Definizione delle Categorie	14
10	Test Case 6: Accesso Non Autorizzato	14
10.1	1. Test case specification identifier	14
10.2	2. Test items	14
10.3	3. Input specifications	14
10.4	4. Output specifications (Oracle)	14
10.5	5. Environmental needs	15
10.6	6. Special procedural requirements	15
10.7	7. Intercase dependencies	15

2. Test Case 1: Creazione Richiesta Valida

2.1. 1. Test case specification identifier

TC-CORE-01 (Test Frame: Tutte Scelte Valide)

2.1.1. 2. Test items

- **Unit Under Test (UUT):** Classe ViaggioService.
- **Method:** creaRichiesta(RichiestaDTO dto, Long clientId).
- **Feature:** Creazione e persistenza di una nuova richiesta (UC3).

2.2. 3. Input specifications

Combinazione Scelte: Origine[Valida] AND Destinazione[Valida] AND Peso[Positivo] AND Data[Futura] AND Committente[Esistente].

Valori Concreti:

- dto:

```
{
    "origine": "Napoli, Via Roma 1",
    "destinazione": "Roma, Via Tiburtina 10",
    "peso": 1500.0,
    "dataRitiro": "2026-06-30"
}
```

- clientId: 1L.

Stato del Mock:

- UserRepository configurato per restituire un oggetto User valido.

2.3. 4. Output specifications (Oracle)

Il test è considerato superato se:

1. Il metodo restituisce un oggetto RichiestaTrasporto non nullo.
2. oggetto.getId() è valorizzato.
3. oggetto.getStato() è IN_ATTESA.
4. Viene invocato save() sul repository.

2.4. 5. Environmental needs

JUnit 5, Mockito.

2.5. 6. Special procedural requirements

Configurare i mock (`when(...).thenReturn(...)`) prima dell'esecuzione.

2.6. 7. Intercase dependencies

Nessuna.

3. Test Case 2: Validazione Input (Error Cases)

3.1. 1. Test case specification identifier

TC-CORE-02 (Test Frame: Scelte [error] Input)

3.2. 2. Test items

- **Unit Under Test (UUT):** ViaggioController.
- **Feature:** Validazione degli input (Bean Validation).

3.3. 3. Input specifications

Questo test copre molteplici combinazioni di errore identificate nell'analisi Category Partition.

Scenario A: Peso Negativo

- **Combinazione:** Origine[Valida] AND Peso[Zero o Negativo].
- **Payload:**

```
{ "origine": "Napoli", "destinazione": "Roma",  
  "peso": -50.0, "dataRitiro": "2026-06-30" }
```

Scenario B: Data Passata

- **Combinazione:** Origine[Valida] AND Data[Passata].
- **Payload:**

```
{ "origine": "Napoli", "destinazione": "Roma",  
  "peso": 100.0, "dataRitiro": "2020-01-01" }
```

3.4. 4. Output specifications (Oracle)

Per entrambi gli scenari, l'oracolo atteso è:

1. **Status Code:** 400 Bad Request.
2. **Body:** Contiene un messaggio di errore specifico per il campo invalido (es. "must be greater than 0").
3. **Isolamento:** Il metodo del Service **NON** deve essere invocato.

3.5. 5. Environmental needs

MockMvc (Spring Boot Test).

3.6. 6. Special procedural requirements

Usare `@WebMvcTest` per testare solo il layer di validazione.

3.7. 7. Intercase dependencies

Nessuna.

4. Analisi Funzionale: Approvazione Richiesta

4.1. Unità Funzionale

La funzionalità testata è l'approvazione di una richiesta da parte del PL e la generazione del viaggio (UC4).

- **Input:** requestId (Long).
- **Stato Sistema:** Presenza e stato della richiesta nel database.

4.2. Definizione delle Categorie

Parametro: Esistenza Richiesta • Esiste nel DB

- Non esiste [error]

Parametro: Stato Corrente Richiesta • IN_ATTESA (Stato valido per approvazione)

- GIA_APPROVATA [error]
- RIFIUTATA [error]

5. Test Case 3: Approvazione Valida

5.1. 1. Test case specification identifier

TC-CORE-03 (Test Frame: Richiesta Esistente + Stato Valido)

5.2. 2. Test items

- **Unit Under Test:** ViaggioService.approvaEGeneraViaggio.

5.3. 3. Input specifications

- **Argomento:** requestId = 10L.
- **Precondizione Mock:** Il repository restituisce una richiesta con ID 10 e stato IN_ATTESA.

5.4. 4. Output specifications (Oracle)

1. **Ritorno:** Oggetto Viaggio non nullo in stato IN_PIANIFICAZIONE.
2. **Side Effect:** Lo stato della richiesta nel mock è aggiornato a APPROVATA.
3. **Persistenza:** `tripRepository.save()` invocato 1 volta.

5.5. 5. Environmental needs

JUnit 5, Mockito.

5.6. 6. Special procedural requirements

Verifica transazionalità (rollback in caso di errore).

5.7. 7. Intercase dependencies

Nessuna.

6. Analisi Funzionale: Assegnazione Risorse

6.1. Unità Funzionale

Funzionalità di associazione di risorse (Autista e Veicolo) a un viaggio pianificato (UC4).

- **Input:** ID Viaggio, ID Autista, Targa Veicolo.
- **Condizioni Ambientali:** Stato delle risorse nel sistema esterno.

6.2. Definizione delle Categorie

Parametro: Stato Viaggio

- IN_PIANIFICAZIONE (Valido)

- Altro stato (es. CONFERMATO) [**error**]

Ambiente: Disponibilità Autista

- Disponibile (Service restituisce true)

- Non Disponibile (Service restituisce false) [**error**]

Ambiente: Disponibilità Veicolo

- Disponibile (Service restituisce true)

- Non Disponibile (Service restituisce false) [**error**]

7. Test Case 4: Assegnazione Valida

7.1. 1. Test case specification identifier

TC-CORE-04 (Test Frame: Tutte Risorse Disponibili)

7.2. 2. Test items

- **Unit Under Test:** ViaggioService.assegnaRisorse.

7.3. 3. Input specifications

- **Argomenti:** tripId=50L, driverId=200L, vehicleId=300L.
- **Precondizione Mock:**
 - Viaggio 50 esiste ed è IN_PIANIFICAZIONE.
 - ResourceService restituisce true per entrambi.

7.4. 4. Output specifications (Oracle)

1. Nessuna eccezione.
2. Viaggio aggiornato con Driver 200 e Veicolo 300.
3. `tripRepository.save()` invocato.

7.5. 5. Environmental needs

JUnit 5, Mockito.

7.6. 6. Special procedural requirements

Nessuno.

7.7. 7. Intercase dependencies

TC-CORE-03.

8. Test Case 5: Gestione Conflitto Risorse

8.1. 1. Test case specification identifier

TC-CORE-05 (Test Frame: Veicolo Non Disponibile [error])

8.2. 2. Test items

- **Unit Under Test:** `ViaggioService.assegnaRisorse`.
- **Feature:** Gestione eccezioni di business.

8.3. 3. Input specifications

- **Argomenti:** Stessi di TC4.
- **Precondizione Mock:**
 - `ResourceService.isVehicleAvailable` restituisce **false**.

8.4. 4. Output specifications (Oracle)

1. **Eccezione:** Viene lanciata `ResourceNotAvailableException`.
2. **Stato:** L'oggetto Viaggio nel DB **non** viene modificato.
3. **Persistenza:** `tripRepository.save()` **non** viene invocato.

8.5. 5. Environmental needs

JUnit 5, Mockito.

8.6. 6. Special procedural requirements

Uso di `assertThrows`.

8.7. 7. Intercase dependencies

Nessuna.

9. Analisi Funzionale: Controllo Accessi (Approvazione)

9.1. Unità Funzionale

Endpoint per l'approvazione del viaggio (UC5).

- **Input:** Token JWT (Ruolo).

9.2. Definizione delle Categorie

Parametro: Ruolo Utente • ROLE_PL (Valido)

- ROLE_COMMITTENTE [error]
- ROLE_AUTISTA [error]
- ROLE_TC [error]
- ROLE_GA [error]
- Nessun Ruolo (Visitatore) [error]

10. Test Case 6: Accesso Non Autorizzato

10.1. 1. Test case specification identifier

TC-CORE-06 (Test Frame: Ruolo Non Autorizzato [error])

10.2. 2. Test items

- **Unit Under Test:** ViaggioController (Security Layer).
- **Feature:** RBAC.

10.3. 3. Input specifications

- **Context:** Utente autenticato con ruolo COMMITTENTE.
- **Request:** POST /api/trips/10/approve.

10.4. 4. Output specifications (Oracle)

1. **Status:** 403 Forbidden.
2. **Service:** Nessuna invocazione a ViaggioService.

10.5. 5. Environmental needs

Spring Security Test, MockMvc.

10.6. 6. Special procedural requirements

Uso di @WithMockUser.

10.7. 7. Intercase dependencies

Nessuna.