



### *Progetto*

## Vendita biglietti per la compagnia di autobus “NetBus”



Gruppo: *NominaSuntSubstantiaRerum*

Studenti:

Alessandro	Mascolo	N46006699	<i>alessa.mascolo@studenti.unina.it</i>
Carmine	Squillace	N46006278	<i>c.squillace@studenti.unina.it</i>

Versione 1 del 23/07/2025

# Indice

<b>1. SPECIFICHE INFORMALI.....</b>	<b>3</b>
<b>2. ANALISI E SPECIFICA DEI REQUISITI .....</b>	<b>4</b>
2.1 ANALISI NOMI-VERBI .....	4
2.2 REVISIONE DEI REQUISITI.....	5
2.3 GLOSSARIO DEI TERMINI.....	6
2.4 CLASSIFICAZIONE DEI REQUISITI .....	7
2.4.1 Requisiti funzionali .....	7
2.4.2 Requisiti sui dati.....	7
2.4.3 Vincoli / Altri requisiti.....	8
2.5 MODELLAZIONE DEI CASI D'USO .....	9
2.5.1 Attori e casi d'uso.....	9
2.5.2 Diagramma dei casi d'uso .....	10
2.5.3 Scenari.....	10
2.6 MODELLAZIONE DEI DATI.....	13
2.6.1 Progettazione concettuale .....	13
2.7 DIAGRAMMA DELLE CLASSI.....	14
2.8 DIAGRAMMI DI SEQUENZA.....	15
2.9 VERIFICA DELLA COMPLETEZZA DEI REQUISITI.....	16
<b>3. STIMA DEI COSTI.....</b>	<b>17</b>
<b>4. PIANO DI TEST FUNZIONALE.....</b>	<b>20</b>
<b>5. PROGETTAZIONE .....</b>	<b>28</b>
5.1 PROGETTAZIONE DELLA BASE DI DATI.....	28
5.1.1 Progettazione logica .....	28
5.2 DIAGRAMMA DELLE CLASSI.....	30
5.3 DIAGRAMMI DI SEQUENZA.....	31
<b>6. IMPLEMENTAZIONE .....</b>	<b>32</b>
<b>7. TESTING .....</b>	<b>38</b>
7.1 TEST STRUTTURALE .....	38
7.1.1 Complessità ciclomatica.....	38
7.1.2 Test di unità .....	40
7.2 TEST FUNZIONALE.....	41

## 1. Specifiche informali

Sistema di vendita (online e in biglietteria) di biglietti per la compagnia di autobus “NetBus”

Si vuole realizzare un’applicazione per la compagnia NetBus che fornisce servizi di trasporto passeggeri operati da autobus su tratte regionali e interregionali. Ogni autobus ha un identificativo ed una capienza in termini di numero di posti disponibili per i passeggeri. Ogni persona può portare un solo bagaglio, che va stivato prima di salire a bordo.

La compagnia opera con i propri autobus su un insieme di tratte tra città italiane. Ogni tratta è caratterizzata da una coppia ordinata di località (partenza, destinazione). Per ogni tratta viene effettuata una sola corsa giornaliera.

Ogni corsa ha una città di partenza, una di destinazione, una data, un orario di partenza, un orario di arrivo, e un prezzo del biglietto prefissato. Le città, le tratte e le corse, col relativo prezzo, sono inserite nel sistema dal direttore vendite.

Nelle stazioni degli autobus nelle località ci sono biglietterie della compagnia che possono emettere biglietti validi per una specifica corsa (tratta, data). Prima di emettere un biglietto, il sistema deve valutare l’effettiva disponibilità di posti, e all’atto dell’emissione dovrà essere aggiornata la disponibilità residua di posti passeggeri.

L’applicazione supporta l’emissione di biglietti da parte degli impiegati delle biglietterie presso i punti di partenza. Ogni impiegato ha un codice identificativo ed una password per accedere al sistema. Per ogni biglietto il sistema deve stampare la data, l’ora di emissione. Per ciascun biglietto il sistema deve registrare internamente, al momento dell’emissione, data e ora di emissione e codice dell’impiegato che lo ha emesso.

È altresì previsto l’uso dell’applicazione da parte di clienti registrati, che possono acquistare via web i biglietti, indicando il numero di posti richiesto, ed effettuando il pagamento con una carta di credito.

Alla fine di ogni mese il sistema genera ed invia automaticamente per e-mail al direttore vendite un report con il numero di biglietti venduti per ogni tratta, ordinati per città di partenza.

## 2. Analisi e specifica dei requisiti

### 2.1 Analisi nomi-verbi

Si vuole realizzare un'applicazione per la compagnia NetBus che fornisce servizi di trasporto passeggeri operati da autobus su tratte regionali e interregionali. Ogni **autobus** ha un **identificativo** ed una **capienza** in termini di numero di posti disponibili per i passeggeri. Ogni persona può portare un solo bagaglio, che va stivato prima di salire a bordo.

La compagnia opera con i propri autobus su un insieme di **tratte** tra **città** italiane. Ogni tratta è caratterizzata da una coppia ordinata di **località** (partenza, destinazione). Per ogni tratta viene effettuata una sola **corsa** giornaliera.

Ogni corsa ha una **città di partenza**, **una di destinazione**, una **data**, un **orario di partenza**, un **orario di arrivo**, e un **prezzo** del **biglietto** prefissato. Le città, le tratte e le corse, col relativo prezzo, sono inserite nel sistema dal **direttore vendite**.

Nelle stazioni degli autobus nelle località ci sono biglietterie della compagnia che possono emettere biglietti validi per una specifica corsa (tratta, data). Prima di emettere un biglietto, il sistema deve valutare l'effettiva disponibilità di posti, e all'atto dell'emissione dovrà essere aggiornata la disponibilità residua di posti passeggeri.

L'applicazione supporta l'**emissione di biglietti** da parte degli **impiegati delle biglietterie** presso i punti di partenza. Ogni impiegato ha un **codice identificativo** ed una **password** per accedere al sistema. Per ogni biglietto il sistema deve stampare la data, l'ora di emissione. Per ciascun biglietto il sistema deve registrare internamente, al momento dell'emissione, **data e ora di emissione** e **codice dell'impiegato** che lo ha emesso.

È altresì previsto l'uso dell'applicazione da parte di **clienti registrati**, che possono **acquistare via web i biglietti**, indicando il numero di posti richiesto, ed effettuando il pagamento con una **carta di credito**.

Alla fine di ogni mese il sistema genera ed invia automaticamente per e-mail al **direttore vendite** un **report** con il numero di biglietti venduti per ogni tratta, ordinati per **città di partenza**.

#### LEGENDA:

**Classe**

**Funzionalità**

**Attributo**

**Attore**

**Classe-Attore**

## 2.2 Revisione dei requisiti

1. *Ogni autobus ha un proprio identificativo.*
2. *Ogni autobus ha una propria capienza.*
3. *Ogni tratta è caratterizzata da una città di partenza.*
4. *Ogni tratta è caratterizzata da una città di destinazione.*
5. *Per ogni tratta si effettua una sola corsa al giorno.*
6. *Ogni corsa ha una città di partenza.*
7. *Ogni corsa ha una città di destinazione.*
8. *Ogni corsa ha una data.*
9. *Ogni corsa ha un orario di partenza.*
10. *Ogni corsa ha un orario di arrivo.*
11. *Ogni corsa ha un prezzo del biglietto prestabilito.*
12. *Il sistema deve offrire al direttore vendite una funzionalità per inserire le città.*
13. *Il sistema deve offrire al direttore vendite una funzionalità per inserire le tratte.*
14. *Il sistema deve offrire al direttore vendite una funzionalità per inserire le corse con il relativo costo del biglietto.*
15. *Ogni impiegato ha un proprio codice identificativo.*
16. *Ogni impiegato ha una propria password.*
17. *Il sistema deve consentire l'autenticazione degli impiegati della biglietteria.*
18. *Il sistema deve offrire agli impiegati delle biglietterie una funzionalità per l'emissione di biglietti validi per una data corsa.*
19. *Per ogni biglietto emesso, il sistema deve registrare internamente e stampare la data e l'ora di emissione ed il codice identificativo dell'impiegato che lo ha emesso.*
20. *Il sistema deve offrire ai clienti una funzionalità per registrarsi.*
21. *Di ogni cliente si vuole memorizzare nome utente, password, indirizzo e-mail e carta di credito.*
22. *Il sistema deve offrire ai clienti registrati una funzionalità per acquistare via web un biglietto.*
23. *Per effettuare l'acquisto di un biglietto il cliente registrato deve specificare il numero di posti richiesto ed una carta di credito.*
24. *Il sistema deve generare ed inviare automaticamente, per mail, alla fine di ogni mese, un report con il numero di biglietti venduti per ogni tratta, ordinati per città di partenza, al direttore vendite.*

## 2.3 Glossario dei termini

Termine	Descrizione	Sinonimi
Autobus	Mezzo di trasporto passeggeri della compagnia, impiegato per effettuare le corse delle tratte previste.	
Capienza	Numero massimo di posti disponibili per i passeggeri in un autobus.	Numero di posti disponibili
Passeggero	Generico cliente che acquista un biglietto e utilizza il mezzo di trasporto per spostarsi lungo una tratta.	Persona, Cliente
Città	Rappresenta uno dei due estremi di una tratta, identificabile come punto di partenza o di destinazione della tratta.	Località
Tratta	Percorso di viaggio definito tra una città di partenza e una di destinazione.	
Corsa	Singolo viaggio effettuato da un autobus della compagnia lungo una specifica tratta, in un determinato giorno ed orario.	Corsa Giornaliera
Biglietteria	Luogo dove i clienti possono recarsi di persona per acquistare un biglietto valido per una data corsa.	
Impiegato	Addetto alla biglietteria incaricato dell'emissione dei biglietti ai clienti.	Impiegato della biglietteria
Cliente registrato	Cliente che ha effettuato la procedura di registrazione.	
Report	Rapporto che fornisce un riepilogo strutturato dei biglietti venduti, ordinati per città di partenza, utilizzato per analisi e monitoraggio dal direttore vendite della compagnia.	

## 2.4 Classificazione dei requisiti

### 2.4.1 Requisiti funzionali

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RFo1	Il sistema deve offrire al direttore vendite una funzionalità per inserire le città.	12
RFo2	Il sistema deve offrire al direttore vendite una funzionalità per inserire le tratte.	13
RFo3	Il sistema deve offrire al direttore vendite una funzionalità per inserire le corse con il relativo costo del biglietto.	14
RFo4	Il sistema deve consentire l'autenticazione degli impiegati della biglietteria.	17
RFo5	Il sistema deve offrire agli impiegati delle biglietterie una funzionalità per l'emissione di biglietti validi per una data corsa.	18
RFo6	Per ogni biglietto emesso, il sistema deve registrare internamente e stampare la data e l'ora di emissione ed il codice identificativo dell'impiegato che lo ha emesso.	19
RFo7	Il sistema deve offrire ai clienti una funzionalità per registrarsi.	20
RFo8	Il sistema deve offrire ai clienti registrati una funzionalità per acquistare via web un biglietto.	22
RFo9	Il sistema deve generare ed inviare automaticamente, per mail, alla fine di ogni mese, un report con il numero di biglietti venduti per ogni tratta, ordinati per città di partenza, al direttore vendite.	24

### 2.4.2 Requisiti sui dati

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RD01	Ogni autobus ha un proprio identificativo.	1
RD02	Ogni autobus ha una propria capienza.	2
RD03	Ogni tratta è caratterizzata da una città di partenza.	3
RD04	Ogni tratta è caratterizzata da una città di destinazione.	4
RD05	Ogni corsa ha una città di partenza.	6
RD06	Ogni corsa ha una città di destinazione.	7

RD07	Ogni corsa ha una data.	8
RD08	Ogni corsa ha un orario di partenza.	9
RD09	Ogni corsa ha un orario di arrivo.	10
RD10	Ogni corsa ha un prezzo del biglietto prestabilito.	11
RD11	Ogni impiegato ha un proprio codice identificativo.	15
RD12	Ogni impiegato ha una propria password.	16
RD13	Di ogni cliente si vuole memorizzare nome utente, password, indirizzo e-mail e carta di credito.	21
RD14	Per effettuare l'acquisto di un biglietto il cliente registrato deve specificare il numero di posti richiesto ed una carta di credito.	23

#### 2.4.3 Vincoli / Altri requisiti

ID	Requisito	Origine (n. frase dei requisiti revisionati)
Vo1	Per ogni tratta si effettua una sola corsa al giorno.	5
RNF01	Per l'invio mensile del report deve essere disponibile un server di posta elettronica esterno al sistema.	24

## 2.5 Modellazione dei casi d'uso

### 2.5.1 Attori e casi d'uso

#### **Attori Primari:**

- Cliente registrato
- Cliente non registrato
- Direttore vendite
- Impiegato biglietteria
- Tempo

#### **Attori Secondari:**

- Servizio email

#### **Casi d'uso:**

- UC1: Invia report mensile
- UC2: Autenticazione
- UC3: Emetti biglietti
- UC4: Acquista biglietti
- UC5: Inserisci tratte
- UC6: Inserisci città
- UC7: Inserisci corse
- UC8: Registrazione

#### **Casi d'uso di inclusione:**

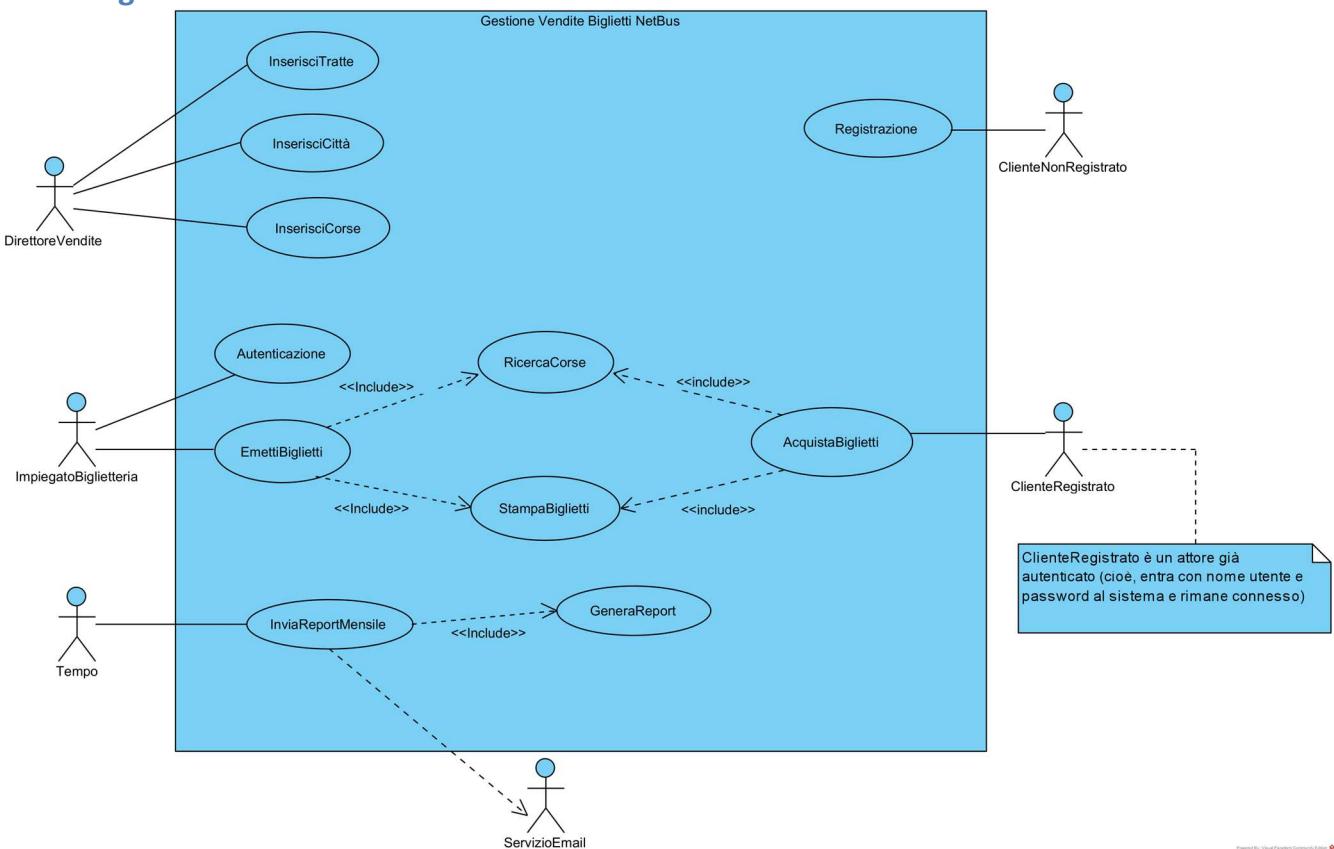
- UC9: Genera report
- UC10: Stampa biglietti

- UC11: Ricerca corse

Caso d'uso	Attori Primari	Attori Secondari	Incl. / Ext.	Requisiti corrispondenti
UC1: Invia report mensile	Tempo	Servizio email	Include Genera report	RF09
UC2: Autenticazione	Impiegato biglietteria	-	-	RF04
UC3: Emetti biglietti	Impiegato biglietteria	-	Include Ricerca corse, Stampa biglietti	RF05
UC4: Acquista biglietti	Cliente registrato	-	Include Ricerca corse, Stampa biglietti	RF08
UC5: Inserisci tratte	Direttore vendite	-	-	RF02
UC6: Inserisci città	Direttore vendite	-	-	RF01
UC7: Inserisci corse	Direttore vendite	-	-	RF03
UC8: Registrazione	Cliente non registrato	-	-	RF07
UC9: Genera report	Tempo	-	Incluso in Invia report mensile	RF09
UC10: Stampa biglietti	Impiegato biglietteria Cliente non registrato	-	Incluso in Emetti biglietti, Acquista biglietti	RF05, RF06, RF08

UC11: Ricerca corse	Impiegato biglietteria - Cliente non registrato	Incluso in Emetti biglietti, RF05, RF08 Acquista biglietti
---------------------	---	---

### 2.6.1 Diagramma dei casi d'uso



### 2.6.2 Scenari

Caso d'uso:	EmettiBiglietti
Attore primario	Impiegato biglietteria
Attore secondario	-
Descrizione	L'impiegato emette uno o più biglietti, ciascuno valido per una corsa che si svolge su una tratta specifica (città di partenza, città di destinazione), in una determinata data ed ora di partenza.
Pre-Condizioni	L'impiegato della biglietteria è stato autenticato.
Sequenza di eventi principale	<ol style="list-style-type: none"> <li>Il caso d'uso inizia quando l'Impiegato della biglietteria richiede l'emissione di biglietti.           <ol style="list-style-type: none"> <li>L'impiegato della biglietteria inserisce i criteri di ricerca per una corsa, ovvero la città di partenza, la città di destinazione, la data ed il numero di biglietti da emettere.</li> </ol> </li> <li>Il sistema verifica l'esistenza di una corsa che soddisfi i criteri specificati dall'impiegato. Se non esiste alcuna corsa che soddisfa i criteri specificati           <ol style="list-style-type: none"> <li>Il sistema comunica all'impiegato che non è stata trovata alcuna corsa che soddisfa i criteri specificati.</li> </ol> </li> </ol>

	<p>3. Il sistema calcola il prezzo per i biglietti.</p> <p>4. Se l'impiegato conferma l'ordine</p> <p>4.1 Il sistema verifica che ci siano sufficienti posti liberi disponibili per la corsa individuata. Se non ci sono sufficienti posti liberi disponibili</p> <p>4.1.1 Il sistema comunica all'impiegato che non ci sono sufficienti posti liberi disponibili per la corsa.</p> <p>4.2 Il sistema stampa i biglietti.</p> <p>5. Altrimenti</p> <p>5.1. Il sistema annulla l'emissione.</p>
<b>Post-Condizioni</b>	Il sistema registra internamente l'emissione, compreso l'identificativo dell'impiegato che ha emesso i biglietti. I biglietti sono stampati.
<b>Casi d'uso correlati</b>	RicercaCorse e StampaBiglietti

<b>Caso d'uso:</b>	<b>Registrazione</b>
<b>Attore primario</b>	Cliente non registrato
<b>Attore secondario</b>	-
<b>Descrizione</b>	Un cliente non registrato può registrarsi, creando un nuovo account, al fine di accedere alle funzionalità riservate ai clienti registrati, come l'acquisto di un biglietto valido per una corsa.
<b>Pre-Condizioni</b>	Il cliente non ha già un account associato.
<b>Sequenza di eventi principale</b>	<p>1. Il caso d'uso inizia quando un cliente non registrato richiede di registrarsi.</p> <p>2. Il sistema richiede l'inserimento di un nome utente, un indirizzo email ed una password.</p> <p>3. Il cliente inserisce le informazioni richieste.</p> <p>4. Il sistema effettua una validazione delle informazioni inserite.</p> <p>4.1. Finché la password è debole (ovvero è composta da meno di 8 caratteri oppure non contiene almeno un numero o un simbolo speciale)</p> <p>4.1.1. Il sistema informa il cliente dei requisiti minimi di sicurezza.</p> <p>4.1.2. Il cliente inserisce nuovamente la password.</p> <p>5. L'account viene creato.</p> <p>6. Il sistema comunica al cliente la conferma della registrazione.</p>
<b>Post-Condizioni</b>	Il cliente deve poter procedere all'autenticazione ed accedere alle funzionalità riservate, inclusa la possibilità di acquistare biglietti.
<b>Casi d'uso correlati</b>	-
<b>Sequenza di eventi alternativi</b>	<p>1. Se l'indirizzo email e/o il nome utente inseriti sono già associati ad un cliente registrato:</p> <p>1.1. Il sistema comunica che l'indirizzo email e/o il nome utente sono già utilizzati.</p> <p>1.2. Il cliente può correggere i dati e riprovare.</p>

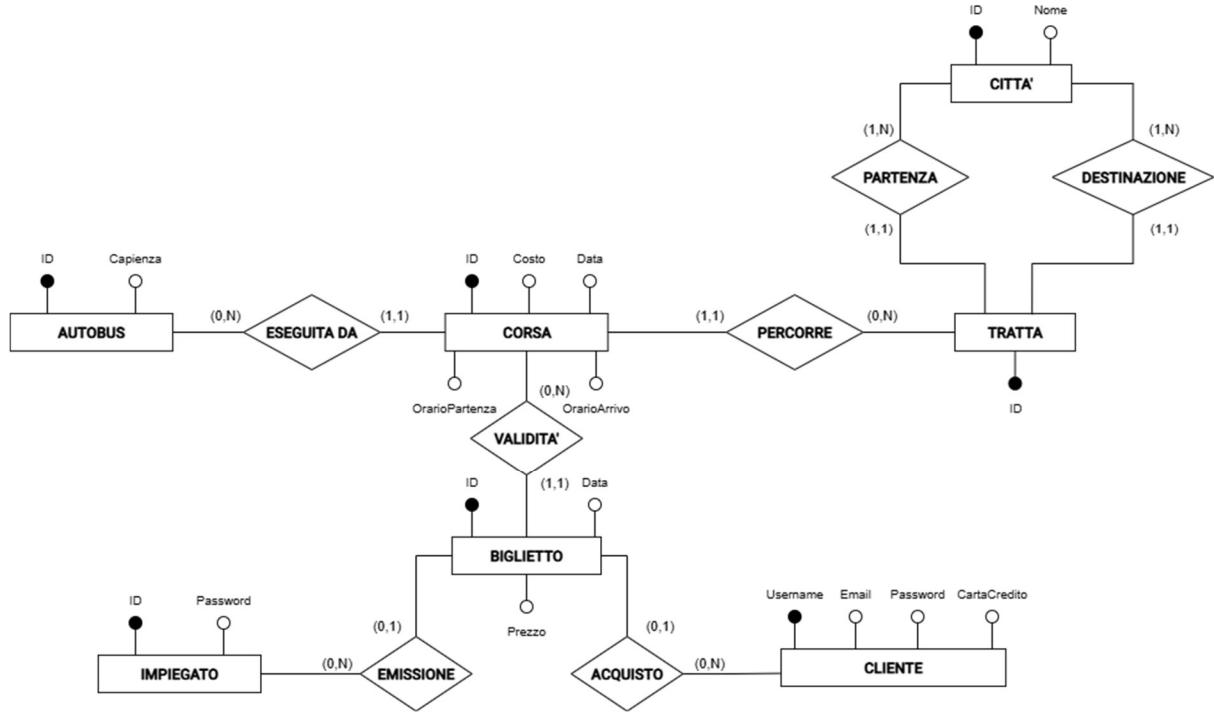
<b>Caso d'uso:</b>	<b>InviaReportMensile</b>
<b>Attore primario</b>	Tempo
<b>Attore secondario</b>	Servizio email
<b>Descrizione</b>	L'ultimo giorno di ogni mese viene inviato un report al direttore vendite

	con il numero di biglietti venduti per ogni tratta, ordinati per città di partenza.
<b>Pre-Condizioni</b>	È l'ultimo giorno di un mese.
<b>Sequenza di eventi principale</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso inizia l'ultimo giorno di un mese.</li> <li>2. Il sistema predisponde un messaggio di posta elettronica contenente il riepilogo del numero di biglietti venduti per ogni tratta, ordinati per città di partenza.</li> <li>3. Il sistema invia il messaggio predisposto e l'indirizzo email del direttore vendite al server di posta elettronica.</li> <li>4. Il server di posta elettronica invia il messaggio al direttore vendite.</li> </ol>
<b>Post-Condizioni</b>	Il direttore vendite riceve il report.
<b>Casi d'uso correlati</b>	GeneraReport
<b>Sequenza di eventi alternativi</b>	<ol style="list-style-type: none"> <li>1. Se il server di posta elettronica non è disponibile:             <ol style="list-style-type: none"> <li>1.1. Il sistema ritenta l'invio dopo 30 minuti.</li> </ol> </li> </ol>

<b>Caso d'uso:</b>	<b>AcquistaBiglietti</b>
<b>Attore primario</b>	Cliente registrato
<b>Attore secondario</b>	-
<b>Descrizione</b>	Un cliente registrato seleziona una corsa disponibile, prevista su una tratta specifica (composta da città di partenza e città di destinazione), in una determinata data e ora di partenza. Il pagamento avviene tramite carta di credito, che può essere facoltativamente memorizzata per acquisti futuri.
<b>Pre-Condizioni</b>	Il cliente è stato autenticato.
<b>Sequenza di eventi principale</b>	<ol style="list-style-type: none"> <li>1. Il caso d'uso inizia quando il Cliente registrato richiede l'acquisto di biglietti per una corsa.             <ol style="list-style-type: none"> <li>1.1. Il cliente inserisce il numero di biglietti per la corsa selezionata e il numero della carta di credito da utilizzare per il pagamento.</li> <li>2. Il sistema verifica la correttezza del numero di biglietti.                 <ol style="list-style-type: none"> <li>2.1. Se il numero di biglietti non è valido il sistema informa il cliente.</li> </ol> </li> <li>3. Se il cliente conferma l'ordine                 <ol style="list-style-type: none"> <li>3.1. Il sistema verifica la validità del metodo di pagamento. Se il numero della carta di credito non è corretto                     <ol style="list-style-type: none"> <li>3.1.1. Il sistema comunica al cliente che il metodo di pagamento non è valido.</li> </ol> </li> <li>3.2. Il sistema verifica che ci siano sufficienti posti liberi disponibili per la corsa individuata. Se non ci sono sufficienti posti liberi disponibili                     <ol style="list-style-type: none"> <li>3.2.1. Il sistema comunica al cliente che non ci sono sufficienti posti liberi disponibili per la corsa.</li> </ol> </li> </ol> </li> <li>4. Il sistema conferma l'acquisto al cliente.</li> </ol> </li> </ol>
<b>Post-Condizioni</b>	Il sistema registra internamente l'emissione. I biglietti sono stampati.
<b>Casi d'uso correlati</b>	RicercaCorse e StampaBiglietti

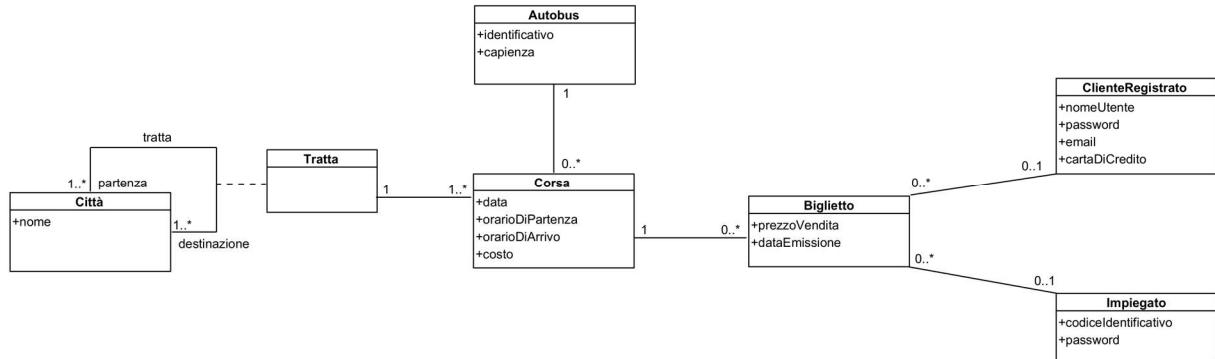
## 2.6 Modellazione dei dati

### 2.6.1 Progettazione concettuale



## 2.7 Diagramma delle classi

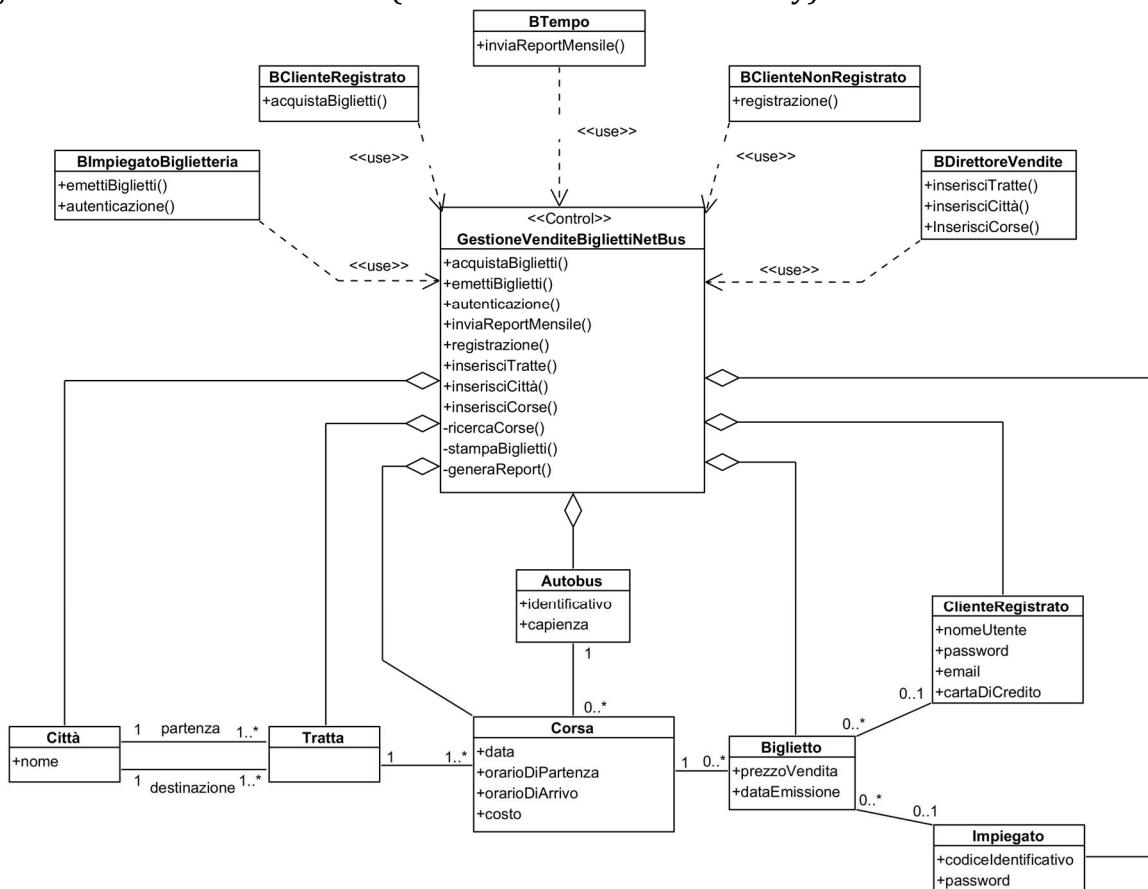
Diagramma delle classi di analisi.



Si noti che:

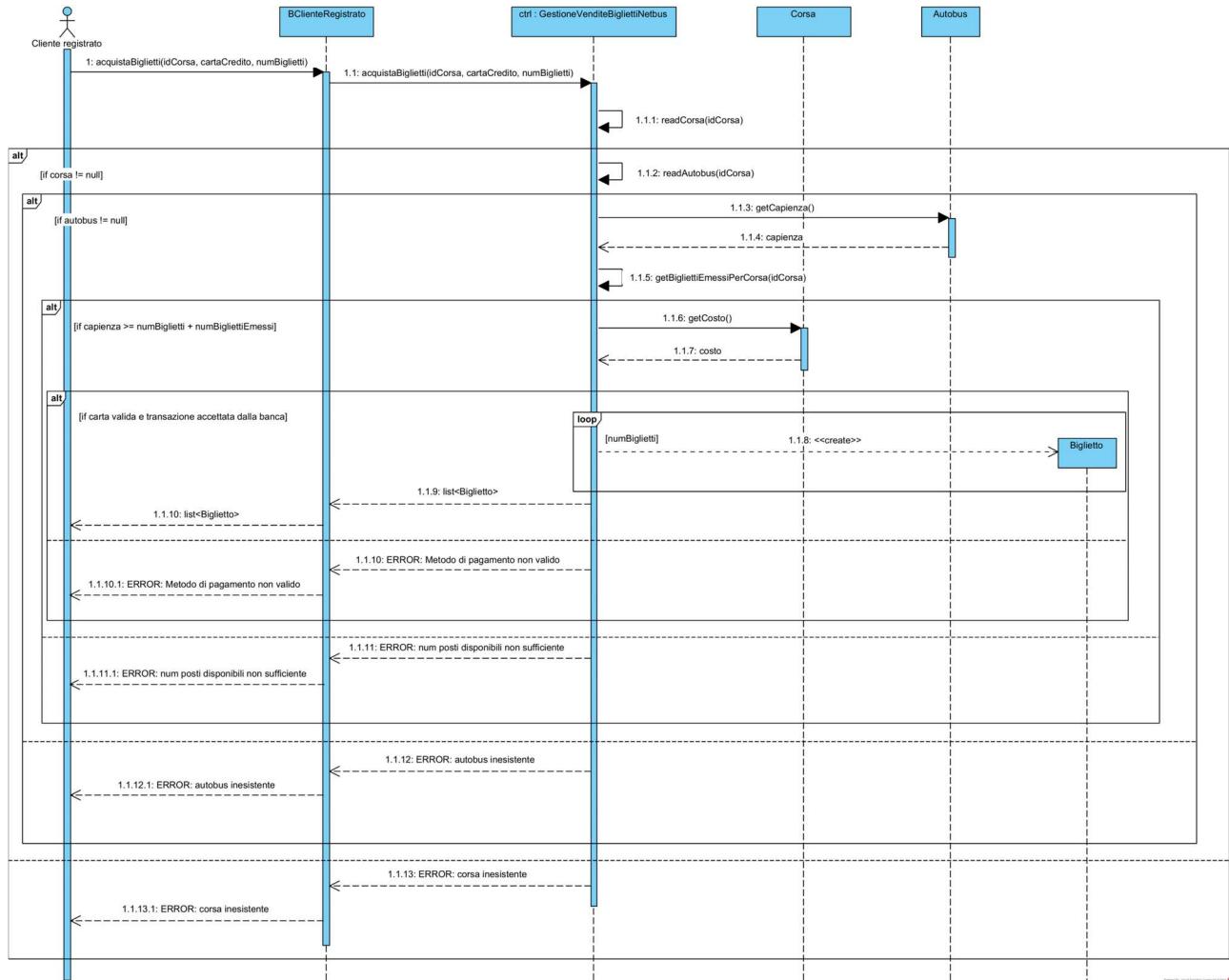
- Ogni biglietto è emesso *o* da un impiegato *o* da un cliente registrato, **non da entrambi contemporaneamente. È obbligatorio che uno dei due sia presente.**
- Un autobus **appena inserito** nel sistema **non ha ancora effettuato corse**. Ci possono essere **autobus di riserva, fuori servizio, o nuovi**. È quindi accettabile che **esistano autobus senza alcuna corsa associata**.
- Per ogni tratta si effettua **una sola corsa giornaliera**.
- Ogni tratta **collega direttamente due città** (partenza e destinazione), **senza effettuare fermate intermedie**.
- All'atto dell'acquisto di un biglietto, il cliente registrato può **scegliere se memorizzare la propria carta di credito per acquisti futuri**.

Diagramma delle classi raffinato (con classi Control e Boundary).



## 2.8 Diagrammi di sequenza

Diagramma di sequenza di analisi per il caso d'uso AcquistaBiglietti.



## 2.9 Verifica della completezza dei requisiti

Legenda: UCD = Use Case Diagram, CD = Class Diagram, SD = Sequence Diagram.

- **RF01** è modellato nell'UCD con l'attore "Direttore vendite" e con il caso d'uso UC6
- **RF02** è modellato nell'UCD con l'attore "Direttore vendite" e con il caso d'uso UC5
- **RF03** è modellato nell'UCD con l'attore "Direttore vendite" e con il caso d'uso UC7
- **RF04** è modellato nell'UCD con l'attore "Impiegato biglietteria" e con il caso d'uso UC2
- **RF05** è modellato nell'UCD con l'attore "Impiegato biglietteria" e i casi d'uso UC3, UC10, UC11
- **RF06** è modellato nell'UCD con il caso d'uso UC10
- **RF07** è modellato nell'UCD con l'attore "Cliente non registrato" e con il caso d'uso UC8
- **RF08** è modellato nell'UCD con l'attore "Cliente registrato" e i casi d'uso UC4, UC10, UC11
- **RF09** è modellato nell'UCD con gli attori "Tempo" (primario), "Servizio email" (secondario) e i casi d'uso UC1, UC9
- TODO:
- **RD01, RD02** sono modellati nel CD con gli attributi "identificativo" e "capienza" della classe "Autobus"
- **RD03, RD04, RD05, RD06** sono modellati nel CD con la classe "Tratta"
- **RD07, RD08, RD09 RD10** sono modellati nel CD con gli attributi della classe "Corsa"
- **RD11, RD12** sono modellati nel CD con gli attributi della classe "Impiegato"
- **RD13** è modellato nel CD con gli attributi della classe "ClienteRegistrato"
- **RD14** è modellato nel SD con i parametri della funzione "AcquistaBiglietti"

### 3. Stima dei costi

Si riporta di seguito la stima dei costi secondo il metodo dei Punti Funzione

- Tabella di riferimento per le complessità di dati e transazioni

	SEMPLICE	MEDIO	COMPLESSO
<b>NILF</b>	7	10	15
<b>NEIF</b>	5	7	10
<b>NEI</b>	3	4	6
<b>NEO</b>	4	5	7
<b>NEQ</b>	3	4	6

### EMETTI BIGLIETTI

	VALORE	SEMPLICE	MEDIO	COMPLESSO	TOT
<b>NILF</b>	1		10		10
<b>NEIF</b>	0				
<b>NEI</b>	4	3			12
<b>NEO</b>	1		5		5
<b>NEQ</b>	1	3			3

**NILF:** I biglietti vengono creati dal sistema, li identifichiamo come ILF [1 medio]

**NEI:** Città di Partenza, Città di Destinazione, Data, Numero Biglietti [4 semplici]

**NEO:** stampa dei biglietti, ottenuti tramite elaborazione del costo [1 medio]

**NEQ:** ricerca della corsa [1 semplice]

### REGISTRAZIONE

	VALORE	SEMPLICE	MEDIO	COMPLESSO	TOT
<b>NILF</b>	1	7			7
<b>NEIF</b>	0				
<b>NEI</b>	3	3			9
<b>NEO</b>	0				
<b>NEQ</b>	1	3			3

**NILF:** L'account cliente viene creato dal sistema, lo identifichiamo come ILF [1 semplice]

**NEI:** Nome utente, Email, Password [3 semplici]

**NEQ:** ricerca l'esistenza di un cliente già registrato con lo stesso nome utente e/o indirizzo email utilizzato [1 semplice]

- Tabella elenco dei fattori correttivi (il cui valore è compreso tra 0 e 5)

<b>FATTORI CORRETTIVI</b>	
<b>COMUNICAZIONE DATI</b>	3
<b>DISTRIBUZIONE ELABORAZIONE</b>	0
<b>PRESTAZIONI</b>	1
<b>UTILIZZO INTENSIVO CONFIGURAZIONE</b>	0
<b>FREQUENZA DELLE TRANSAZIONI</b>	3
<b>INSERIMENTO DATI INTERATTIVO</b>	3
<b>EFFICIENZA PER L'UTENTE FINALE</b>	3
<b>AGGIORNAMENTO INTERATTIVO</b>	2
<b>COMPLESSITA' ELABORATIVA</b>	0
<b>RIUSABILITA'</b>	3
<b>FACILITA' INSTALLAZIONE</b>	2
<b>FACILITA' GESTIONE OPERATIVA</b>	2
<b>MOLTEPLICITA' DI SITI</b>	0

FACILITA' DI MODIFICA	3	ambiente hardware e software). L'applicazione è stata sviluppata per minimizzare gli impatti delle modifiche.
	25	

## EMETTI BIGLIETTI

UFP = 30
LOC/FP = 1590

FP = 27
JAVA = 1431

## REGISTRAZIONE

UFP = 19
LOC/FP = 1273

FP = 17,1
PHP = 1146 <sup>1</sup>

<sup>1</sup> [\(PDF\) Function Point Analysis FPA on A Team Planning Website Based on PHP and MYSQL](#) (~67 LOC/FP for PHP) – prof. Anders Lassen, Department of Computer Science – University of Copenhagen

## 4. Piano di test funzionale

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*EmettiBiglietti*”.**

Città di partenza	Città di destinazione	Data	NUMEROBIGLIETTI	ELEMENTO DI SISTEMA DATABASE
<ul style="list-style-type: none"> <li>Stringa di caratteri di lunghezza &lt;= 100</li> <li>Stringa di caratteri di lunghezza &gt; 100 [ERROR]</li> <li>Stringa di lunghezza 0 [ERROR]</li> <li>Stringa che contiene simboli che non sono caratteri [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Stringa di caratteri di lunghezza &lt;= 100</li> <li>Stringa di caratteri di lunghezza &gt; 100 [ERROR]</li> <li>Stringa di lunghezza 0 [ERROR]</li> <li>Stringa che contiene simboli che non sono caratteri [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Data con formato e valori validi (gg/mm/aaaa)</li> <li>Data con formato valido ma valori non validi [ERROR]</li> <li>Data con formato non valido [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Numero intero &gt;0</li> <li>Numero intero &lt;=0 [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>DisponibilitaBiglietti &gt;= NUMEROBIGLIETTI</li> <li>DisponibilitaBiglietti &lt; NUMEROBIGLIETTI [ERROR]</li> <li>Corsa non esistente [ERROR]</li> </ul>

Ci sono 5 categorie, di cui: 2 categorie con 4 classi di valori; 2 categorie con 3 classi di valori e 1 categoria con 2 classi di valori. Il numero di test da effettuarsi senza particolari vincoli è:  $4 \cdot 4 \cdot 3 \cdot 3 \cdot 2 = 2^5 \cdot 3^2 = 288$

Introduciamo i vincoli [ERROR].

Il numero di test da eseguire per testare singolarmente i vincoli è 11 (3 per Città di partenza, 3 per Città di destinazione, 2 per Data, 2 per ElementoDiSistemaDatabase e 1 per NUMEROBIGLIETTI).

Il numero di test risultante è:  $(1 \cdot 1 \cdot 1 \cdot 1 \cdot 1) + 11 = 12$



		ElementoDiSistemaDatabase validi	all’impiegato l’inserimento del nome della città di partenza.	NUMEROBIGLIETTI: 2}	non può essere vuoto!	nome della città di partenza
4	Città di partenza stringa con simboli	Città di partenza stringa con simboli [ERROR], Città di destinazione, Data, NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	L’impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all’impiegato l’inserimento del nome della città di partenza.	{Città di partenza: “N\$p0l1”, Città di destinazione: “Roma”, Data: “30/07/2025”, NUMEROBIGLIETTI: 2}	Nome della città di partenza non corretto!	Il Sistema chiede nuovamente di inserire il nome della città di partenza
5	Città di destinazione stringa > 100 caratteri	Città di partenza valida, Città di destinazione stringa > 100 caratteri [ERROR], Data, NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	L’impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all’impiegato l’inserimento del nome della città di destinazione.	{Città di partenza: “Napoli”, Città di destinazione: “aa”, Data: “30/07/2025”, NUMEROBIGLIETTI: 2}	Nome della città di destinazione troppo lungo!	Il Sistema chiede nuovamente di inserire il nome della città di destinazione
6	Città di destinazione stringa vuota	Città di partenza valida, Città di destinazione stringa vuota [ERROR], Data, NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	L’impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all’impiegato l’inserimento del nome della città di destinazione.	{Città di partenza: “Napoli”, Città di destinazione: “”, Data: “30/07/2025”, NUMEROBIGLIETTI: 2}	Il nome della città di destinazione non può essere vuoto!	Il Sistema chiede nuovamente di inserire il nome della città di destinazione
7	Città di destinazione stringa con	Città di partenza valida, Città di destinazione stringa con simboli [ERROR],	L’impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione	{Città di partenza: “Napoli”, Città di destinazione:	Nome della città di	Il Sistema chiede nuovamente

	simboli	Data, NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di destinazione.	{"R\$m\$", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	destinazione non corretto!	di inserire il nome della città di destinazione
8	Data formato valido ma valori non validi	Città di partenza, Città di destinazione validi, Data formato valido ma valori non validi [ERROR], NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento della data della corsa.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "34/07/2025", NUMEROBIGLIETTI: 2}	Data non corretta!	Il Sistema chiede nuovamente di inserire la data
9	Data formato non valido	Città di partenza, Città di destinazione validi, Data formato non valido [ERROR], NUMEROBIGLIETTI, ElementoDiSistemaDatabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento della data della corsa.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "30\072025", NUMEROBIGLIETTI: 2}	Formato Data non valido!	Il Sistema chiede nuovamente di inserire la data
10	NUMEROBIGLIETTI intero <=0	Città di partenza, Città di destinazione, Data validi NUMEROBIGLIETTI <= 0 [ERROR] ElementoDiSistemaDatabase valido	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del numero di biglietti da emettere.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: -2}	Numero di biglietti non valido!	Il Sistema chiede nuovamente di inserire il numero di biglietti
11	Disponibilità Biglietti < NUMEROBIGLIETTI	Città di partenza, Città di destinazione, Data, NUMEROBIGLIETTI validi ElementoDiSistemaDatabase non valido [ERROR]	Esiste un solo posto disponibile nel database.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: 1}	Non ci sono sufficienti posti disponibili	Esiste un solo posto disponibile nel database

				NUMEROBIGLIETTI: 2}	per la corsa!	
12	Nessuna corsa soddisfa i criteri di ricerca	Città di partenza, Città di destinazione, Data, NUMEROBIGLIETTI validi ElementoDiSistemaDatabase corsa non esistente [ERROR]	Il database è inizializzato correttamente ma non è disponibile una corsa per la data o l'ora selezionata tra le città di partenza e destinazione indicate.	{Città di partenza: "New York", Città di destinazione: "Parigi", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Non esistono corse disponibili per la data e/o le città indicate!	Il Sistema chiede nuovamente di inserire i parametri per una nuova corsa

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*Registrazione*”.**

NOME UTENTE	EMAIL	PASSWORD	ELEMENTO DI SISTEMA DATABASE
<ul style="list-style-type: none"> <li>Stringa di caratteri di lunghezza <math>\leq 30</math></li> <li>Stringa di caratteri di lunghezza <math>&gt; 30</math> [ERROR]</li> <li>Stringa di lunghezza 0 [ERROR]</li> <li>Stringa che contiene simboli che non sono caratteri [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Stringa di caratteri in cui è presente il simbolo @</li> <li>Stringa di caratteri in cui non è presente il simbolo @ [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Stringa di caratteri di lunghezza <math>\geq 8</math> che contiene almeno un numero o simbolo speciale</li> <li>Stringa di caratteri di lunghezza <math>\geq 8</math> che non contiene nessun numero e simbolo speciale [ERROR]</li> <li>Stringa di caratteri di lunghezza <math>&lt; 8</math> [ERROR]</li> </ul>	<ul style="list-style-type: none"> <li>Nome Utente e Email non registrati</li> <li>Nome Utente o Email già registrati [ERROR]</li> </ul>

Ci sono 4 categorie, di cui: 1 categoria con 4 classi di valori; 1 categoria con 3 classi di valori e 2 categorie con 2 classi di valori. Il numero di test da effettuarsi senza particolari vincoli è:  $4 \cdot 3 \cdot 2 \cdot 2 = 2^4 \cdot 3 = 48$

Introduciamo i vincoli [ERROR].

Il numero di test da eseguire per testare singolarmente i vincoli è 7 (3 per Nome utente, 1 per Email, 2 per Password e 1 per ElementoDiSistemaDatabase).

Il numero di test risultante è:  $(1 \cdot 1 \cdot 1 \cdot 1) + 7 = 8$

## TEST SUITE

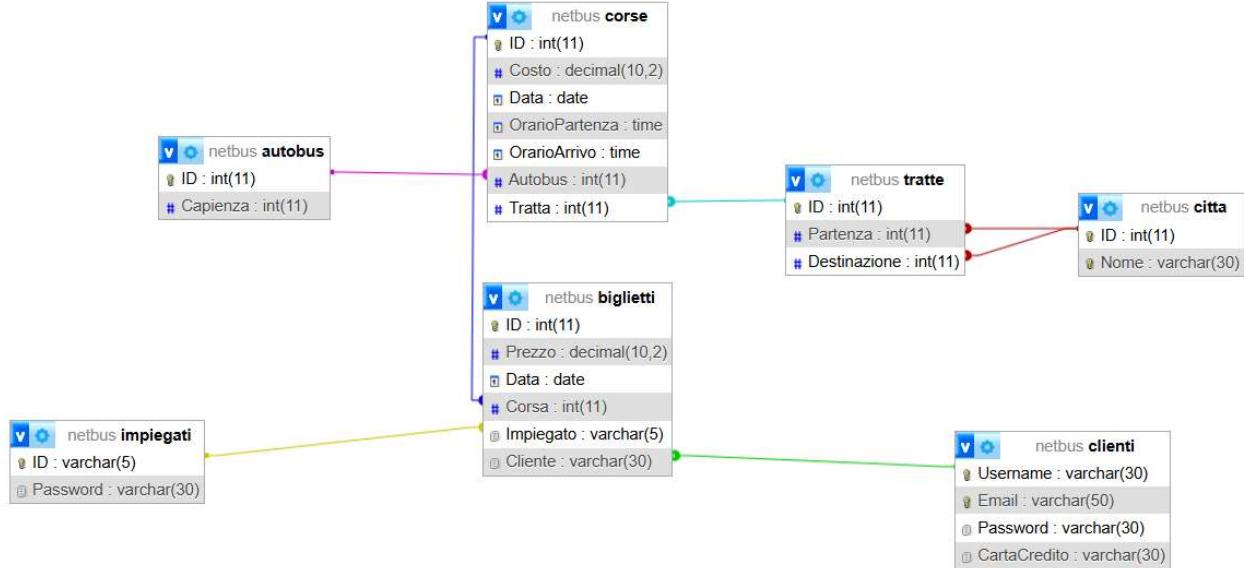
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese
1	Tutti input validi	Nome utente valido Email valida Password valida ElementoDiSistemaDatabase valido	Non esiste un cliente già registrato con lo stesso nome utente o email.	{Nome utente: "Carmine", Email: "carminesquillace@blu.it", Password: "Passw0rd"}	Registrazione completata con successo	Il cliente deve poter accedere all'account creato per poter acquistare biglietti.
2	Nome utente stringa > 30 caratteri [ERROR], Email, Password, ElementoDiSistemaDatabase validi			{Nome utente: "cccccccccccccccccccccccccc", Email: "carminesquillace@blu.it", Password: "Passw0rd"}	Nome utente troppo lungo!	Il Sistema chiede nuovamente di inserire il nome utente
3	Nome utente stringa vuota [ERROR], Email, Password, ElementoDiSistemaDatabase validi			{Nome utente: "", Email: "carminesquillace@blu.it", Password: "Passw0rd"}	Il nome utente non può essere vuoto!	Il Sistema chiede nuovamente di inserire il nome utente
4	Nome utente stringa con simboli [ERROR], Email, Password, ElementoDiSistemaDatabase validi			{Nome utente: "C_rm1n€", Email: "carminesquillace@blu.it", Password: "Passw0rd"}	Il nome utente non può contenere simboli speciali!	Il Sistema chiede nuovamente di inserire il nome utente
5	Email senza	Nome utente valido		{Nome utente: "Carmine",	L'indirizzo	Il Sistema

	simbolo @	Email senza simbolo @ [ERROR] Password, ElementoDiSistemaDatabase validi		Email: "carminesquillace.it", Password: "Passw0rd"	email deve contenere il simbolo @	chiede nuovamente di inserire l'indirizzo email
6	Password che non contiene nessun numero e simbolo speciale	Nome utente, Email validi Password debole [ERROR], ElementoDiSistemaDatabase valido		{Nome utente: "Carmine", Email: "carminesquillace@blu.it", Password: "Password"}	La password deve contenere almeno un numero o un simbolo speciale	Il Sistema chiede nuovamente di inserire la password
7	Password corta (stringa < 8 caratteri)	Nome utente, Email validi, Password corta [ERROR], ElementoDiSistemaDatabase valido		{Nome utente: "Carmine", Email: "carminesquillace@blu.it", Password: "Passw"}	La password deve essere composta da almeno 8 caratteri	Il Sistema chiede nuovamente di inserire la password
8	Nome utente o email già registrati	Nome utente, Email, Password validi ElementoDiSistemaDatabase [ERROR]	Esiste un cliente già registrato con lo stesso nome utente o email.	{Nome utente: "Alessandro", Email: "alessa.mascolo@studenti.unina.it", Password: "Passw0rd"}	Nome utente o email già registrati	Il Sistema chiede nuovamente di inserire il nome utente e l'email

## 5. Progettazione

### 5.1 Progettazione della base di dati

#### 5.1.1 Progettazione logica



Schemi di relazione:

**Biglietti**(ID, Prezzo, Data, Corsa: Corse, Impiegato: Impiegati, Cliente: Clienti)

**Impiegati**(ID, Password)

**Clienti**(Username, Email, Password, CartaCredito)

**Corse**(ID, Costo, Data, OrarioPartenza, OrarioArrivo, Autobus: Autobus, Tratta: Tratte)

**Città**(ID, Nome)

**Tratte**(ID, Partenza: Città, Destinazione: Città)

**Autobus**(ID, Capienza)

```

1 -- 
2 -- Database: `netbus` 
3 -- 
4 CREATE DATABASE `netbus`;
5 USE `netbus`;
6 
7 -- CREATE TABLE
8 CREATE TABLE BIGLIETTI(
9     ID INT AUTO_INCREMENT,
10    Prezzo DECIMAL(10, 2) NOT NULL,
11    Data DATE NOT NULL,
12    Corsa INT NOT NULL,
13    Impiegato VARCHAR(5),
14    Cliente VARCHAR(30),
15    CONSTRAINT PK_BIGLIETTI PRIMARY KEY(ID)
16 );
17 
18 CREATE TABLE IMPIEGATI(
19     ID VARCHAR(5),
20     Password VARCHAR(30) NOT NULL,
21     CONSTRAINT PK_IMPIEGATI PRIMARY KEY(ID)
22 );
23 
24 CREATE TABLE CLIENTI(
25     Username VARCHAR(30),
26     Email VARCHAR(50) NOT NULL,

```

```

27|     Password VARCHAR(30) NOT NULL,
28|     CartaCredito VARCHAR(30),
29|     CONSTRAINT PK_CLIENTI PRIMARY KEY(Username),
30|     CONSTRAINT UQ_CLIENTI UNIQUE>Email)
31| );
32|
33| CREATE TABLE CORSE(
34|     ID INT AUTO_INCREMENT,
35|     Costo DECIMAL(10, 2) NOT NULL,
36|     Data DATE NOT NULL,
37|     OrarioPartenza TIME NOT NULL,
38|     OrarioArrivo TIME NOT NULL,
39|     Autobus INT NOT NULL,
40|     Tratta INT NOT NULL,
41|     CONSTRAINT PK_CORSE PRIMARY KEY(ID)
42| );
43|
44| CREATE TABLE CITTA(
45|     ID INT AUTO_INCREMENT,
46|     Nome VARCHAR(30) NOT NULL,
47|     CONSTRAINT PK_CITTA PRIMARY KEY(ID),
48|     CONSTRAINT UK_CITTA UNIQUE(Nome)
49| );
50|
51| CREATE TABLE TRATTE(
52|     ID INT AUTO_INCREMENT,
53|     Partenza INT NOT NULL,
54|     Destinazione INT NOT NULL,
55|     CONSTRAINT PK_TRATTE PRIMARY KEY(ID)
56| );
57|
58| CREATE TABLE AUTOBUS(
59|     ID INT AUTO_INCREMENT,
60|     Capienza INT NOT NULL,
61|     CONSTRAINT PK_AUTOBUS PRIMARY KEY(ID)
62| );
63|
64| -- FOREIGN KEY
65| ALTER TABLE BIGLIETTI ADD CONSTRAINT FK_BIGLIETTI_CORSE FOREIGN KEY(Corsa) REFERENCES CORSE(
DELETE CASCADE ON UPDATE CASCADE;
66| ALTER TABLE BIGLIETTI ADD CONSTRAINT FK_BIGLIETTI_IMPIEGATI FOREIGN KEY(Impiegato) REFERENCES
IMPIEGATI(ID) ON DELETE CASCADE ON UPDATE CASCADE;
67| ALTER TABLE BIGLIETTI ADD CONSTRAINT FK_BIGLIETTI_CLIENTI FOREIGN KEY(Cliente) REFERENCES
CLIENTI(Username) ON DELETE CASCADE ON UPDATE CASCADE;
68| ALTER TABLE CORSE ADD CONSTRAINT FK_CORSE_AUTOBUS FOREIGN KEY(Autobus) REFERENCES AUTOBUS(ID)
DELETE CASCADE ON UPDATE CASCADE;
69| ALTER TABLE CORSE ADD CONSTRAINT FK_CORSE_TRATTE FOREIGN KEY(Tratta) REFERENCES TRATTE(ID) O
DELETE CASCADE ON UPDATE CASCADE;
70| ALTER TABLE TRATTE ADD CONSTRAINT FK_TRATTE_CITTA_PARTENZA FOREIGN KEY(Partenza) REFERENCES
CITTA(ID) ON DELETE CASCADE ON UPDATE CASCADE;
71| ALTER TABLE TRATTE ADD CONSTRAINT FK_TRATTE_CITTA_DESTINAZIONE FOREIGN KEY(Destinazione)
REFERENCES CITTA(ID) ON DELETE CASCADE ON UPDATE CASCADE;
72|
73| -- INSERT
74| INSERT INTO `citta` (`ID`, `Nome`) VALUES
75| (4, 'Cesena'),
76| (3, 'Milano'),
77| (1, 'Napoli'),
78| (2, 'Roma');
79|
80| INSERT INTO `autobus` (`ID`, `Capienza`) VALUES
81| (1, 70),
82| (2, 50),
83| (3, 100);
84|
85| INSERT INTO `impiegati` (`ID`, `Password`) VALUES
86| ('MI018', 'HelloWorld_'),

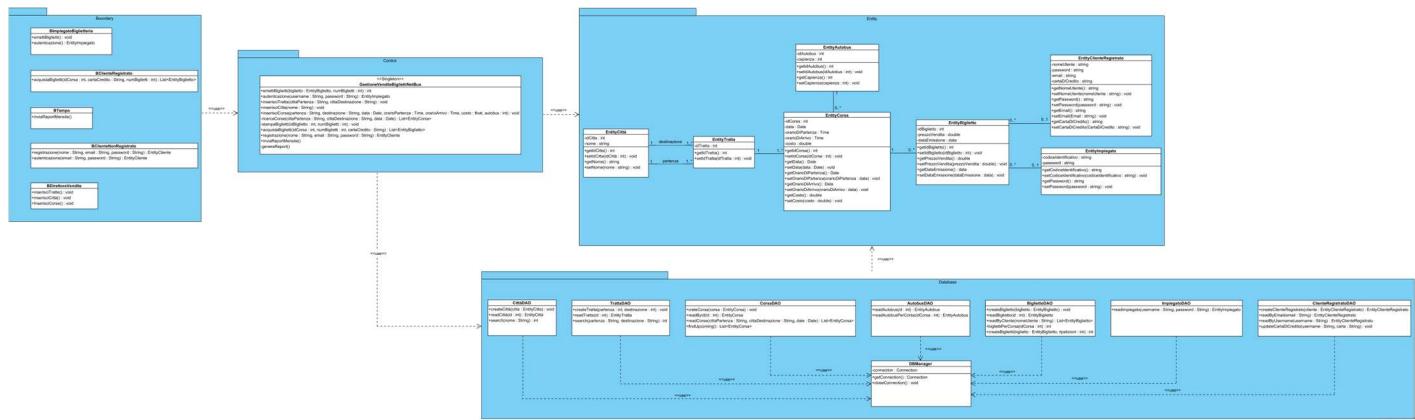
```

```

87| ('NA128', 'Passw0rd!'),
88| ('R0049', 'N3t_BUs_');
89|
90| INSERT INTO `tratte` (`ID`, `Partenza`, `Destinazione`) VALUES
91| (1, 1, 2),
92| (2, 3, 4),
93| (3, 3, 2);
94|
95| INSERT INTO `corse` (`ID`, `Costo`, `Data`, `OrarioPartenza`, `OrarioArrivo`, `Autobus`, `Tr
VALUES
96| (1, 5.00, '2025-07-30', '08:30:00', '10:00:00', 3, 1),
97| (2, 12.50, '2025-07-30', '13:00:00', '17:30:00', 1, 3),
98| (3, 8.30, '2025-07-31', '17:15:00', '19:25:00', 2, 2);

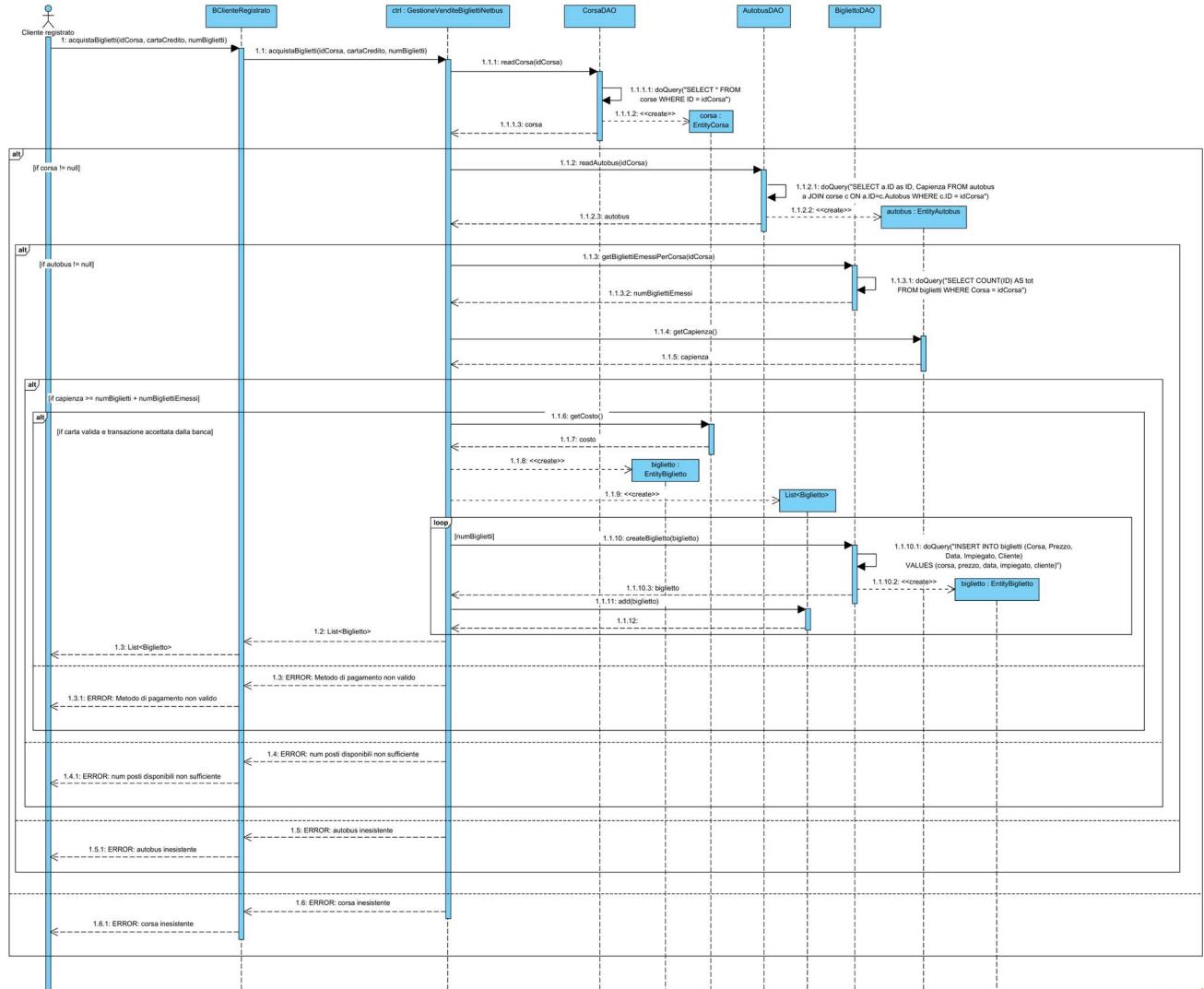
```

## 5.2 Diagramma delle classi



## 5.3 Diagrammi di sequenza

Si mostra il diagramma di sequenza di progettazione per il caso d'uso AcquistaBiglietti sviluppato fino alla codifica in PHP.



## 6. Implementazione

### 6.1.1 JavaDOC

#### Packages

Package  
it.nominasuntsubstantiarerum.netbus  
it.nominasuntsubstantiarerum.netbus.boundary  
it.nominasuntsubstantiarerum.netbus.boundary.gui  
it.nominasuntsubstantiarerum.netbus.control  
it.nominasuntsubstantiarerum.netbus.database  
it.nominasuntsubstantiarerum.netbus.entity  
it.nominasuntsubstantiarerum.netbus.exception  
it.nominasuntsubstantiarerum.netbus.util

### Package it.nominasuntsubstantiarerum.netbus.boundary

#### Classes

Class  
BDirettoreVendite  
BImpiegatoBiglietteria  
BTempo

### Package it.nominasuntsubstantiarerum.netbus.boundary.gui

#### Classes

Class  
EmissioneBiglietti  
InserisciCorse  
LoginDialog  
MenuDirettoreVendite  
MenulImpiegatoBiglietteria  
RicercaCorse

### Package it.nominasuntsubstantiarerum.netbus.control

#### Classes

Class  
GestioneVenditeBigliettiNetBus

### Package it.nominasuntsubstantiarerum.netbus.database

Classes
Class
AutobusDAO
BigliettoDAO
CittaDAO
CorsaDAO
DBManager
ImpiegatoDAO
TrattaDAO

## Package it.nominasuntsubstantiarerum.netbus.entity

Classes
Class
EntityAutobus
EntityBiglietto
EntityCitta
EntityClienteRegistrato
EntityCorsa
EntityImpiegato
EntityTratta

## Package it.nominasuntsubstantiarerum.netbus.exception

Exception Classes
Class
CredentialException
DAOException
DBConnectionException
OperationException

## Package it.nominasuntsubstantiarerum.netbus.util

Classes
Class
Time

### 6.1.2 Deploy

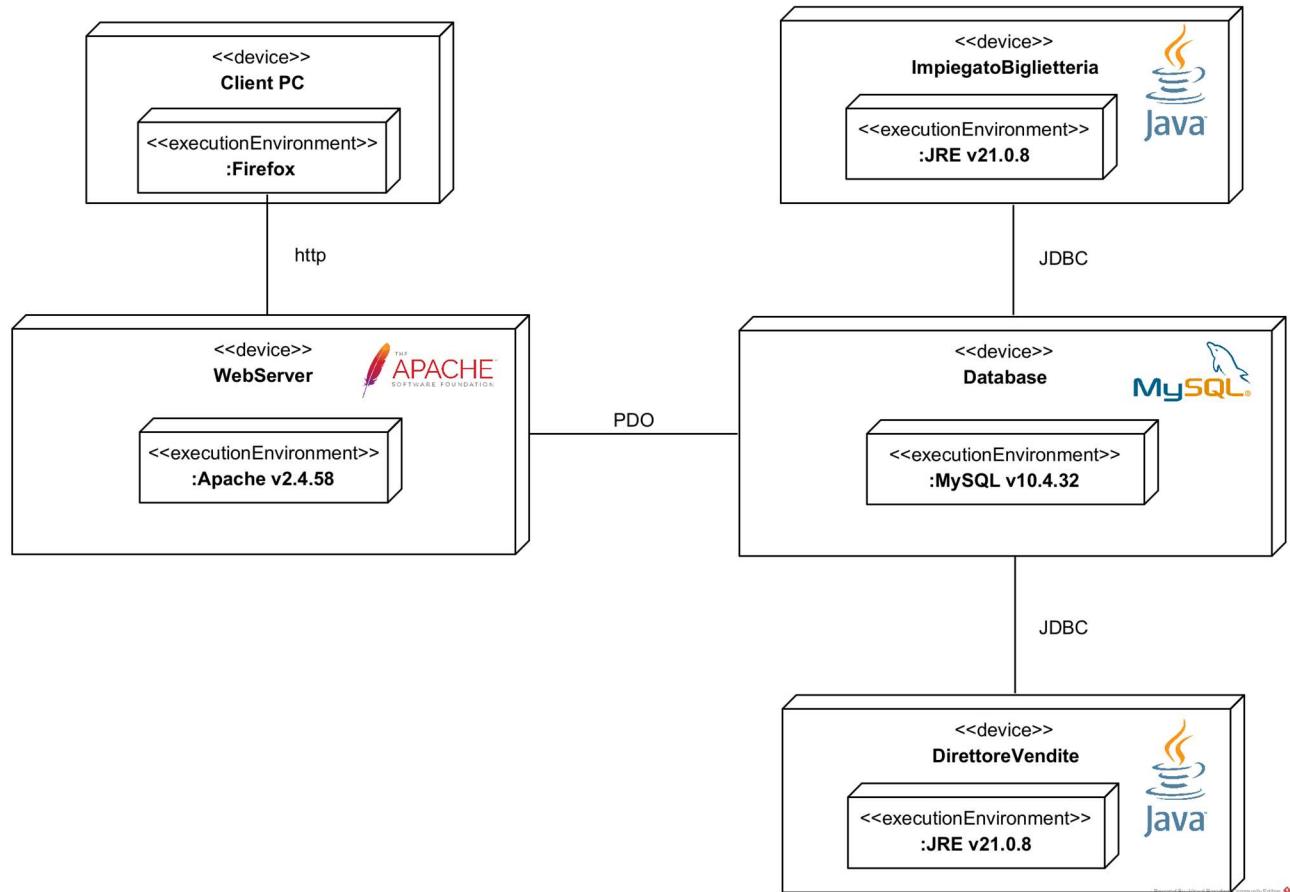
Per il deployment dell'applicazione JAVA sono richieste le seguenti dipendenze:

- *com.jgoodies.common\_1.8.1.v20240327-0800.jar*
- *com.jgoodies.forms\_1.9.0.v20240327-0800.jar*
  - utilizzate per la gestione della GUI (Graphical User Interface)
- *mysql-connector-j-8.0.32.jar*
  - utilizzata per il collegamento al database MySQL

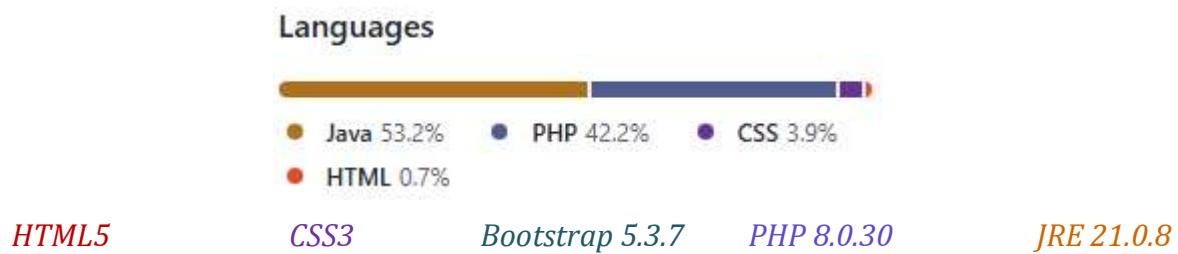
Pertanto, dovranno essere incluse nel build path del progetto.

Per il deployment del sito è stato utilizzato XAMPP (v.3.3.0), per il web server Apache (v.2.4.58) e il DBMS MySQL (MariaDB v.10.4.32), necessari all'esecuzione del sito in ambiente locale.

### Diagramma di deployment



### 6.1.3 Metriche



#### SITO

Total : 27 files, 12702 NCNB, 273 CLOC, 1278 blanks, all 14253 lines (2210 lines<sup>2</sup>)

##### Languages

language	files	code	comment	blank	total
CSS	2	11,349	51	953	12,353
PHP	24	1,337	222	325	1,884
HTML	1	16	0	0	16

##### Files

filename	language	code	comment	blank	total
c:\Users\Carmino Squillace\Desktop\sito\mis\biglietti.php	PHP	91	6	18	115
c:\Users\Carmino Squillace\Desktop\sito\mis\boundary\ClientNonRegistrato.php	PHP	21	2	6	29
c:\Users\Carmino Squillace\Desktop\sito\mis\boundary\ClienteRegistrato.php	PHP	21	1	6	28
c:\Users\Carmino Squillace\Desktop\sito\mis\buy.php	PHP	179	11	34	224
c:\Users\Carmino Squillace\Desktop\sito\mis\control\gestionevenditeBigliettiNetbus.php	PHP	120	61	49	230
c:\Users\Carmino Squillace\Desktop\sito\mis\database\AutobusDAO.php	PHP	26	6	6	38
c:\Users\Carmino Squillace\Desktop\sito\mis\database\BigliettoDAO.php	PHP	92	24	21	137
c:\Users\Carmino Squillace\Desktop\sito\mis\database\CittaDAO.php	PHP	26	6	8	40
c:\Users\Carmino Squillace\Desktop\sito\mis\database\ClientiDAO.php	PHP	83	24	23	130
c:\Users\Carmino Squillace\Desktop\sito\mis\database\CorsaDAO.php	PHP	53	11	8	72
c:\Users\Carmino Squillace\Desktop\sito\mis\database\DBManager.php	PHP	30	6	6	42
c:\Users\Carmino Squillace\Desktop\sito\mis\database\TrattaDAO.php	PHP	25	6	6	37
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\entity\entityautobus.php	PHP	22	2	6	30
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\Entitybiglietto.php	PHP	54	6	14	74
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\EntityCitta.php	PHP	22	2	6	30
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\Entitycliente.php	PHP	38	4	10	52
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\Entitycorsa.php	PHP	62	7	16	85
c:\Users\Carmino Squillace\Desktop\sito\mis\entity\EntityTratta.php	PHP	30	3	9	42
c:\Users\Carmino Squillace\Desktop\sito\mis\index.php	PHP	126	8	19	153
c:\Users\Carmino Squillace\Desktop\sito\mis\login.php	PHP	74	8	20	102
c:\Users\Carmino Squillace\Desktop\sito\mis\logout.php	PHP	5	0	1	6
c:\Users\Carmino Squillace\Desktop\sito\mis\print.php	PHP	15	4	8	27
c:\Users\Carmino Squillace\Desktop\sito\mis\register.php	PHP	90	14	20	124
c:\Users\Carmino Squillace\Desktop\sito\mis\site\footer.html	HTML	16	0	0	16
c:\Users\Carmino Squillace\Desktop\sito\mis\site\header.php	PHP	32	0	5	37
c:\Users\Carmino Squillace\Desktop\sito\mis\style\bootstrap.css	CSS	11,120	24	899	12,043
c:\Users\Carmino Squillace\Desktop\sito\mis\style\style.css	CSS	229	27	54	310
Total		12,702	273	1,278	14,253

#### APP JAVA

Total : 8 packages, 30 files, 1876 NCNB, 164 CLOC, 289 blanks, all 2329 lines

Language	files	blank	comment	code
Java	30	289	164	1876

Le metriche relative alle *Lines of Code* (LOC) sono state ottenute utilizzando *VS Code Counter* (v3.5.0), un plug-in per l'IDE Visual Studio Code dedicato all'analisi quantitativa del sorgente.

<sup>2</sup> 12043 lines sono riservate a Bootstrap

#### 6.1.4 Confronto con la stima dei costi

La stima iniziale dei costi, effettuata tramite analisi dei Function Points (FP) per la funzionalità di registrazione dei clienti sul sito web, aveva previsto un ammontare di circa 1146 LOC. A seguito dello sviluppo effettivo, il numero complessivo delle righe di codice risulta essere pari a 2210. Ciò comporta uno scostamento di +1064 righe, pari a un incremento di circa 92,8% rispetto alla stima iniziale.

Lo scostamento è attribuibile a diversi fattori. In primo luogo, la stima basata sui FP considera principalmente la logica di business legata alla registrazione, escludendo il codice necessario all'interfaccia utente (HTML, CSS, Javascript), alla gestione delle sessioni, alla validazione dei dati lato client e agli aspetti di sicurezza. Tali elementi, pur non essendo direttamente correlati alla logica di business, sono indispensabili per il corretto funzionamento e la fruibilità dell'applicazione.

In secondo luogo, nello sviluppo effettivo sono state implementate ulteriori funzionalità non incluse nella stima iniziale, tra cui l'autenticazione del cliente e la gestione dell'acquisto dei biglietti. Queste componenti, sebbene separate dalla registrazione, condividono parti comuni del codice (come la gestione delle sessioni, il layout delle pagine etc..) e hanno contribuito all'aumento complessivo delle righe.

Per ragioni sostanzialmente analoghe, la stima dei costi relativa alla funzionalità di emissione biglietti per l'applicazione JAVA aveva previsto un totale di 1431 righe di codice. Il numero effettivo di LOC risulta invece pari a 1884, con uno scostamento di +453 LOC, corrispondente a un incremento di circa 31,6% rispetto alla stima iniziale.

In conclusione, gli scostamenti osservati risultano giustificati dal fatto che le stime iniziali si concentrano su singole funzionalità, senza tenere conto dell'intero contesto applicativo e delle componenti ausiliarie, nonché del codice necessario alla gestione dell'interfaccia grafica.

## 6.1.5 Versioning

Il versionamento del progetto è stato gestito tramite [GitHub](#), piattaforma di *Configuration Management* basata su **Git**.

Total : 69 commits, 1 branch

**About**

No description, website, or topics provided.

- Readme
- Activity
- 0 stars
- 0 watching
- 0 forks

**Releases**

No releases published

[Create a new release](#)

**Packages**

No packages published

[Publish your first package](#)

**Contributors**

- CarmineSquillace
- Manoplay Alessandro Mascolo

**Languages**

Java 53.2% PHP 42.2% CSS 3.9% HTML 0.7%

**Suggested workflows**

Based on your tech stack

- Symfony Configure Test a Symfony project.
- Clojure Configure Build and test a Clojure project with Leiningen.
- Laravel Configure Test a Laravel project.

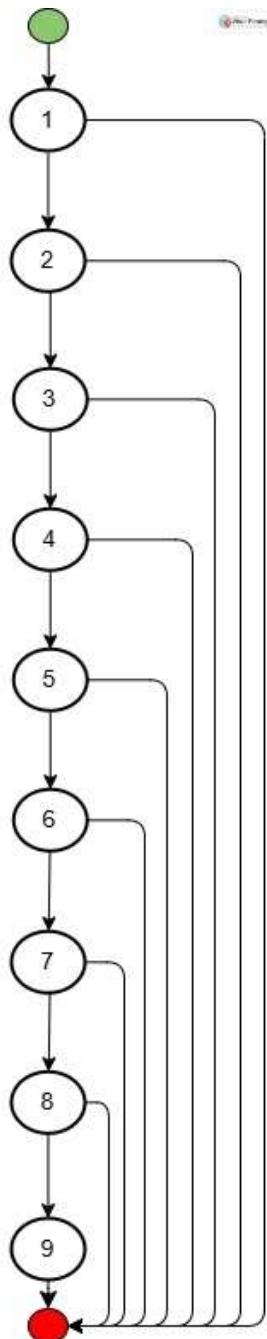
[More workflows](#) [Dismiss suggestions](#)

© 2025 GitHub, Inc. [Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)

## 7. Testing

### 7.1 Test strutturale

#### 7.1.1 Complessità ciclomatica



```

1. public static function registrazione(String $nome, String $email, String $password,
String $passwordConfirm): EntityCliente {
2.     if ($password !== $passwordConfirm)
3.         throw new Exception('Le password non coincidono.');
4.
5.     if (strlen($nome) > 30)
6.         throw new Exception('Nome utente troppo lungo.');
7.
8.     if ($nome === "")
9.         throw new Exception('Il nome utente non può essere vuoto.');
10.
11.    if (preg_match('/[^a-zA-Z0-9]/', $nome))
12.        throw new Exception('Il nome utente non può contenere simboli speciali.');
13.
14.    if (ClienteDAO::readByEmail($email))
15.        throw new Exception('Email già registrata.');
16.
17.    if (ClienteDAO::readByUsername($nome))
18.        throw new Exception('Nome già registrato.');
19.
20.    if (strlen($password) < 8)
21.        throw new Exception('La password deve avere una lunghezza minima di 8
caratteri.');
22.
23.    if (!preg_match('/\d/', $password) and !preg_match('/[^a-zA-Z0-9]/', $password))
24.        throw new Exception('La password deve contenere almeno un numero o un simbolo
speciale.');
25.
26.    $cliente = new EntityCliente($nome, $email, $password, null);
27.    return ClienteDAO::createCliente($cliente);
28. }
  
```

8 if in serie  $\rightarrow 2^8 = 256$  path.

#### NUMERO CICLOMATICICO:

numero di regioni chiuse del grafo = 9

numero di nodi predicati (1, 2, 3, 4, 5, 6, 7, 8) + 1 = 9

# archi - # nodi + 2 = (16 - 9) + 2 = 9

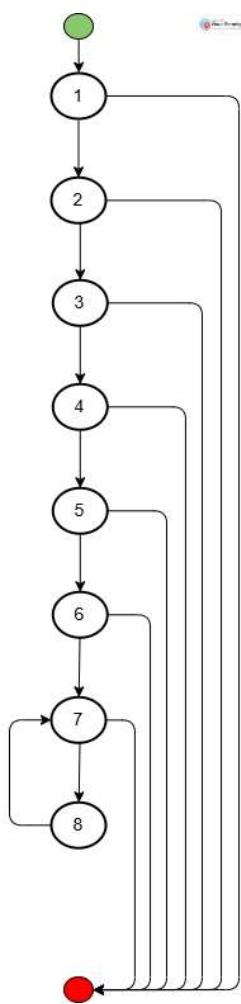
#### CAMMINI:

##### Cammino Base (senza deviazioni):

- 1) 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9

##### Cammini con deviazioni (uno per ogni biforcazione):

- 2) 1
- 3) 1 → 2
- 4) 1 → 2 → 3
- 5) 1 → 2 → 3 → 4
- 6) 1 → 2 → 3 → 4 → 5
- 7) 1 → 2 → 3 → 4 → 5 → 6
- 8) 1 → 2 → 3 → 4 → 5 → 6 → 7
- 9) 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8



```

1.      /**
2.       * Acquista uno o più biglietti per la corsa indicata
3.       * @return EntityBiglietto[]
4.       * @throws Exception
5.       */
6.   public static function acquistaBiglietti(int $idCorsa, String $cartaCredito, int
$numBiglietti): array
7.   {
8.     $cliente = self::getLoggedCliente();
9.
10.    // validazione cliente
11.    if (!$cliente)
12.      throw new Exception('Devi essere autenticato per acquistare.');
13.
14.    // validazione corsa
15.    $corsa = CorsaDAO::readById($idCorsa);
16.    if (!$corsa)
17.      throw new Exception('Corsa non trovata.');
18.
19.    // validazione numero biglietti
20.    if ($numBiglietti<=0)
21.      throw new Exception('Devi acquistare almeno un biglietto.');
22.
23.    $autobusAssociato = AutobusDAO::readAutobusPerCorsa($idCorsa);
24.
25.    if ($autobusAssociato === null)
26.      throw new Exception("Autobus associato alla corsa non trovato.");
27.
28.    $numBigliettiTotaliEmessi = BigliettoDAO::bigliettiPerCorsa($idCorsa);
29.
30.    if ($numBigliettiTotaliEmessi + $numBiglietti > $autobusAssociato->getCapienza())
31.      throw new Exception("Impossibile emettere " . $numBiglietti . " biglietti:
supererebbe la capienza dell'autobus (" . $autobusAssociato->getCapienza() . ").");
32.
33.    // validazione metodo di pagamento
34.    if (strlen($cartaCredito)!=16 || preg_match('/[^0-9]/', $cartaCredito))
35.      throw new Exception('Metodo di pagamento non valido.');
36.
37.    // Determina il totale
38.    $tot = $corsa->getCosto()*$numBiglietti;
39.
40.    // ....effettua la transazione richiamando una API dell'istituto di credito....
41.
42.    // emetti i biglietti
43.    $biglietto = new EntityBiglietto(-1, $corsa->getIdCorsa(), $corsa->getCosto(), new
DateTime(), $cliente->getNomeUtente(), null);
44.
45.    $list = [];
46.    for($i=0; $i<$numBiglietti; $i++)
47.      $list[] = BigliettoDAO::createBiglietto($biglietto);
48.
49.    return $list;
50.  }
  
```

### NUMERO CICLOMATICICO:

numero di regioni chiuse del grafo = 8

numero di nodi predicati (1, 2, 3, 4, 5, 6, 7) + 1 = 8

# archi - # nodi + 2 = (14 - 8) + 2 = 8

**CAMMINI:****Cammino Base (senza deviazioni):**

1) 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 7

**Cammini con deviazioni (uno per ogni biforcazione):**

- 2) 1
- 3) 1 → 2
- 4) 1 → 2 → 3
- 5) 1 → 2 → 3 → 4
- 6) 1 → 2 → 3 → 4 → 5
- 7) 1 → 2 → 3 → 4 → 5 → 6
- 8) 1 → 2 → 3 → 4 → 5 → 6 → 7

**7.1.2 Test di unità**

```

 $T_1 = \{$ 
  {"Carmine", "carminesquillace@blu.it", "Passw0rd", "Passw0rd"},  

  {"Carmine", "carminesquillace@blu.it", "Passw0rd", "password"},  

  {"Carmineeeeeeeeeeeeeeeeeeeeeeee", "carminesquillace@blu.it", "Passw0rd", "Passw0rd"},  

  {"", "carminesquillace@blu.it", "Passw0rd", "Passw0rd"},  

  {"çàrm!n€", "carminesquillace@blu.it", "Passw0rd", "Passw0rd"},  

  {"Alessandro", "alessa.mascolo@studenti.unina.it", "Passw0rd", "Passw0rd"},  

  {"Alessandro", "alessa.mascolo@studenti.unina.it", "Passw0rd", "Passw0rd"},  

  {"Carmine", "carminesquillace@blu.it", "pasw", "pasw"},  

  {"Carmine", "carminesquillace@blu.it", "Password", "Password"}  

 $\}$ 

```

```

 $T_2 = \{$ 
  {1, "1234567891234567", 2, },  

  {1, "1234567891234567", 2, },  

  {-1, "1234567891234567", 2, }  

  {1, "1234567891234567", -1, },  

  {3, "1234567891234567", 1, },  

  {1, "1234567891234567", 200, },  

  {1, "123A5678B", 1, },  

  {1, "1234567891234567", 0, }  

 $\}$ 

```

## 7.2 Test funzionale

Durante l'esecuzione dei test funzionali pianificati per la funzionalità di emissione biglietti dell'applicazione JAVA, i test case con ID da 2 a 7 sono risultati falliti. Il test ha evidenziato che l'applicazione **non eseguiva alcun controllo sulla validità degli input** forniti proseguendo comunque nella ricerca di una corsa anche in presenza di dati non corretti o incompleti.

Questa mancanza di validazione portava a comportamenti non desiderati, come l'avvio di ricerche per corse inesistenti o impossibili, senza fornire un feedback adeguato all'impiegato della biglietteria.

Sono stati pertanto aggiunti opportuni controlli di validità sugli input prima di eseguire la ricerca di una corsa, in modo da informare correttamente l'impiegato in caso di input non validi.

Infine, il test case con ID 11 ha evidenziato un grave problema funzionale: l'applicazione **non verificava il numero di posti liberi disponibili** sull'autobus assegnato alla corsa selezionata, consentendo l'emissione di biglietti anche in caso questi eccedessero il numero di posti liberi disponibili. Pertanto, è stato introdotto un controllo preventivo durante il processo di emissione, volto a verificare che il numero di biglietti richiesto non superi l'effettività disponibilità di posti liberi. In caso contrario, l'applicazione informa l'impiegato dell'impossibilità di procedere con l'emissione, evitando così situazioni incoerenti.

Difetti simili sono stati rilevati anche durante l'esecuzione dei test funzionali relativi alla funzionalità di registrazione clienti sul sito web della compagnia. In particolare:

- Il sistema consentiva la registrazione con nomi utente non validi (vedi test case con ID 2 e 4);
- Il sistema consentiva la registrazione con password troppo deboli, prive di numeri o caratteri speciali (vedi test case con ID 6).

Per la risoluzione sono stati introdotti controlli di validità tramite **espressioni regolari (regex)**, al fine di garantire l'inserimento di credenziali conformi ai requisiti di sicurezza e formato previsti.

➤ *Poiché i difetti riscontrati, sono riconducibili a fault di tipo "missing logic" - ovvero relativi alla mancanza di controlli fondamentali nel flusso applicativo - risulterebbero difficilmente individuabili mediante test strutturali (white-box).*

<b>Test Case ID</b>	<b>Descrizione</b>	<b>Classi di equivalenza coperte</b>	<b>Pre-condizioni</b>	<b>Input</b>	<b>Output Attesi</b>	<b>Post-condizioni Attese</b>	<b>Output Ottenuti</b>	<b>Post-condizioni Ottenute</b>	<b>Esito (FAIL, PASS)</b>
1	Tutti input validi	Città di partenza valida Città di destinazione valida Data valida NUMEROBIGLIETTI valido ElementoDiSistemaDa tabase valido	Il database è inizializzato correttamente. È disponibile una corsa per la data e l'ora selezionate tra le città di partenza e destinazione indicate. C'è disponibilità per il numero di biglietti selezionato. L'impiegato è stato autenticato dal sistema.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	I biglietti sono stampati.	Il sistema registra internamente l'emissione, compreso l'identificativo dell'impiegato che ha emesso i biglietti.	I biglietti sono stampati.	Il sistema registra internamente l'emissione, compreso l'identificativo dell'impiegato che ha emesso i biglietti.	PASS
2	Città di partenza stringa > 100 caratteri [ERROR], Città di destinazione, Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	Città di partenza	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di partenza.	{Città di partenza: "nn", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Nome della città di partenza troppo lungo!	Il Sistema chiede nuovamente di inserire il nome della città di partenza	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	FAIL

3	Città di partenza stringa vuota [ERROR], Città di destinazione, Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di partenza.	{Città di partenza: "", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Il nome della città di partenza non può essere vuoto!	Il Sistema chiede nuovamente di inserire il nome della città di partenza	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	<span style="color: red; text-decoration: underline;">FAIL</span>
4	Città di partenza stringa con simboli [ERROR], Città di destinazione, Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di partenza.	{Città di partenza: "N\$p0l1", Città di destinazione: "Roma", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Nome della città di partenza non corretto!	Il Sistema chiede nuovamente di inserire il nome della città di partenza	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	<span style="color: red; text-decoration: underline;">FAIL</span>
5	Città di destinazione stringa > 100 caratteri [ERROR], Città di destinazione stringa > 100 caratteri [ERROR], Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di destinazione.	{Città di partenza: "Napoli", Città di destinazione: "aa", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Nome della città di destinazione troppo lungo!	Il Sistema chiede nuovamente di inserire il nome della città di destinazione	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	<span style="color: red; text-decoration: underline;">FAIL</span>

6	Città di destinazione stringa vuota	Città di partenza valida, Città di destinazione stringa vuota [ERROR], Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di destinazione.	{Città di partenza: "Napoli", Città di destinazione: "", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Il nome della città di destinazione non può essere vuoto!	Il Sistema chiede nuovamente di inserire il nome della città di destinazione	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	<b>FAIL</b>
7	Città di destinazione stringa con simboli	Città di partenza valida, Città di destinazione stringa con simboli [ERROR], Data, NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del nome della città di destinazione.	{Città di partenza: "Napoli", Città di destinazione: "R\$m\$", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Nome della città di destinazione non corretto!	Il Sistema chiede nuovamente di inserire il nome della città di destinazione	Il sistema non mostra alcun messaggio di errore	Non viene trovata alcuna corsa che soddisfa i parametri di ricerca.	<b>FAIL</b>
8	Data formato valido ma valori non validi	Città di partenza, Città di destinazione validi, Data formato valido ma valori non validi [ERROR], NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento della data della corsa.	{Città di partenza: "Napoli", Città di destinazione: "Roma", Data: "34/07/2025", NUMEROBIGLIETTI: 2}	Data non corretta!	Il Sistema chiede nuovamente di inserire la data	Inserisci una data valida (dd/mm/yy yy).		<b>PASS</b>
9	Data	Città di partenza, Città	L'impiegato, dopo	{Città di partenza: Formato	Il Sistema	Inserisci			

	formato non valido	di destinazione validi, Data formato non valido [ERROR], NUMEROBIGLIETTI, ElementoDiSistemaDa tabase validi	essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento della data della corsa.	“Napoli”, Città di destinazione: “Roma”, Data: “30\072025”, NUMEROBIGLIETTI: 2}	Data non valido!	chiede nuovamente di inserire la data	una data valida (dd/mm/yy yy).		PASS
10	NUMEROBIGLIETTI intero <=0	Città di partenza, Città di destinazione, Data validi NUMEROBIGLIETTI <= 0 [ERROR] ElementoDiSistemaDa tabase valido	L'impiegato, dopo essersi autenticato, ha selezionato la funzionalità di emissione biglietti. Il sistema chiede all'impiegato l'inserimento del numero di biglietti da emettere.	{Città di partenza: “Napoli”, Città di destinazione: “Roma”, Data: “30/07/2025”, NUMEROBIGLIETTI: -2}	Numero di biglietti non valido!	Il Sistema chiede nuovamente di inserire il numero di biglietti	Il numero di biglietti viene automaticamente reimpostato a 1.		PASS
11	DisponibilitaBiglietti < NUMEROBIGLIETTI	Città di partenza, Città di destinazione, Data, NUMEROBIGLIETTI validi ElementoDiSistemaDa tabase non valido [ERROR]	Esiste un solo posto disponibile nel database.	{Città di partenza: “Napoli”, Città di destinazione: “Roma”, Data: “30/07/2025”, NUMEROBIGLIETTI: 2}	Non ci sono sufficienti posti disponibili per la corsa!	Esiste un solo posto disponibile nel database	I biglietti sono stampati.	Il sistema registra internamente l'emissione, compreso l'identificativo dell'impiegato che ha emesso i biglietti.	FAIL

12	Nessuna corsa soddisfa i criteri di ricerca	Città di partenza, Città di destinazione, Data, NUMEROBIGLIETTI validi ElementoDiSistemaDa tabase corsa non esistente [ERROR]	Il database è inizializzato correttamente ma non è disponibile una corsa per la data o l'ora selezionata tra le città di partenza e destinazione indicate.	{Città di partenza: "New York", Città di destinazione: "Parigi", Data: "30/07/2025", NUMEROBIGLIETTI: 2}	Non esistono corse disponibili per la data e/o le città indicate!	Il Sistema chiede nuovamente di inserire i parametri per una nuova corsa	Nessuna corsa trovata.		<b>PASS</b>

Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output Attesi	Post-condizioni Attese	Output Ottenuati	Post-condizioni Ottenuute	Esito (FAIL, PASS)
1	Tutti input validi	Nome utente valido Email valida Password valida ElementoDiSistema Database valido	Non esiste un cliente già registrato con lo stesso nome utente o email.	{Nome utente: "Carmine", Email: " <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ", Password: "Passw0rd"}	Registrazione completata con successo	Il cliente deve poter accedere all'account creato per poter acquistare biglietti.	Registrazione avvenuta con successo !	L'account del cliente viene correttamente memorizzato nel database in modo da consentire l'accesso all'account .	<span style="color: green;">PASS</span>
2	Nome utente stringa > 30 caratteri [ERROR], Email, Password, ElementoDiSistema Database validi			{Nome utente: "cccccccccccccccccccc", Email: " <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ", Password: "Passw0rd"}	Nome utente troppo lungo!	Il Sistema chiede nuovamente di inserire il nome utente	Registrazione avvenuta con successo !	L'account del cliente viene memorizzato nel database troncando tutti i successivi 30 caratteri del nome utente.	<span style="color: red;">FAIL</span>
3	Nome	Nome utente stringa		{Nome utente: "", Il nome utente}	Il Sistema	Il form			<span style="color: green;">PASS</span>

	utente stringa vuota	vuota [ERROR], Email, Password, ElementoDiSistema Database validi		Email: “ <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ”, Password: “Passw0rd”}	non può essere vuoto!	chiede nuovamente di inserire il nome utente	richiede la compilazione del campo in quanto mandato.		
4	Nome utente stringa con simboli	Nome utente stringa con simboli [ERROR], Email, Password, ElementoDiSistema Database validi		{Nome utente: “C_rm1n€”, Email: “ <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ”, Password: “Passw0rd”}	Il nome utente non può contenere simboli speciali!	Il Sistema chiede nuovamente di inserire il nome utente	Registrazione avvenuta con successo !	L'account del cliente viene memorizzato nel database.	<span style="color: red;">FAIL</span>
5	Email senza simbolo @	Nome utente valido Email senza simbolo @ [ERRRR] Password, ElementoDiSistema Database validi		{Nome utente: “Carmine”, Email: “ <a href="mailto:carminesquillace.it">carminesquillace.it</a> ”, Password: “Passw0rd”}	L'indirizzo email deve contenere il simbolo @	Il Sistema chiede nuovamente di inserire l'indirizzo email	Il form richiede un formato corretto per l'indirizzo email.		<span style="color: green;">PASS</span>
6	Password che non contiene nessun numero e simbolo speciale	Nome utente, Email validi Password debole [ERROR], ElementoDiSistema Database valido		{Nome utente: “Carmine”, Email: “ <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ”, Password: “Password”}	La password deve contenere almeno un numero o un simbolo speciale	Il Sistema chiede nuovamente di inserire la password	Registrazione avvenuta con successo !	L'account del cliente viene memorizzato nel database.	<span style="color: red;">FAIL</span>

7	Password corta (stringa < 8 caratteri)	Nome utente, Email validi, Password corta [ERROR], ElementoDiSistema Database valido		{Nome utente: “Carmine”, Email: “ <a href="mailto:carminesquillace@blu.it">carminesquillace@blu.it</a> ”, Password: “Passw”}	La password deve essere composta da almeno 8 caratteri	Il Sistema chiede nuovamente di inserire la password	Il form richiede una stringa composta dal almeno 8 caratteri.		<b>PASS</b>
8	Nome utente o email già registrati	Nome utente, Email, Password validi ElementoDiSistema Database [ERROR]	Esiste un cliente già registrato con lo stesso nome utente o email.	{Nome utente: “Alessandro”, Email: “ <a href="mailto:alessa.mascolo@studenti.unina.it">alessa.mascolo@studenti.unina.it</a> ”, Password: “Passw0rd”}	Nome utente o email già registrati	Il Sistema chiede nuovamente di inserire il nome utente e l'email	Nome utente o email già registrati		<b>PASS</b>