



# UNIVERSITÀ DEGLI STUDI FIRENZE

**SCUOLA DI INGEGNERIA-DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE**

.....  
**TRAVELBUS**

**BUS BOOKING SYSTEM**  
.....

Autore:

Boris nono (7127902)

Alain Maetial Mbogning

Professore:

Enrico vicario

Indice

1. Introduzione.....

1.1 statement.....
1.2 architettura e pratiche utilizzate.....
2. Analisi e specifica dei requisiti .....
2.1 Processo Utente.....
2.1.1 Use case #1: Registrazione..... .....
2.1.2 Use case #2: Autenticazione.....
2.1.3 Use case #3 :Ricerca e Selezione della corsa.....
2.1.4 Use case # 4 : Prenotazione della corsa.....
2.1.5 Use case #5 :Modifica o Cancellazione della Prenotazione.....
Use Case Template :Utente.....
Use Case Diagram : Utente.....
2.1.6. Diagramma Dello Stato di Pagamento.....
2.2.Processo Amministratore.....
2.2.1 Use case #6: Aggiunta di una corsa.....
2.2.2 Use case #7:Cancellazione di una corsa.....
2.2.3 Use case #8 :Monitoraggio delle statistiche.....
Use Case Template : Admin.....
Use case Diagram : admin
2.2.4. Diagramma delle classi.....
2.3. Diagramma delle attività.....
2.4. Testing.....
3. Mockup.....

## 1. Introduzione

Si propone la realizzazione di un sistema software che consenta agli utenti di **prenotare biglietti per autobus** in modo semplice ed efficiente.

## 1.1 statement

Il sistema deve offrire le seguenti funzionalità:

- **Registrazione degli utenti** generici, che devono fornire i propri **dati personali** (nome, cognome e indirizzo email).
- **Autenticazione** tramite email e password per accedere alla piattaforma.
- **Ricerca delle corse disponibili**, filtrando per:
  - **Data** del viaggio.
  - **Orario di partenza**.
  - **Numero di posti disponibili**.
  - **Prezzo del biglietto**.
- **Prenotazione del viaggio**, con notifica di conferma contenente i dettagli della corsa scelta.
- **Gestione prenotazioni**, permettendo agli utenti di **modificare o annullare** una prenotazione esistente.
- **Gestione amministrativa**, con la possibilità per gli amministratori di:
  - **Aggiungere** nuove corse.
  - **Modificare** le corse esistenti.
  - **Cancellare** corse non più disponibili.
  - **Monitorare le statistiche di occupazione** per ottimizzare il servizio.

## 1.2. Architettura e Pratiche utilizzate

Il progetto è stato sviluppato in java . Per mantenere una separazione delle responsabilità , la struttura è stata divisa in tre parti principali:

- **Model:** Gestisce i dati e la logica del software, fornendo metodi per l'accesso e la manipolazione delle informazioni.
- **View:** Si occupa della presentazione dei dati contenuti nel Model e gestisce l'interazione con gli utenti e gli agenti esterni.
- **Controller:** Riceve i comandi dell'utente (solitamente attraverso la View) e li elabora, modificando lo stato del Model e aggiornando la View di conseguenza.

Questa separazione consente una progettazione **modulare, scalabile e manutenibile**, facilitando l'evoluzione del sistema e l'integrazione di nuove funzionalità

## 2. Analisi e Specifica dei Requisiti

L'analisi e la specifica dei requisiti definiscono le caratteristiche essenziali che l'applicazione deve rispettare per garantire il corretto funzionamento del sistema. I requisiti rappresentano le **proprietà, i comportamenti e le funzionalità** che devono essere implementate al termine dello sviluppo.

Per effettuare un'analisi dettagliata, vengono individuati i seguenti elementi fondamentali:

- **Casi d'uso**, che descrivono le interazioni tra gli utenti e il sistema.
- **Attori**, ovvero i ruoli degli utenti che interagiscono con l'applicazione. Possono includere:
  - **Utenti generici**, che effettuano prenotazioni.
  - **Amministratori**, responsabili della gestione delle corse e del monitoraggio delle statistiche.
- **Relazioni tra attori e casi d'uso**, che stabiliscono le connessioni tra le operazioni eseguite dagli utenti e le funzionalità offerte dal sistema.

Un **attore** può svolgere **uno o più casi d'uso**, mentre un **caso d'uso** può coinvolgere **uno o più attori**. Questa relazione consente di costruire un modello chiaro dell'interazione tra l'utente e l'applicazione, facilitando la progettazione e lo sviluppo del sistema.

### 2.1. Processo Utente

#### 2.1.1. Use Case #1: *Registrazione*

Un nuovo utente può creare un account inserendo:

- **Nome**
- **Cognome**
- **Indirizzo email**
- **Password sicura**

### **2.1.2 Use case #2 :Autenticazione**

L'utente deve inserire:

- **Email**
- **Password**

### **2.1.3 Use case #3: Ricerca e Selezione della Corsa**

L'utente può cercare una corsa utilizzando filtri come:

- **Data e ora di partenza**
- **Destinazione**
- **Numero di posti disponibili**
- **Prezzo del biglietto**

Dopo aver trovato la corsa desiderata, l'utente visualizza i dettagli della corsa:

- **Luogo di partenza**
- **Luogo di arrivo**
- **Data e ora di partenza**
- **Data e ora di arrivo**
- **Durata del viaggio**
- **Numero dell'autobus**
- **Prezzo del biglietto**

### **2.1.4 Use case #4 : Prenotazione della Corsa**

L'utente conferma la prenotazione e procede con il pagamento. Una volta completato, riceve un'**email di conferma** contenente:

- **Dettagli del viaggio**
- **Codice di prenotazione**
- **Ricevuta del pagamento**

### **2.1.5 Use case #5 : Modifica o Cancellazione della Prenotazione**

L'utente può autenticarsi in qualsiasi momento per:

- **Modificare la propria prenotazione** (se ancora possibile).

- **Annnullare la corsa**, con eventuale rimborso se previsto dalle politiche di cancellazione.

Dopo ogni modifica, il sistema aggiorna i dati e invia una nuova email di conferma con le informazioni aggiornate.

### USE Case Template : Utente

#### Use case #1 : Registrazione

Breve Descrizione : L'utente si registra al sistema con i suoi dati personali (nome ,cognome e indirizzo mail) e una password

Attori : Utenti generi(Passaggeri) , amministratori

Precondizione : l'utente deve essere sull'apposita pagina di registrazione (Mockup #1)

Flusso : 1) L'utente sceglie la pagina di registrazione

- 2) L'utente inserisce i suoi dati personali ( nome ,cognome , indirizzo mail)
- 3) L'utente sceglie una password (da ricordare) e la inserisce

Flusso alternativo : se l'utente è già presente nel sistema , esso restituisce un messaggio d'errore e ritorna nella Home Page ( Test #1)

Post-Condizione : l'utente si è registrato con successo

## **Use case #2 : Autenticazione (Login)**

Breve Descrizione : L'utente accede al sistema con le sue credenziali( Mockup #2)

Attori: Utenti generici , amministratori

Precondizione: L'utente deve essere registrato al sistema e sull'apposita pagina di login

- Flusso :
- 1) L'utente sceglie la pagina di login
  - 2) L'utente inserisce le sue credenziali ( Indirizzo mail e password)
  - 3) L'utente clicca su Accedi
  - 4) Il sistema verifica l la correttezza delle sue credenziali
  - 5) Il sistema autorizza il collegamento dell'utente

Flusso Alternativo : Le credenziali non sono corrette il sistema restituisce un messaggio di errore e ritorna alla pagina di login

Post-condizione : L'utente è logato

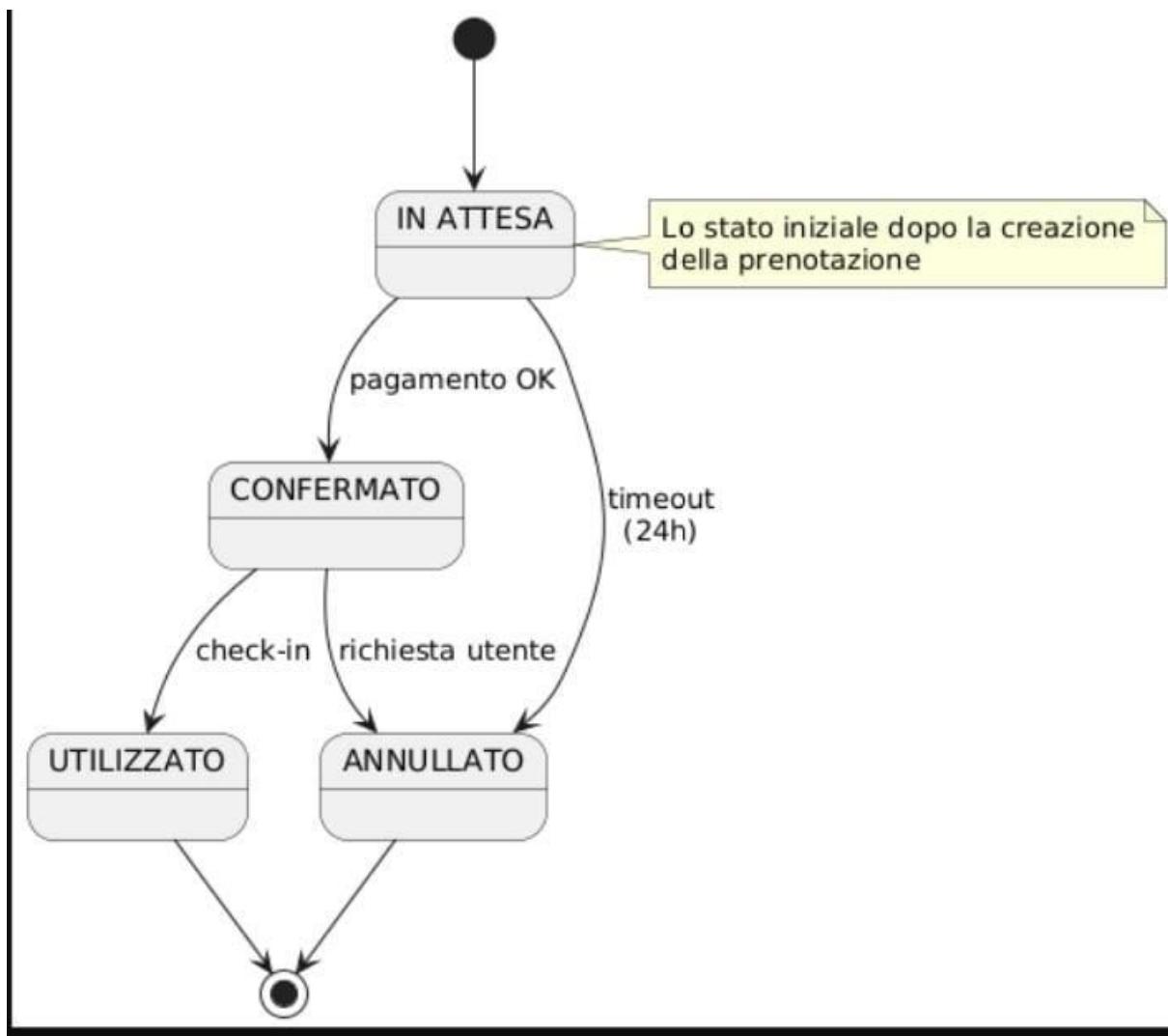
## **Use case #4 : Prenotazione della corsa**

Attori : Utenti Generici

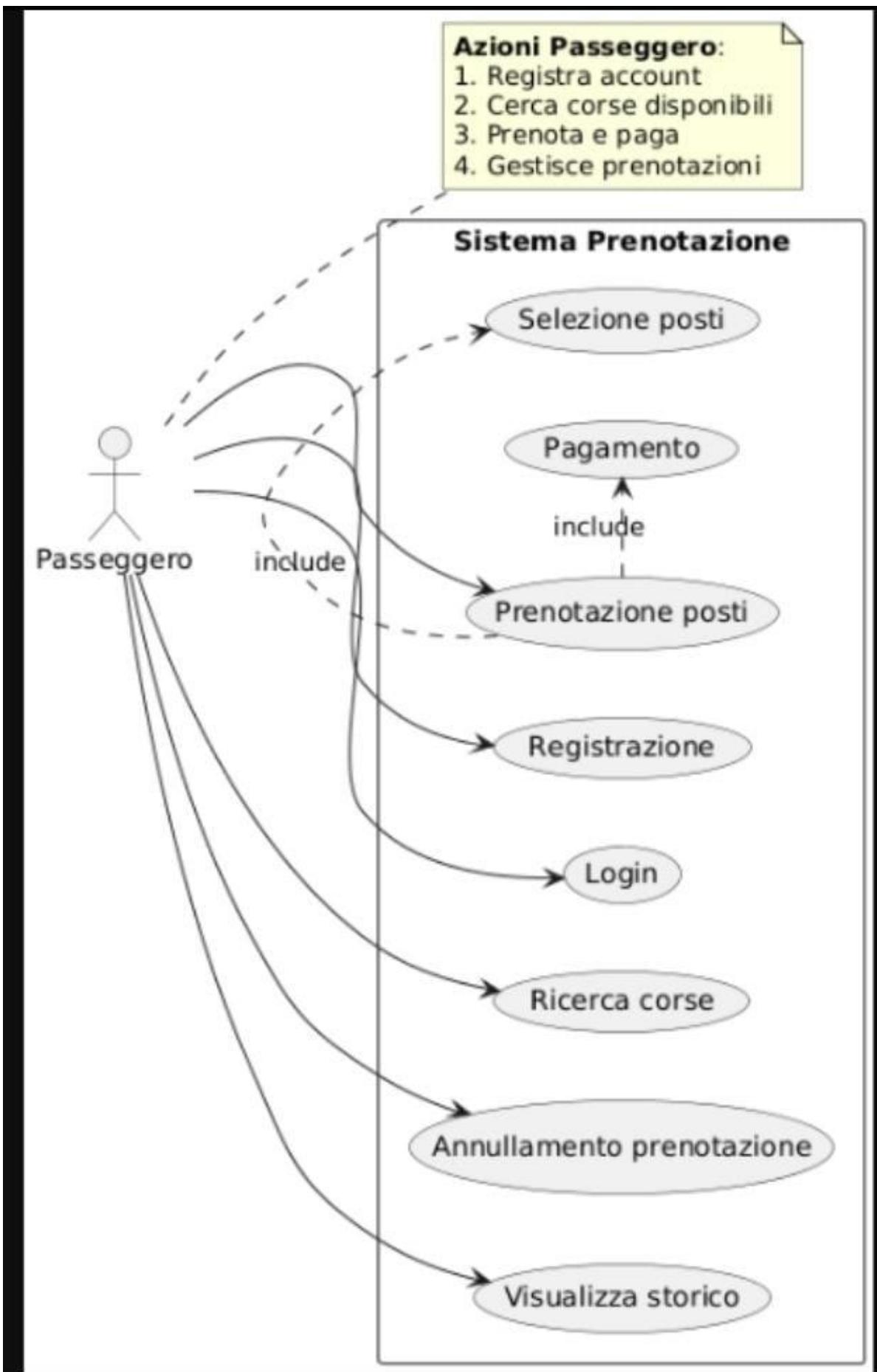
Precondizione : L'utente si è logato al sistema

- Flusso :
- 1) L'utente si loga al sistema
  - 2) L'utente entra i dati relativi al viaggio ( luogo e data di partenza , di arrivo) (Mockup #3)
  - 3) L'utente sceglie se vuole fare un andata ritorno o solo andata
  - 4) se il viaggio è andata ritorno sceglie anche la data di ritorno
  - 5) Il sistema genera una lista di viaggi disponibili corrispondenti ai dati inseriti con prezzo e numero di posti ancora disponibili per ciascun viaggio
  - 6) L'Utente sceglie un viaggio conveniente e conferma la prenotazione con il pagamento de relativo prezzo del viaggio ( Test #3)
  - 7) il sistema notifica l'utente dell'avvenuta con successo della prenotazione con informazioni relativi al viaggio
- Flusso alternativo : non ci sono viaggi disponibili relativi ai dati di viaggio inseriti

### 2.1.6 Diagramma dello stato di pagamento



## **Use case Diagram : Utente**



## 2.2. Processo Amministratore

### 2.2.1 Use case #6 : Aggiunta di una Corsa

Gli amministratori possono aggiungere nuove corse definendo:

- **Origine e destinazione**
- **Data e orari**
- **Autobus assegnato**
- **Numero di posti disponibili**
- **Prezzo del biglietto**

Una volta creata la corsa, questa diventa visibile agli utenti nel sistema di prenotazione.

### 2.2.2 Use Case #7: Cancellazione di una Corsa

Se necessario, l'amministratore può **cancellare una corsa**. Il sistema:

- Rimuove la corsa dall'elenco disponibile.
- Invia notifiche agli utenti che avevano prenotato.
- Gestisce eventuali rimborsi in base alla politica aziendale.

### 2.2.3 Use case #8 : Monitoraggio delle Statistiche

Gli amministratori possono visualizzare le **statistiche di occupazione**, tra cui:

- Numero di prenotazioni per corsa.
- Percentuale di riempimento degli autobus.
- Storico delle prenotazioni cancellate/modificate.

#### Use Case Template : Amministratore

Use case #6     Aggiunta di una corsa

Breve descrizione : si decide di aggiungere un nuovo viaggio alla lista dei viaggi

Attori : Admin

Precondizione : L'utente è registrato al sistema

Flusso :  
1) L'utente si loga al sistema  
2) L'utente accede all'apposita pagina dedicata agli amministratori

3) l'utente aggiunge una nuova corsa con dettagli ( luogo data e ora di partenza ,luogo e ora di arrivo e il prezzo )

Post-condizione : il viaggio è visibile dagli utenti generici

Use case #7 : cancellazione di una corsa

Breve descrizione : per qualche ragione si decide di cancellare una corsa

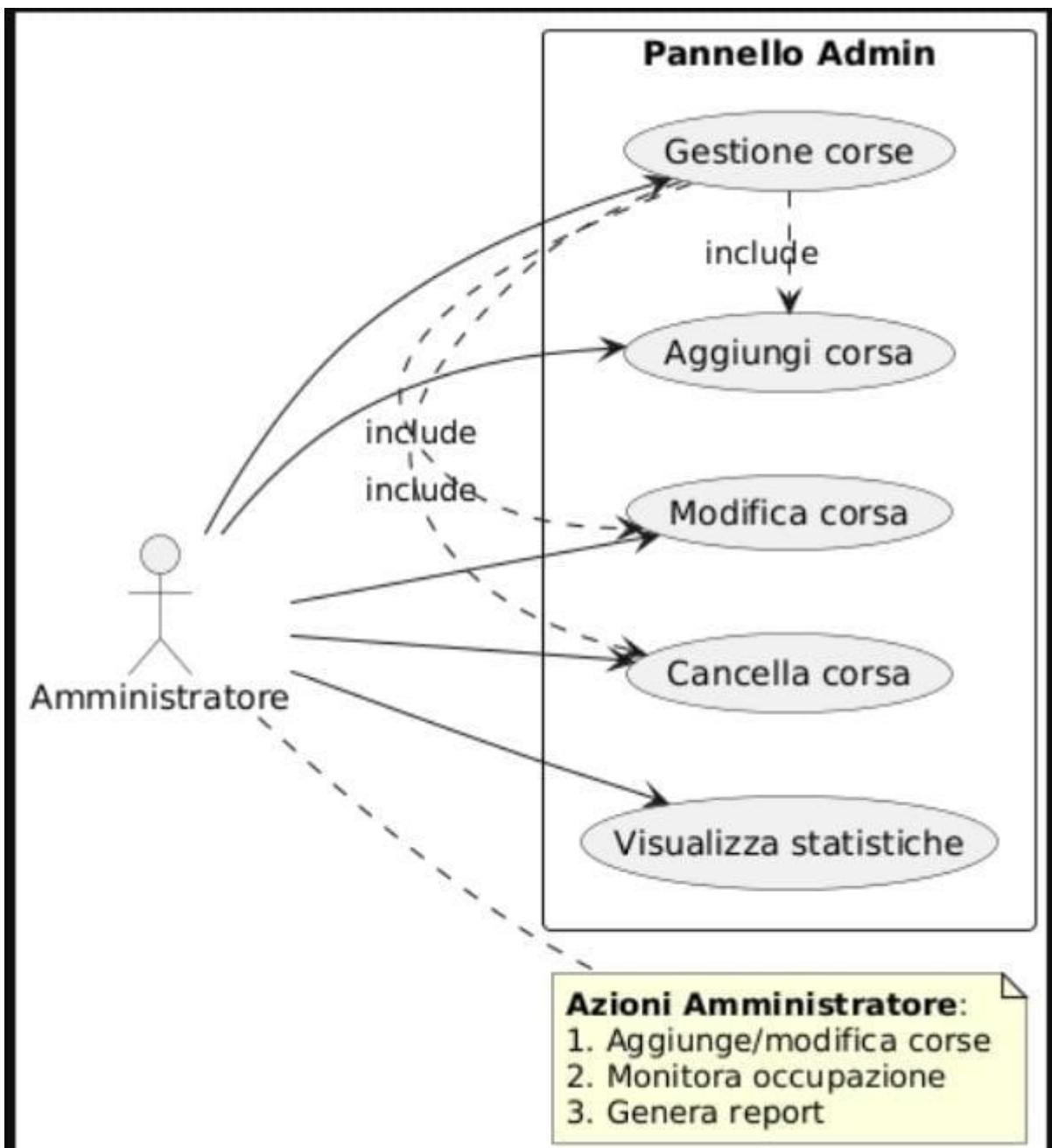
Attore : Admin

Precondizione : L'utente è già registrato al sistema

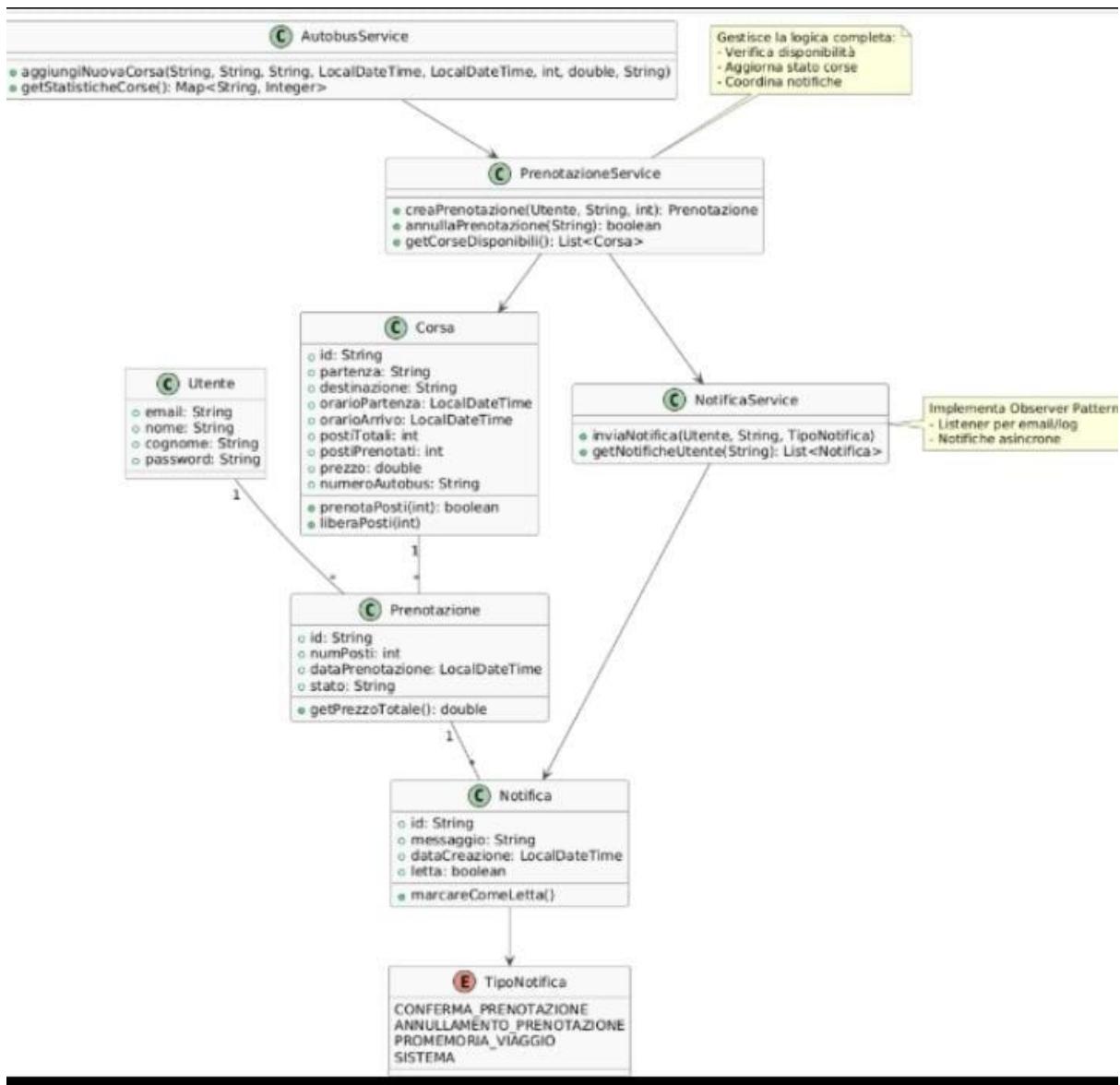
Flusso : 1) L'utente si loga al sistema  
2) L'utente accede all'apposita pagina dedicata agli amministratori  
3) sceglie il viaggio da cancellare e lo cancella

Post\_-condizione : il viaggio no è più visibile dagli utenti generici

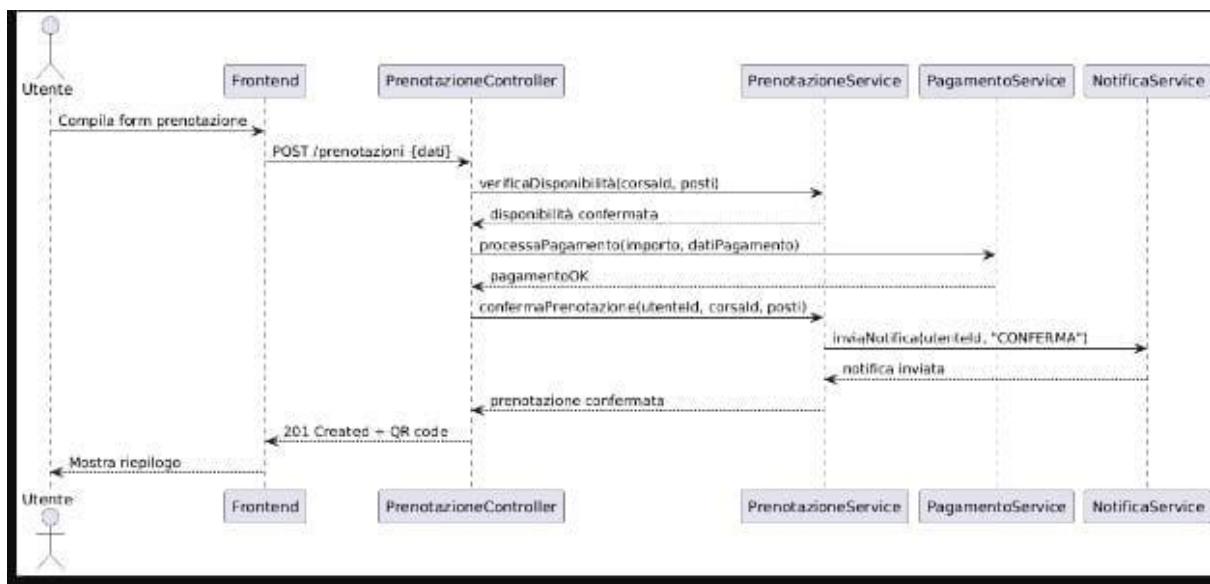
#### DIAGRAMMA DEI CASI D'USO : AMMINISTRATORE



#### 2.2.4. DIAGRAMMA DELLE CLASSI



### 2.3. diagramma delle attività



## 2.4. Testing

### *Obiettivi del Unit Testing*

Il Unit Testing di questo sistema mira a:

- Verificare la correttezza delle funzionalità principali.
- Identificare bug nelle singole unità di codice.
- Assicurare la modularità del sistema e facilitare le future integrazioni.
- Migliorare la manutenibilità e la stabilità del software.

le unità principali da testate includono:

- **Modulo di autenticazione:** Controlla il login e la registrazione degli utenti.

```

@Test
void testRegisterAndAuthenticateUser() {
    assertTrue(userService.registerUser("Mario", "Rossi",
    "mario@example.com", "password123"));
    assertNotNull(userService.authenticateUser("mario@example.com",
    "password123"));
    assertNull(userService.authenticateUser("mario@example.com",
    "wrongpassword"));
}

```

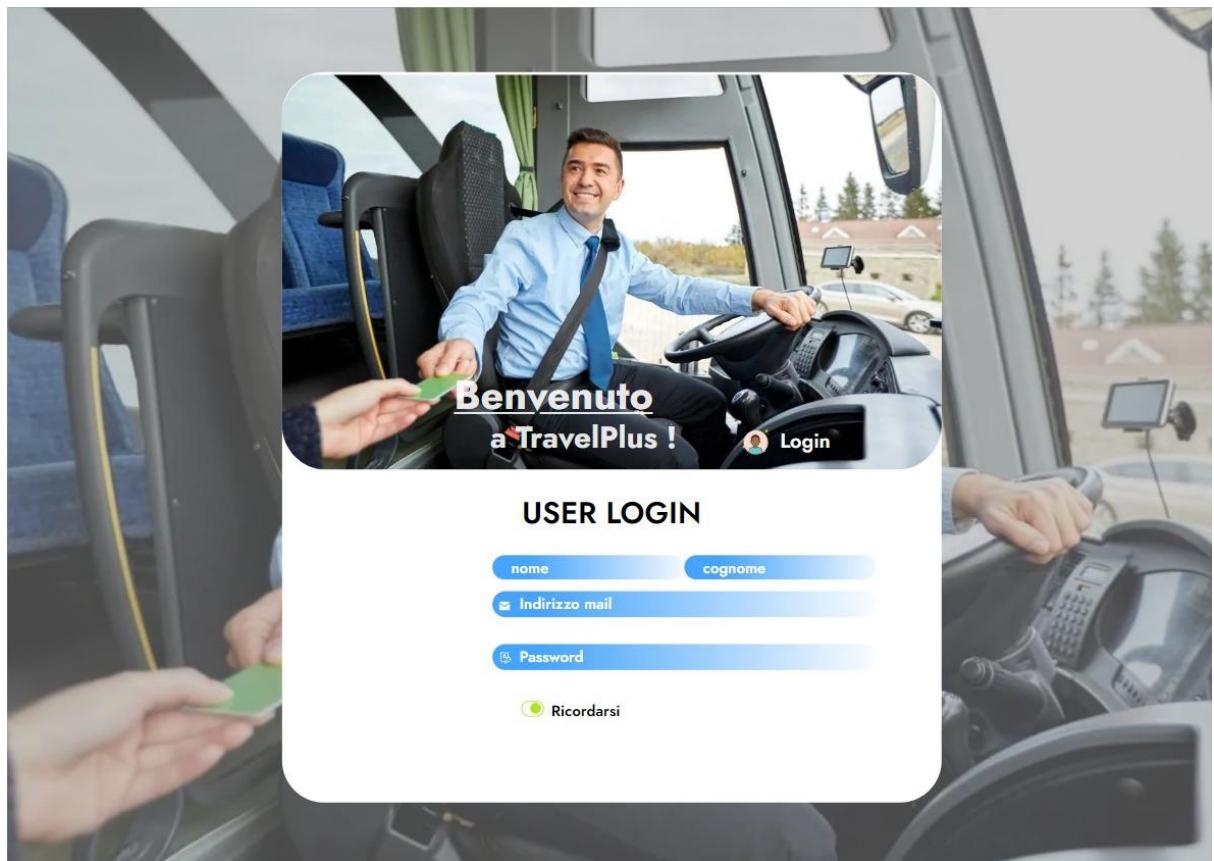
- **Gestione delle prenotazioni:** Verifica la corretta registrazione, modifica e cancellazione delle prenotazioni.

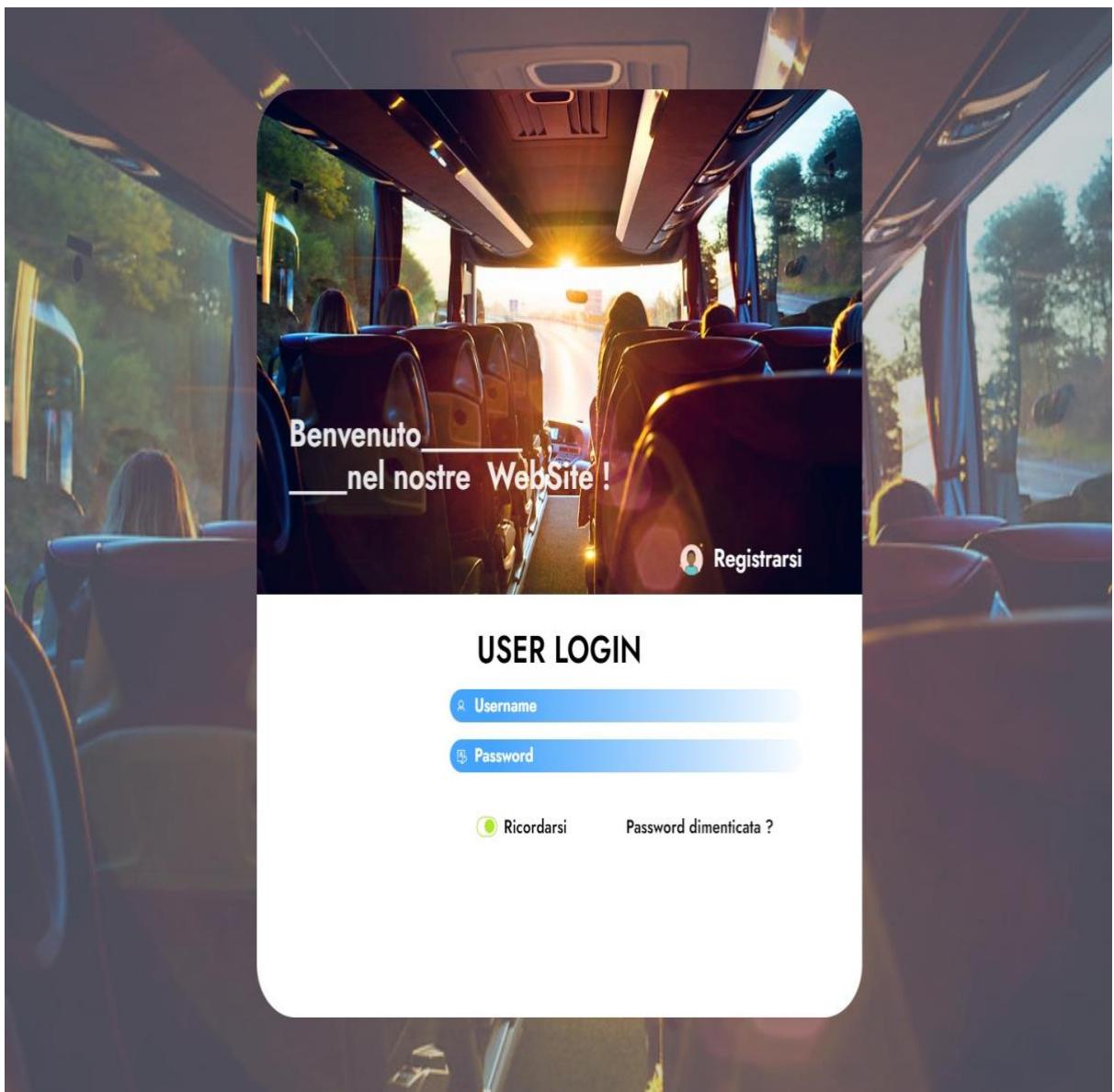
```
@Test
void testCreateAndCancelBooking() {
    Booking booking = bookingService.createBooking(testUser,
testTrip, "A1");
    assertNotNull(booking);
    assertEquals(1, testTrip.getBookedSeats().size());

    assertTrue(bookingService.cancelBooking(booking.getId()));
    assertEquals(0, testTrip.getBookedSeats().size());
    assertTrue(booking.isCancelled());
}
```

### 3. MOCKUP

I mockup sono stati realizzati con lo strumento Lunacy e simulano alcune funzionalità del sistema





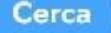
 Travel Plus

Planifica il tuo viaggio    Informazioni    Monitorare il viaggio    Auto



Solo Andata     Andata e ritorno

Da:  Firenze    A:  Bologna    Andata:  Oggi, 13 maggio    

**Cerca**