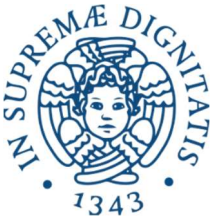
PROGETTO WOODSTOCK
INGEGNERIA DEL SOFTWARE



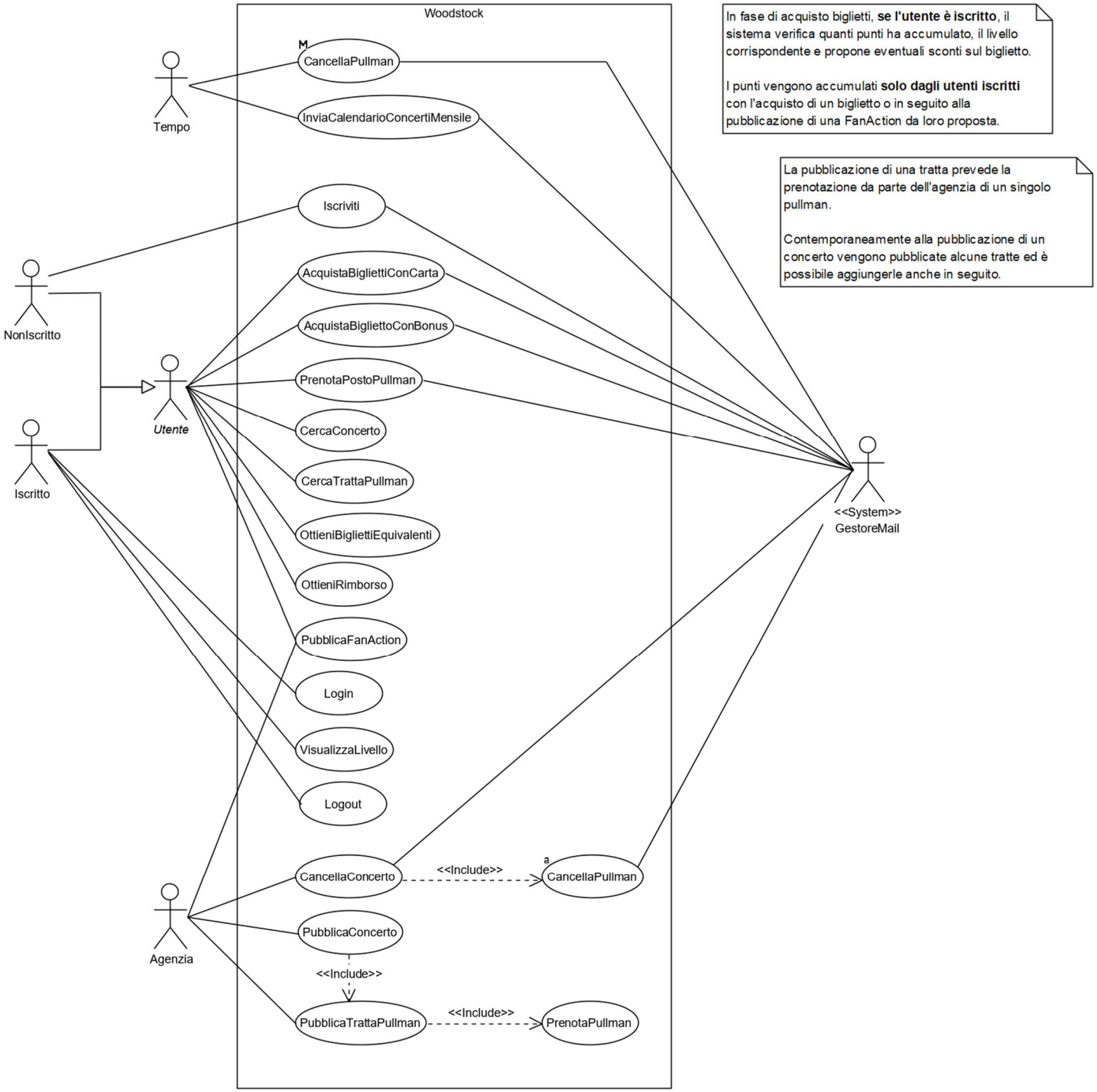
MEMBRI DEL GRUPPO

Niccolò Campitelli	583283
Radu Paraschiv Dobrila	584367
Elia Grassetti	582588

DOMANDE AL COMMITTENTE [risposta ipotizzata]

1. Con lo stesso account Bonus Cultura si possono acquistare più biglietti per lo stesso evento (con più buoni)?
   [No]
2. La pubblicazione di un concerto è sempre accompagnata dalla pubblicazione di una o più tratte del pullman?
   [Sì]
3. Possono essere inserite nuove tratte dopo la pubblicazione di un concerto?
   [Sì]
4. Possono esserci più pullman per la stessa tratta?
   [No, ma possono esserci più tratte con stesso percorso per concerto]
5. Quando vengono prenotati dei posti su un pullman non confermato, il sistema addebita subito il prezzo totale?
   [Sì]
6. Se si sceglie la spedizione a casa dei biglietti si riceve anche il codice QR per ritirarli?
   [No]
7. Se il concerto è cancellato e si sceglie il rimborso, vengono rimborsati anche i costi di spedizione dei biglietti?
   [Sì]
8. Se il concerto è cancellato e si sceglie il cambio data, è possibile ricevere il codice QR invece di effettuare nuovamente la spedizione?
   [No]
9. Dopo la cancellazione di un concerto, entro quanto tempo è possibile effettuare il rimborso o il cambio data?
   [Entro 12h dall'inizio del concerto]

[PUNTO 1] DIAGRAMMA DEI CASI D'USO



[PUNTO 1] NARRATIVA CASO D'USO

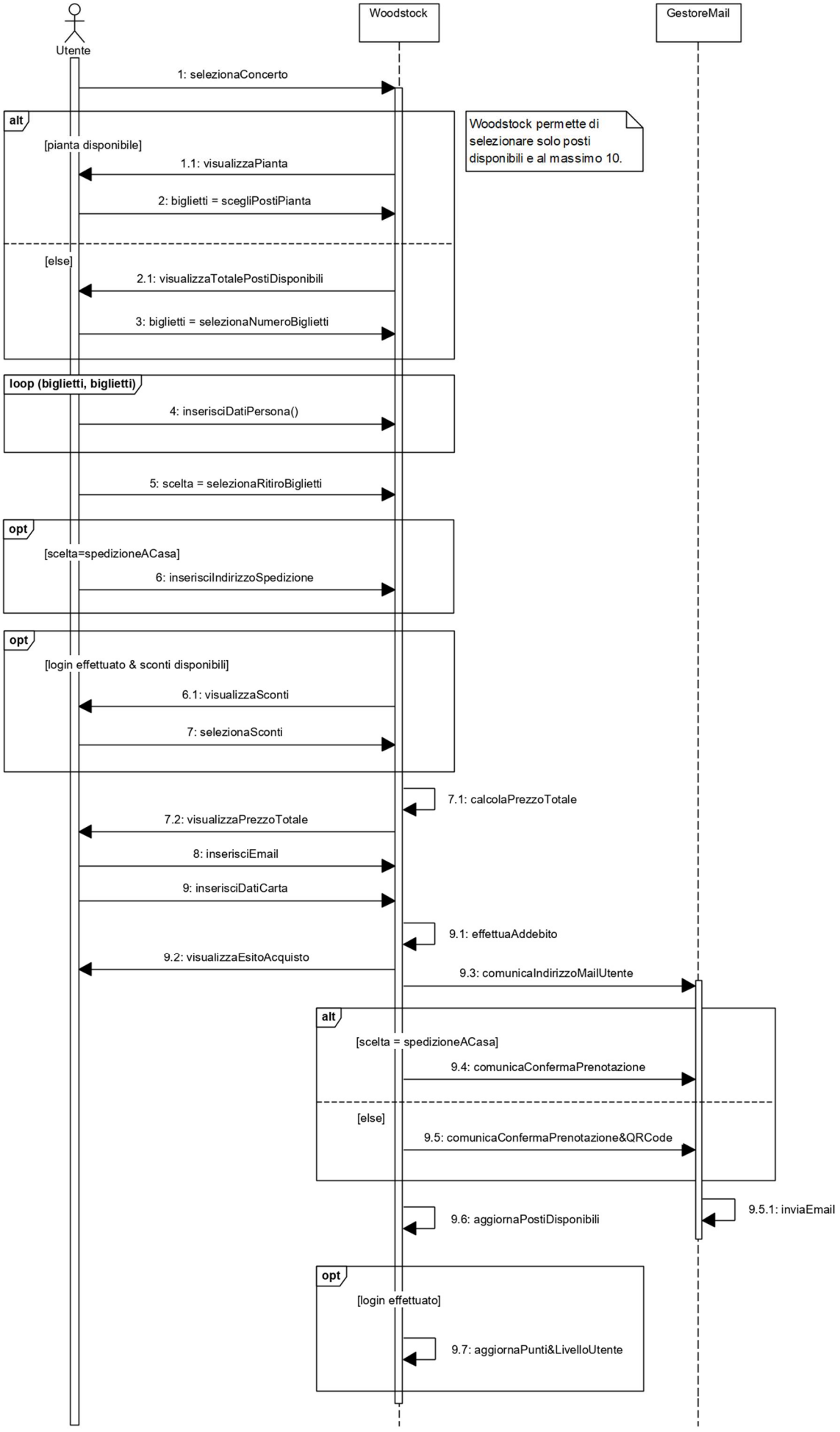
Nome: PrenotaPostoPullman
Breve descrizione: L'utente desidera prenotare un posto sul pullman
Attori primari: Utente
Attori secondari: GestoreMail
Precondizioni: (Utente ha almeno un biglietto)
Postcondizioni: (Posti del pullman prenotati) (Posti disponibili aggiornati)

Sequenza principale degli eventi:

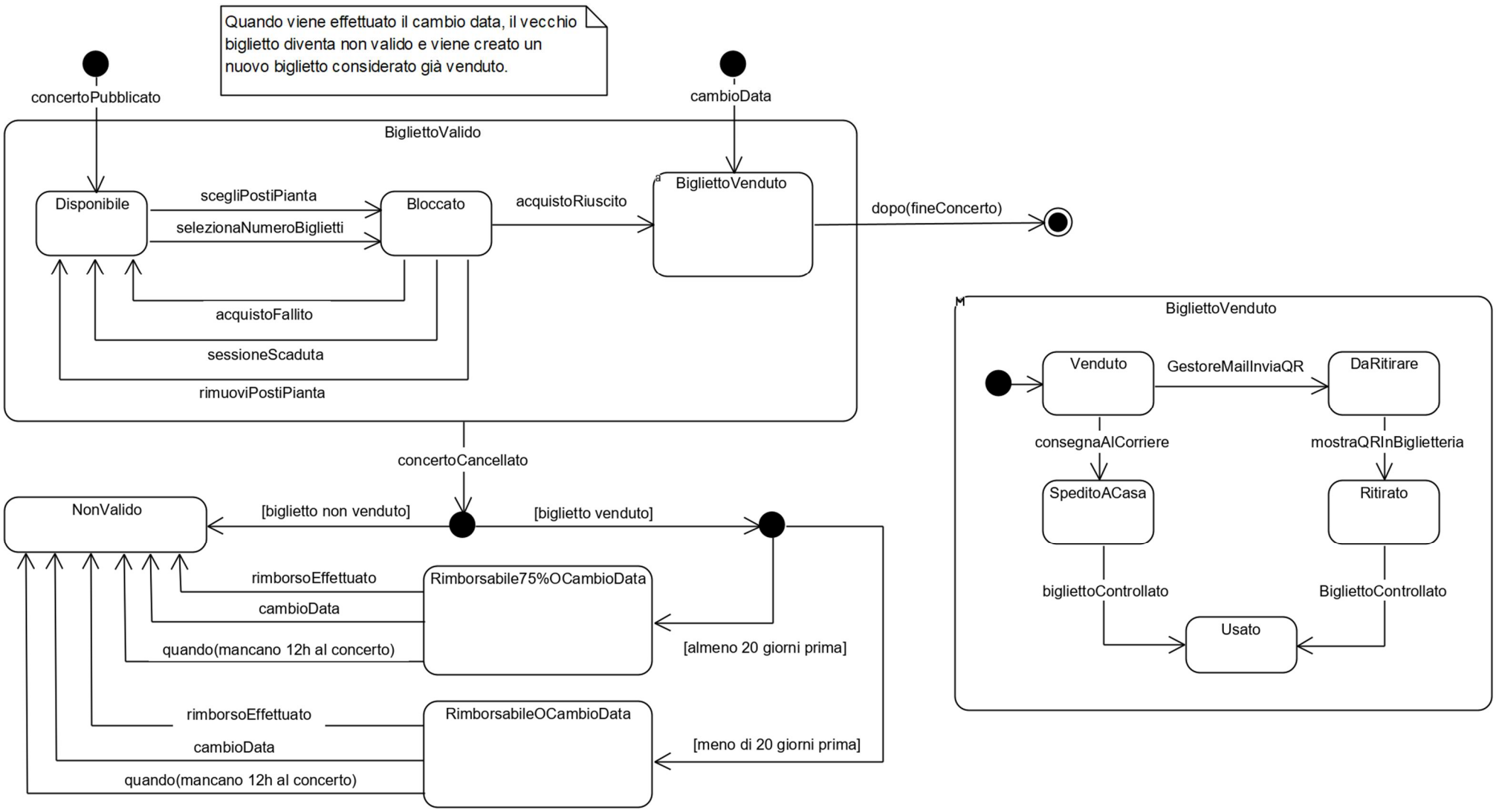
1. L'utente seleziona la tratta che vuole prenotare.
2. Il sistema mostra all'utente quanti sono i posti disponibili.
3. L'utente seleziona il totale di posti che vuole prenotare tra i disponibili.
4. L'utente inserisce un codice biglietto per ogni posto selezionato.
5. Il sistema verifica la validità dei codici biglietto.
6. Il sistema verifica che la tratta selezionata corrisponda al concerto di ogni biglietto.
7. L'utente inserisce email e carta di credito.
8. Il sistema verifica email e carta di credito.
9. Il sistema addebita il prezzo totale sulla carta di credito.
10. Il sistema memorizza email, carta di credito e codici biglietto.
11. Il sistema aggiorna il numero di posti disponibili.
12. SE(pullman già confermato)
    a. Il GestoreMail invia una mail all'utente confermando l'avvenuta prenotazione.
13. ALTRIMENTI SE(appena raggiunto o superato 40% dei posti prenotati)
    a. Il sistema conferma il pullman.
    b. Il GestoreMail conferma via mail la disponibilità del pullman a tutti gli utenti che avevano prenotato dei posti.

Sequenze alternative degli eventi: (pullman pieno) (codice biglietto non valido) (tratta non corrispondente al concerto) (email non valida) (carta di credito non valida) (pagamento fallito)

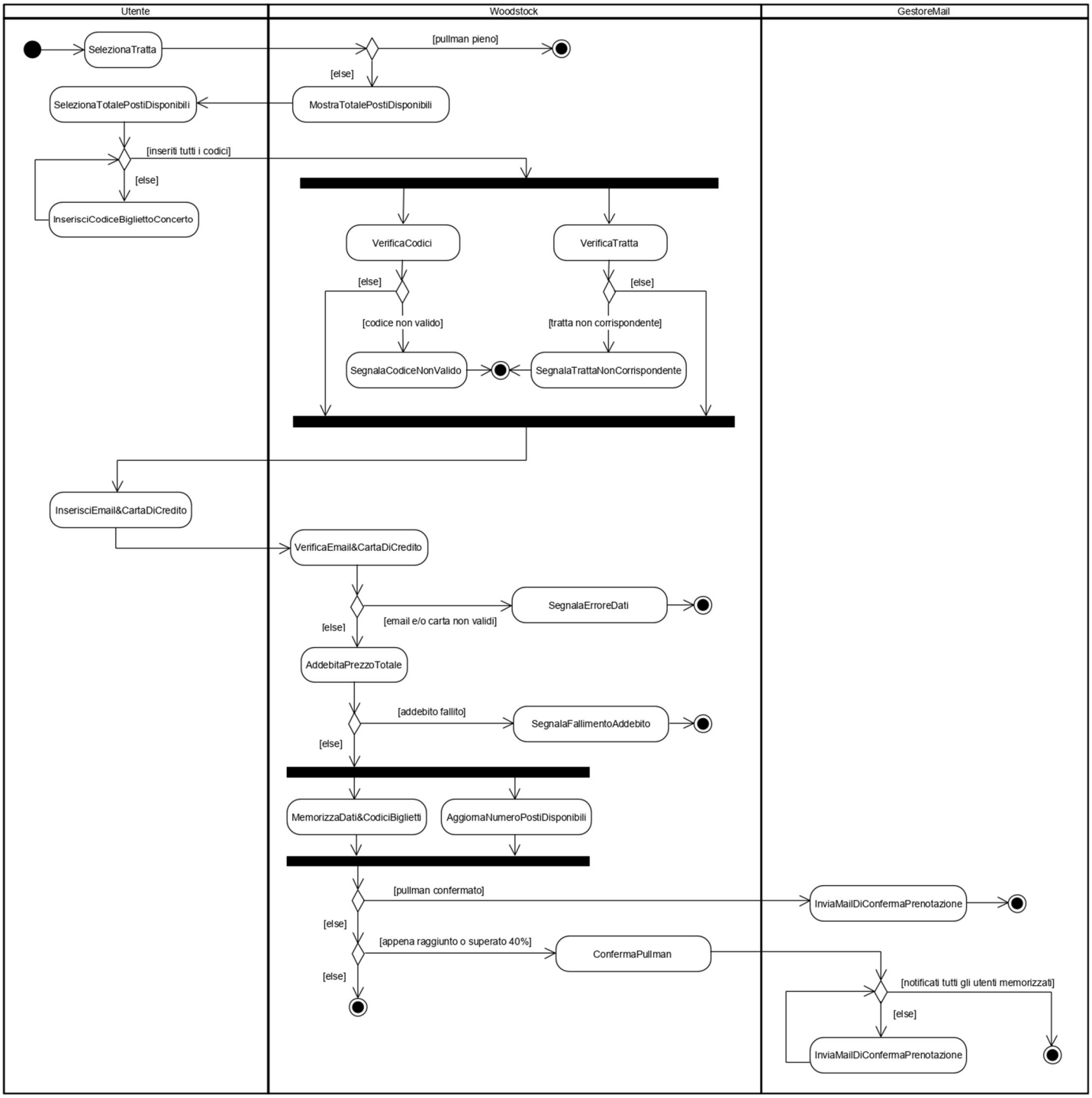
[PUNTO 2] DIAGRAMMA DI SEQUENZA - PrenotaPostoPullman



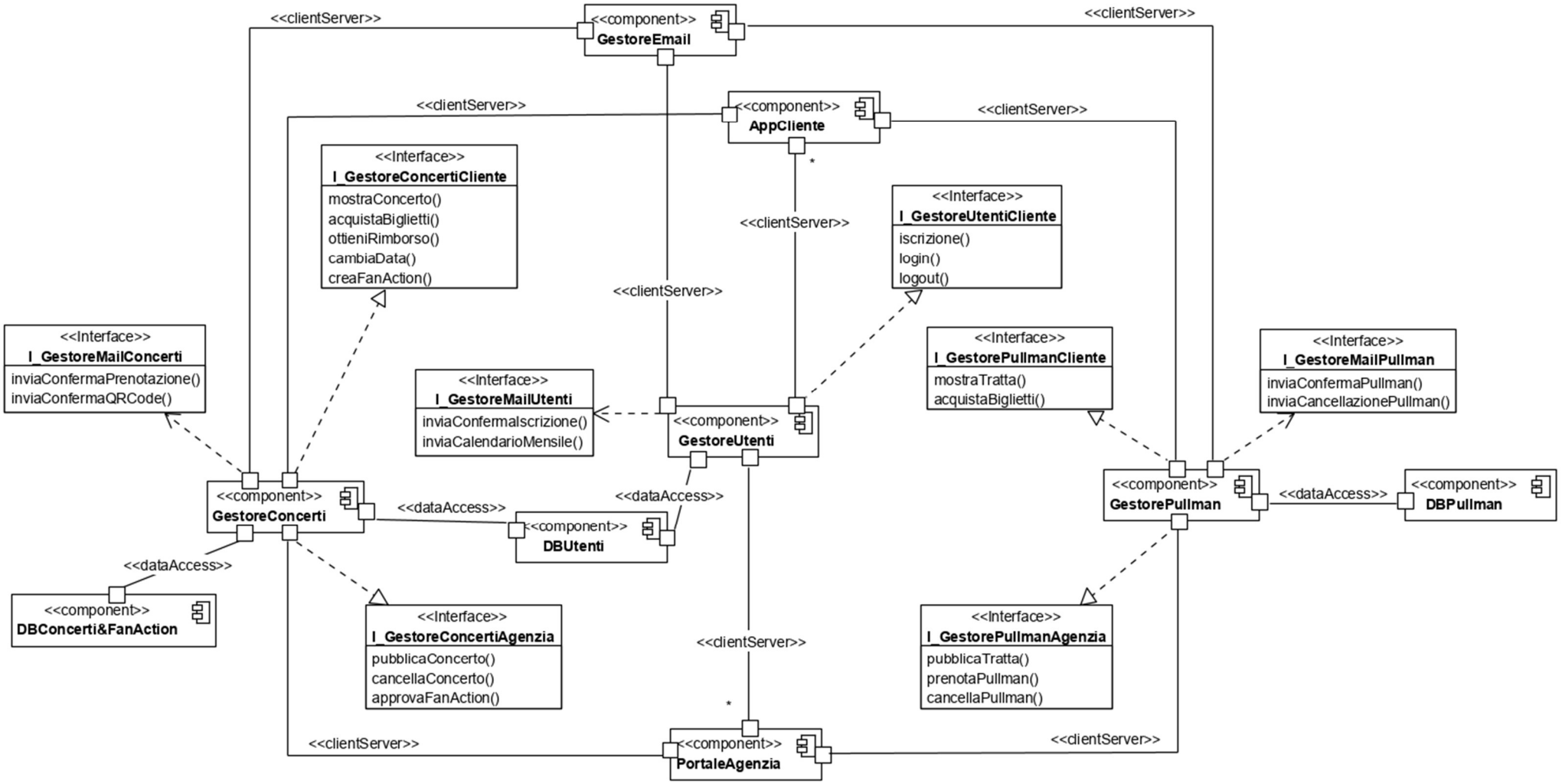
[PUNTO 3] DIAGRAMMA DI MACCHINA A STATI - Biglietto



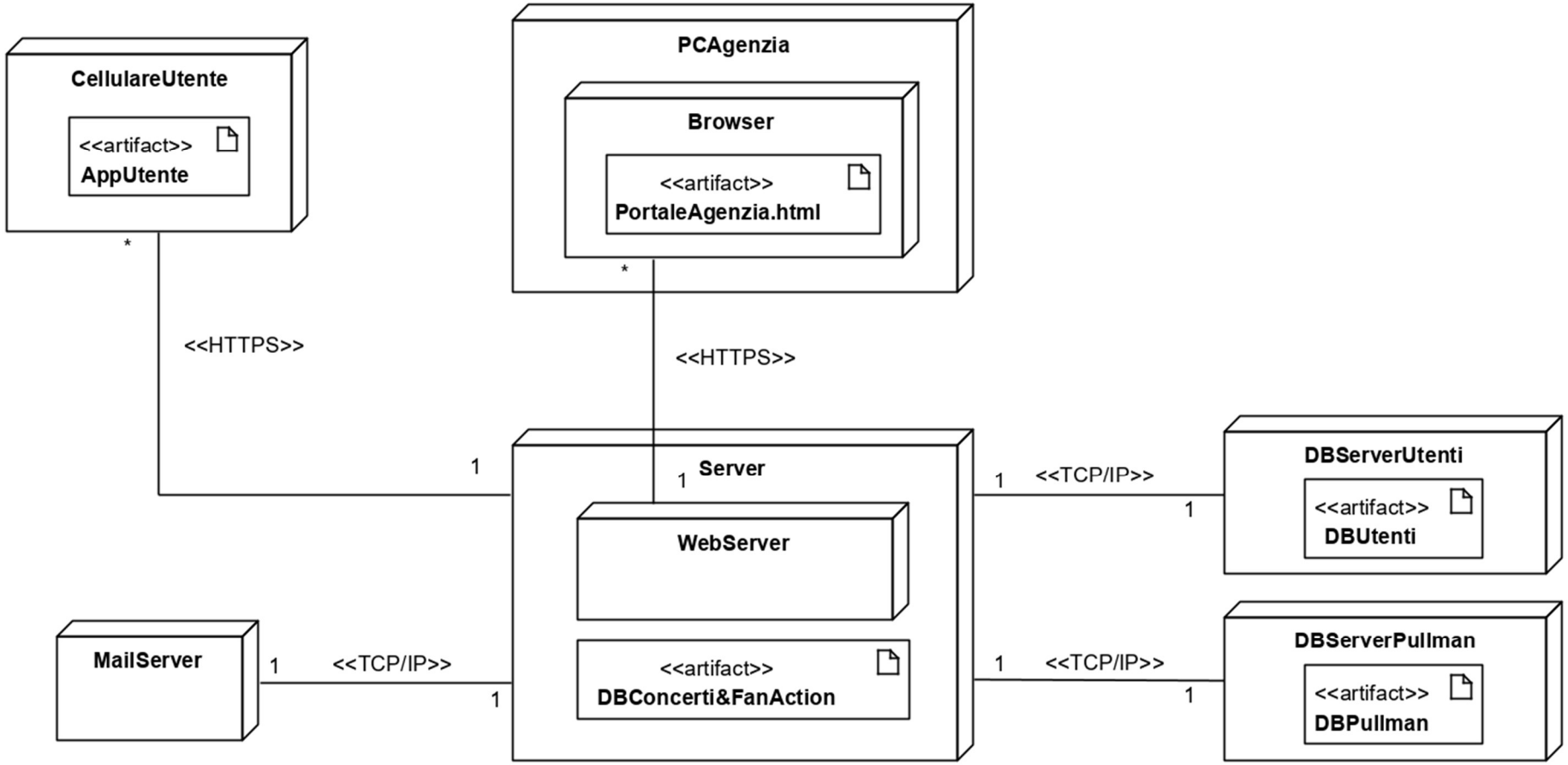
[PUNTO 4] DIAGRAMMA DI ATTIVITA' - PrenotaPostoPullman



[PUNTO 5] DIAGRAMMA COMPONENTI E CONNETTORI



[PUNTO 5] DIAGRAMMA DI DISLOCAZIONE



[PUNTO 6] IMPLEMENTAZIONE METODO – acquistaBase

```
int acquistaBase(Concerto c, int numeroBiglietti, Utente utente, boolean spedizione){
    if(c == null)
        throw new NullPointerException();
    if(numeroBiglietti <= 0 || numeroBiglietti > 10 || numeroBiglietti > c.getBigliettiDisponibili())
        throw new IllegalParameter();
    int prezzoTot = numeroBiglietti * c.getPrezzoBiglietti();
    String indirizzo;
    if(utente != null){
        int sconto = utente.getScontoLivello();
        prezzoTot = prezzoTot – (sconto*prezzoTot/100);
        utente.aggiornaLivello(prezzoTot);
        indirizzo = utente.getIndirizzo();
    }
    else
        indirizzo = inputIndirizzo();
    Carta carta = getCartaCredito();
    if(spedizione)
        prezzoTot +=7;
    if(!effettuaPagamento(carta, prezzoTot))
        throw new PaymentFailureException;
    if(spedizione)
        effettuaSpedizione(indirizzo, numeroBiglietti);
    return prezzoTot;
}
```

[PUNTO 6] PARTIZIONE DATI D'INGRESSO IN CLASSI – BATTERIA DI TEST

Si utilizzano due STUB per il metodo *effettuaPagamento* che restituiscono rispettivamente solo true o solo false. Si utilizza uno STUB per il metodo *effettuaSpedizione* che ritorna senza fare nulla.

CATEGORIA	ASPETTATIVA	CASI DI TEST
Concerto non valido	Errore	<**null**, 10, null, true> <**null**, 1, utente, false> con (utente!=null) <**null**, 5, utente, true> con (utente!=null)
Numero biglietti non valido	Errore	<c, **0**, null, true> con (c!=null) <c, **11**, null, false> con (c!=null) <c, **7**, utente, true> con (c!=null), (utente!=null), (**c.bigliettiDisponibili=5**)
Utente iscritto	Restituisce prezzo totale scontato in base al livello utente	<c, 2, **utente**, false> con (c!=null), **(utente!=null)**, (c.biglettiDisponibili=2) <c, 4, **utente**, true> con (c!=null), **(utente!=null)**, (c.biglettiDisponibili=6) <c, 10, **utente**, true> con (c!=null), **(utente!=null)**, (c.biglettiDisponibili=40)
Utente non iscritto	Restituisce prezzo totale non scontato	<c, 2, **null**, true> con (c!=null), (c.biglettiDisponibili=10) <c, 1, **null**, false> con (c!=null), (c.biglettiDisponibili=1) <c, 8, **null**, true> con (c!=null), (c.biglettiDisponibili=14)
Spedizione	Effettua spedizione e restituisce prezzo totale maggiorato	<c, 9, utente, **true**> con (c!=null), (utente!=null), (c.biglettiDisponibili=200) <c, 5, null, **true**> con (c!=null), (c.biglettiDisponibili=6) <c, 10, utente, **true**> con (c!=null), (utente!=null), (c.biglettiDisponibili=15)
Senza spedizione	Restituisce prezzo totale non maggiorato	<c, 6, utente, **false**> con (c!=null), (utente!=null), (c.biglettiDisponibili=13) <c, 3, null, **false** > con (c!=null), (c.**biglettiDisponibili**=55) <c, 3, utente, **false** > con (c!=null), (utente!=null), (c.biglettiDisponibili=10)

Pagamento fallito	Errore	<c, 7, utente, true> con (c!=null), (utente!=null), (c.biglettiDisponibili=7)
		<c, 2, null, true> con (c!=null), (c.biglettiDisponibili=19)
		<c, 5, utente, true> con (c!=null), (utente!=null), (c.biglettiDisponibili=52)
		
		Si utilizza lo STUB:
		effettuaPagamento(Carta carta, int prezzoTot){
		return **false**;
		}
		
		*per tutte le altre categorie si utilizza lo **STUB** che restituisce **true**.

[PUNTO 6] TESTING COMBINATORIO PER RIDURRE I CASI DI TEST

1) **Vincoli di errore**: si prende un singolo caso di test per ogni categoria che produce un errore.
2) **Vincoli di proprietà**: si producono casi di test in cui *spedizione=true* SSE *utente!=null* una volta garantito il corretto funzionamento del metodo *inputIndirizzo()*.
3) **Singoletti**: si fissa un concerto!=null con almeno un posto disponibile per tutti i casi di test.
4) **Pairwise testing**: si fissa un concerto!=null, si limita il range di numeroBiglietti in [1,10], si limitano le possibilità di utente a null o specifico utente!=null e si producono solo le combinazioni di casi di test per ogni coppia di valori diversi dei parametri.