



## Progetto di Ingegneria del Software 2

A.A. 2013/2014

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT  
for

# TRAVELDREAM

Casamassima Nicola

Lazzati Riccardo

Marcu Bogdan Alexandru

## Sommario

1. INTRODUCTION .....	4
1.1 PURPOSE OF THE DOCUMENT.....	4
1.2 SCOPE OF THE SYSTEM .....	4
1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS .....	5
1.4 REFERENCES INCLUDED IN THE RASD.....	5
1.5 OVERVIEW .....	5
2. OVERALL DESCRIPTION.....	6
2.1 PRODUCT PERSPECTIVE .....	6
2.1.1 System interfaces .....	6
2.1.2 User interfaces.....	6
2.1.3 Hardware interfaces .....	6
2.1.4 Software interfaces.....	6
2.1.5 Communication interfaces .....	7
2.1.6 Memory constraints .....	7
2.1.7 Operations .....	7
2.1.8 Site adaptation requirements.....	7
2.2 PRODUCT FUNCTIONS .....	8
2.2.1 Functional and non functional requirements.....	8
2.3 USER CHARACTERISTICS .....	9
2.4 CONSTRAINTS .....	9
2.4.1 Regulatory policies .....	10
2.4.2 Hardware limitations .....	10
2.4.3 Interfaces to other applications .....	10
2.4.4 Parallel operations.....	10
2.4.5 Audit functions .....	10
2.4.6 Control functions .....	10
2.4.7 High-order language requirements .....	10
2.4.8 Signal handshake protocol.....	11
2.4.9 Reliability requirements .....	11
2.4.10 Criticality of the application .....	11

2.4.11 Safety and security considerations.....	11
2.5 Assumptions and dependencies.....	11
2.6 Requirements to analize in the future.....	13
3.1 EXTERNAL INTERFACE REQUIREMENTS.....	13
3.1.1 User interfaces.....	13
3.2 FUNCTIONAL REQUIREMENTS.....	18
3.2.1 Attori.....	18
3.2.2 Scenari .....	18
3.3 NON FUNCTIONAL REQUIREMENTS .....	74
3.3.1 Performance requirements .....	74
3.3.2 Design constraints .....	74
3.3.3 Requisiti legati alla base dati .....	74
3.2.3 Vincoli generali di progetto .....	74
3.2.4 Standard adottati.....	74
3.3 SYSTEM ATTRIBUTES .....	75
3.4.1 Reliability .....	75
3.4.2 Availability .....	75
3.4.3 Security .....	75
3.4.4 Maintainability.....	75
3.4.5 Portability .....	75
4. APPENDICES.....	76
4.1 ALLOY CODE.....	76
<b>4.1.2 Signatures.....</b>	76
4.1.3 Fact .....	79
4.1.4 Assertion.....	82
4.1.5 Verifiche Assertions .....	83
4.1.6 Predicates .....	84
4.1.7 Verifica predicates .....	84
4.1.6 Mondi Generati .....	85

## 1.INTRODUCTION

### 1.1 PURPOSE OF THE DOCUMENT

La funzione di questo documento è di fornire una descrizione dettagliata di tutti i requisiti per lo sviluppo del sistema e-commerce TravelDream. In questo senso, verrà illustrato il principale scopo del sistema e allo stesso tempo, una descrizione di tutti i suoi componenti, completa di vincoli di sistema, di interfaccia e di interazione con l'esterno.

### 1.2 SCOPE OF THE SYSTEM

Il sistema di e-commerce richiesto è un applicativo che permette ai dipendenti di un'agenzia viaggi di poter gestire i loro componenti base(voli,alberghi,escursioni) e creare pacchetti-viaggio da poter rivendere ai propri clienti.

- L'applicativo deve permettere a un impiegato di creare,modificare o eliminare dal sistema i prodotti offerti della società, organizzarli in pacchetti e renderli disponibili alla vendita online.
  
- Il sistema deve permettere ad un cliente la registrazione, la creazione del proprio profilo personale, la creazione del proprio itinerario e la possibilità di visualizzare quei pacchetti che rispondono alle richieste del cliente stesso.  
L'applicativo dovrà guidare l'utente all'acquisto e consentirgli di modificare il pacchetto qualora quello proposto dal sistema non vada bene.
  
- L'utente deve essere in grado inoltre di poter condividere e consigliare ad alcuni amici il proprio pacchetto;gli amici da lui scelti saranno informati tramite email(servizio esterno al progetto) e,dopo averlo visualizzato e previa registrazione al sito stesso,potranno acquistare a loro volta il pacchetto consigliatogli.
  
- Il sistema infine deve permettere al cliente la creazione di una GiftList: esso deve poter scegliere alcuni componenti del pacchetto da lui scelto il cui pagamento verrà richiesto ad alcuni amici.
  
- Il sistema in futuro dovrà essere in grado di supportare la possibilità di fornire all'utente anche il servizio di trasporti (auto a noleggio,card per mezzi pubblici).

## 1.3 DEFINITIONS, ACRONYMS, ABBREVIATIONS

Vengono qui definiti alcuni termini con significati particolari che saranno usati all'interno del documento per riferirsi ad alcuni aspetti del progetto.

### ❖ Acronimi

- RASD: Requirements Analysis and Specification Document
- DBMS: Data Management System
- FTP: File Transfer Protocol
- TCP: Trasmission Contol Protocol

### ❖ Definizioni:

- Utenti Registrati: utenti iscritti al sistema del quale possono usare tutte le funzionalità
- Utenti non registrati: utenti della rete non iscritti al sistema, visitatori; possono effettuare solo operazioni di ricerca e visualizzazione
- Impiegato: utente del sistema che si occupa della gestione backend del sito
- Piattaforma, Applicativo: il sito TravelDream
- Alloy: linguaggio per testare la consistenza del modello relazionale

## 1.4 REFERENCES INCLUDED IN THE RASD

Tutte le informazioni ci sono state fornite tramite documentazione messaci a disposizione dai docenti sulla piattaforma BEEP.

## 1.5 OVERVIEW

Il documento è diviso in 4 parti:

1. **Introduction**, dove vengono presentati gli aspetti generali del sistema.
2. **Overall Description**, dove si definiscono i requisiti del sistema, gli utenti destinati ad utilizzarlo, le funzionalità che il sistema metterà a loro disposizione, e allo stesso tempo, esplicitare e descrivere le limitazioni del sistema e le assunzioni prese sul dominio.
3. **Specific Requirements**, dove verrà mostrata un'analisi dettagliata dei requisiti, in particolare tramite l'uso di scenari e use case. Proseguendo con l'analisi, le funzionalità per soddisfare i requisiti saranno evidenziate tramite i diagrammi UML.
4. **Appendices**, che contiene il modello Alloy utilizzato per l'analisi formale dei requisiti.

## 2. OVERALL DESCRIPTION

Questa sezione fornisce una visione d'insieme di tutto il sistema, in tutti i suoi contesti, per mostrare l'interazione di esso col mondo esterno e allo stesso tempo, descrivere le sue macro-funzionalità. Vengono mostrati gli attori del sistema e, per ognuno di essi, le funzionalità messe a disposizione. In fine, le limitazioni ed assunzioni per il sistema saranno presentate.

### 2.1 PRODUCT PERSPECTIVE

#### 2.1.1 System interfaces

Essendo un sistema e-commerce web-based, dovrà interfacciarsi con un database dove saranno contenuti i dati. Il software che sarà implementato non dovrà interfacciarsi con altri sistemi dello stesso tipo.

#### 2.1.2 User interfaces

Le interfacce utenti saranno chiare ed intuitive dal punto di vista grafico, offrendo un'esperienza di navigazione veloce per rendere l'utilizzo del sistema piacevole e rassicurante. L'esperienza dell'utente nell'utilizzare il sistema è fondamentale per sistemi e-commerce, quindi per l'interfaccia utente si dovrà prestare particolare attenzione.

#### 2.1.3 Hardware interfaces

L'interfaccia hardware prevista è un normale terminale client (PC o altro) dotato di una connessione internet. Per ora non è prevista un'implementazione per dispositivi mobile.

#### 2.1.4 Software interfaces

Il software sarà basato su un'interfaccia web, per cui sarà necessario un qualsiasi browser e una connessione internet. Come sistemi operativi saranno supportati Windows, Mac e Linux, invece per quanto riguarda i browser si consiglia Google Chrome, vista la disponibilità su tutti i sistemi operativi. Il sistema dovrebbe essere comunque funzionante anche sugli altri browser più popolari, come ad esempio Opera o Mozilla. Per quanto riguarda Internet Explorer e Safari, la compatibilità dovrà essere testata.

## 2.1.5 Communication interfaces

Le interfacce di comunicazione del sistema riguardano essenzialmente i protocolli di rete necessari per il trasferimento delle informazioni sul web. In particolare le seguenti interfacce e relative applicazioni sono alla base del corretto funzionamento del sistema.

Protocollo di Rete	Porta	Applicazione
TCP/HTTP/HTTPS	80	Browser
TCP/FTP	<porta del DBMS>	DBMS
FTP	21	Client FTP o repository remoto

Inoltre il sistema si appoggerà ad un servizio di mail esterno per la gestione degli inviti tra utenti e non. Tale servizio ovviamente farà uso di protocolli SMTP e POP3 o IMAP e di client-server per email.

## 2.1.6 Memory constraints

Per quanto riguarda la memoria centrale, non ci sono particolari esigenze. Per quanto riguarda la memoria di massa, dobbiamo assolutamente garantire lo spazio per contenere i sorgenti del sistema e le tabelle del database che si vanno a creare, per cui si consiglia una memoria fisica di 2-3 GB.

In ogni caso è buona norma prevedere una memoria di dimensioni maggiori di quanto effettivamente necessario, al fine di prevenire una crescita esponenziale del numero degli utenti registrati del sistema. Per questo si può pensare di avere una ridondanza di memoria secondaria nell'ordine dei 50 GB.

## 2.1.7 Operations

Le operazioni degli utenti registrati, non registrati e degli impiegati sono descritte nella sezione 2.2 “Product functions”.

## 2.1.8 Site adaptation requirements

Per quanto riguarda il server su cui risiederà la logica applicativa ed il database del software, al fine di installare il prodotto software in maniera corretta, è necessario che tale server abbia installati i seguenti componenti:

- 1) una Java Virtual Machine (JVM);
- 2) l'Application Server (AS) che sarà determinato nella successiva fase di sviluppo, quella di Design;
- 3) un DataBase Management System (DBMS);
- 4) una memoria secondaria con uno spazio sufficiente.

Per quanto riguarda gli utenti del sistema, affinché possano utilizzare il sistema è sufficiente che dispongano di un browser per la navigazione web installato sul proprio dispositivo di accesso alla rete. Tuttavia, il sistema, per quanto compatibile con tutti i browser, verrà testato ed ottimizzato solo per una ristretta gamma di questi ultimi.

## 2.2 PRODUCT FUNCTIONS

### 2.2.1 Functional and non functional requirements

I requisiti funzionali e non funzionali vengono riportati in base agli attori, definiti come segue:

1. Utente registrato
2. Utente non registrato
3. Impiegato

L'elenco dei requisiti tiene conto delle assunzioni definite nella sezione "2.5 Assumptions and dependencies". La soddisfazione dei requisiti elencati di seguito comporta il sistema a rispettare gli obiettivi.

#### 2.2.1.1 Utente registrato

I requisiti funzionali dell'utente registrato corrispondono alle azione ed attività che l'utente può compiere all'interno del sistema, quindi:

- Login
- Cercare un viaggio
- Personalizzare un viaggio
- Creare una Giftlist
- Invitare amici a un viaggio
- Acquistare un viaggio
- Acquistare prodotti di una Giftlist creata
- Aderire a un viaggio
- Acquistare prodotti di una Giftlist creata da un altro utente registrato
- Visualizzare cronologia dei viaggi

I requisiti non funzionali dell'utente registrato sono i seguenti:

- Il sistema deve gestire una grande quantità di utenti
- Le informazioni relative ai viaggi devono essere salvate in modo sicuro
- Disporre di una connessione internet

#### 2.2.1.2 Utente non registrato

Similmente a quanto detto per l'utente registrato, i requisiti funzionali dell'utente non registrato corrispondono alle azioni che esso può compiere all'interno del sistema, per cui:

- Visualizzare un viaggio a cui è stato invitato
- Visualizzare una Giftlist a cui è stato invitato
- Registrarsi

I requisiti non funzionali dell'utente non registrato sono i seguenti:

- Disporre di una connessione internet

#### 2.2.1.3 Impiegato

I requisiti funzionali dell'impiegato corrispondono alle azioni che l'impiegato può compiere all'interno del sistema, cioè:

- Login
- Creare un prodotto base
- Modificare un prodotto base
- Eliminare un prodotto base
- Creare un pacchetto predefinito

I requisiti non funzionali dell'impiegato sono i seguenti:

- Disporre di una connessione internet

### 2.3 USER CHARACTERISTICS

Non ci sono restrizioni per l'utenza che utilizza il sistema, chiunque può avere accesso al sistema ed usufruire delle sue funzioni, indipendentemente dal sesso, età, posizione sociale, titolo di studio, posizione geografica ecc.

### 2.4 CONSTRAINTS

#### 2.4.1 Regulatory policies

Il sistema non ha una politica di regolazione.

#### 2.4.2 Hardware limitations

Le limitazioni sono le minime richieste sulla memoria e sulla capacità del server a gestire l'utenza.

#### 2.4.3 Interfaces to other applications

Il sistema deve interfacciarsi con i sistemi esterni di pagamento e di gestione mail.

#### 2.4.4 Parallel operations

Il sistema dovrà essere in grado di gestire più connessioni contemporaneamente, specialmente durante le sessioni loggate degli utenti registrati. In particolare, si dovrà garantire il corretto funzionamento durante la fase di ricerca e selezione del viaggio, in quanto funzione base del sistema. Quindi il sistema dovrà supportare un numero abbastanza grande di ricerche.

#### 2.4.5 Audit functions

Il sistema non prevede alcun controllo di revisione di questa versione. È comunque gradita una futura implementazione di revisione.

#### 2.4.6 Control functions

Il sistema non prevede interfacce fisiche dedicate.

#### 2.4.7 High-order language requirements

Il sistema verrà sviluppato usando il linguaggio Java e, in particolare, le tecnologie J2EE.

## 2.4.8 Signal handshake protocol

Il sistema non prevede la stesura di un protocollo per la comunicazione ad hoc.

## 2.4.9 Reliability requirements

Per garantire l'affidabilità del sistema, si chiede un back-up regolare del database in modo da soddisfare la correttezza del funzionamento anche durante una perdita di dati improvvisa o comunque un guasto.

## 2.4.10 Criticality of the application

L'aspetto fondamentale sarà quello di tenere traccia dei dati salvati nel database - i dati degli utenti registrati, dei loro viaggi e giftlist, i dati degli impiegati e dei prodotti base - in quanto da questi dati dipende l'integrità del sistema.

## 2.4.11 Safety and security considerations

Il sistema prevede la limitazione degli utenti non registrati alle sole visualizzazioni di viaggi o giftlist a cui sono stati invitati, a meno di non effettuare una registrazione nel sistema. In più, gli utenti registrati e gli impiegati possono usufruire delle funzioni del sistema solamente dopo il login.

## 2.5 Assumptions and dependencies

Prima della specifica dei requisiti, è doveroso fare alcune assunzioni e precisazioni atte ad avere il controllo globale dei vincoli e di ciò che ci si aspetta dal sistema:in seguito all'analisi di ciò che ci è stato richiesto di realizzare,abbiamo steso i seguenti vincoli per delimitare un profilo specifico e non ambiguo dell'applicazione:

- Il pacchetto predefinito contiene volo e albergo e una parte optional cioè le escursioni.
- L'utente inizia selezionando luogo e destinazione e numero di partecipanti. In base alle sue scelte, il sistema gli mette a disposizione diversi pacchetti predefiniti i quali soddisfano i vincoli messi dall'utente.
  - L'utente può entrare nella sezione modifica del pacchetto dove può andare a modificare il volo(A/R),l'albergo oppure togliere e/o aggiungere escursioni

- in caso di modifica ,il sistema sarà in grado di filtrare i contenuti(volli,alberghi disponibili per la scelta,escursioni)in base alla scelta di data,destinazione e numero di persone per il quale l'utente sta prenotando,rendendo disponibili solo quei componenti che soddisfano le condizioni
- nella sezione “Modifica” è contenuta la parte di “Condividi il pacchetto” con un amico e la parte “Crea la tua GiftList”
- Nel caso l'utente decida di creare una giftlist oppure condividere il viaggio,gli verrà chiesto di inserire gli indirizzi mail dei destinatari nell'apposito form
  - Nel caso scelga la condivisione con un amico,terminate la fase di immissione mail dei destinatari ,il sistema richiede il pagamento,decrementa i posti del numero pari a quello prenotato dall'utente e la transazione si chiude.
  - nel caso in cui un utente decida di creare la GiftCard ,dopo l'inserimento delle mail dei destinatari.il sistema lo invita a selezionare quali componenti del pacchetto aggiungere alla lista e ,successivamente,pagare l'importo differenza tra ciò che è il costo totale del pacchetto e ciò che viene pagato nella giftlist.Una volta inviate le mail,parte un timeout il quale,tutto ciò che non risulta pagato fino a 3 giorni prima del viaggio,viene automaticamente addebitato all'utente principale.
- Un utente registrato può accedere al viaggio condivisogli da un amico tramite il link mandatogli tramite mail(valido fino 3 giorni prima dalla partenza del viaggio),visualizzarne i dettagli e,qualora interessato,acquistarlo(se i posti sono ancora disponibili)
  - un utente esterno che riceve la mail con l'invito al viaggio,può visualizzare i contenuti del pacchetto accedendo al sito tramie link riportato nella mail ma,qualora voglia acquistarlo,deve registrarsi alla piattaforma,la visualizzazione del viaggio sarà possibile fino a un massimo di 3 giorni prima della data di partenza.
- un utente esterno che riceve la mail con l'invito a partecipare a una GiftCard può visualizzarne i dettagli (fino 3 giorni prima) e ,qualora voglia aderire,deve registrarsi al sito
- un utente registrato che riceve la mail con l'invito a partecipare a una GiftCard può visualizzarne i dettagli (fino 3 giorni prima) e ,qualora voglia aderire,selezionare dalla lista quale componente del pacchetto pagare
- Gli impiegati possono gestire i componenti dei pacchetti aggiungendoli,modificandoli e eliminandoli e con essi creare i pacchetti .
- Gli impiegati sono già registrati nel sistema,l'amministratore di sistema ha già proceduto ad inserire le loro credenziali nel database per mezzo di software terziario,non è prevista la parte di registrazione dell'impiegato

## 2.6 Requirements to analize in the future

Non vi sono grossi sviluppi programmati per il futuro. L'unico, che merita di avere una struttura già pensata in fase di progettazione, è quello di una estensione futura all'utilizzo di mezzi di trasporto alternativi all'aereo come treni, autobus.

Per far ciò si realizzerà il database e l'intero sistema tenendo conto di questa esigenza.

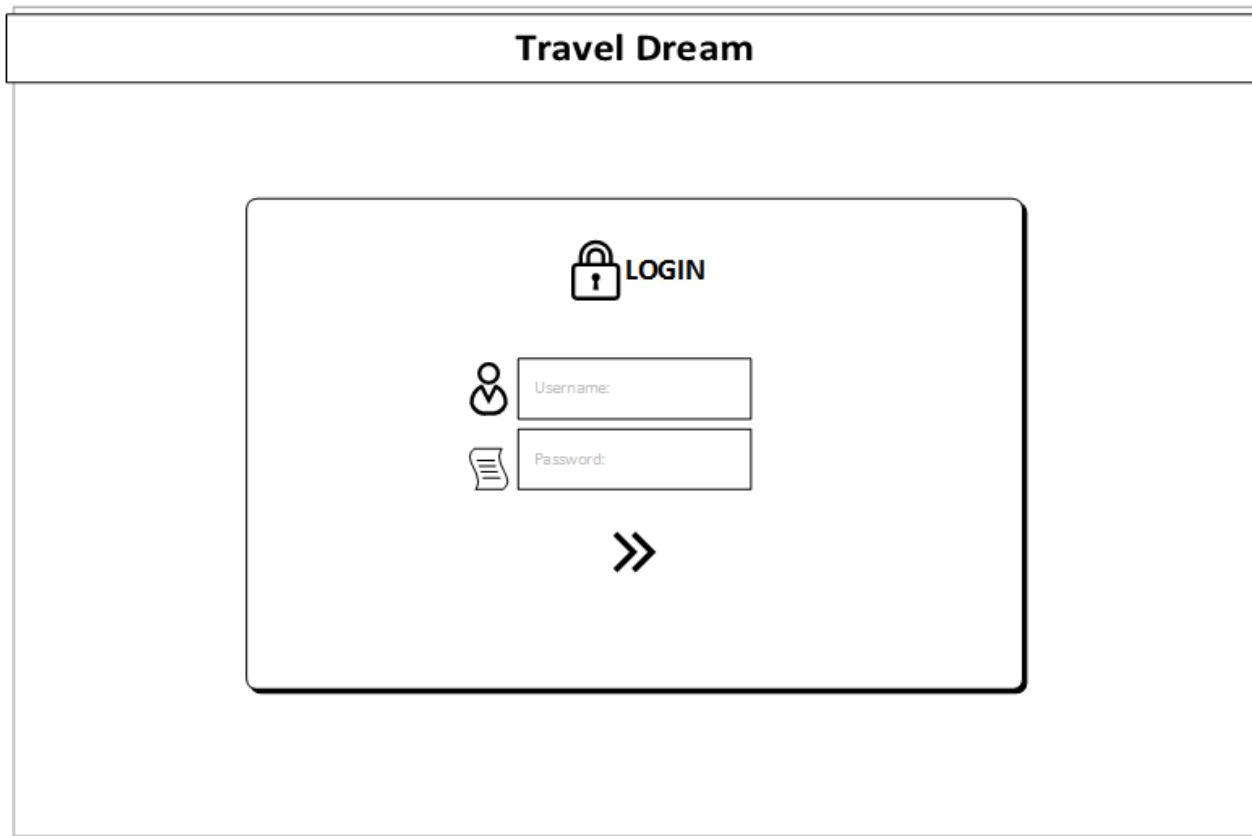
## 3. SPECIFIC REQUIREMENTS

### 3.1 EXTERNAL INTERFACE REQUIREMENTS

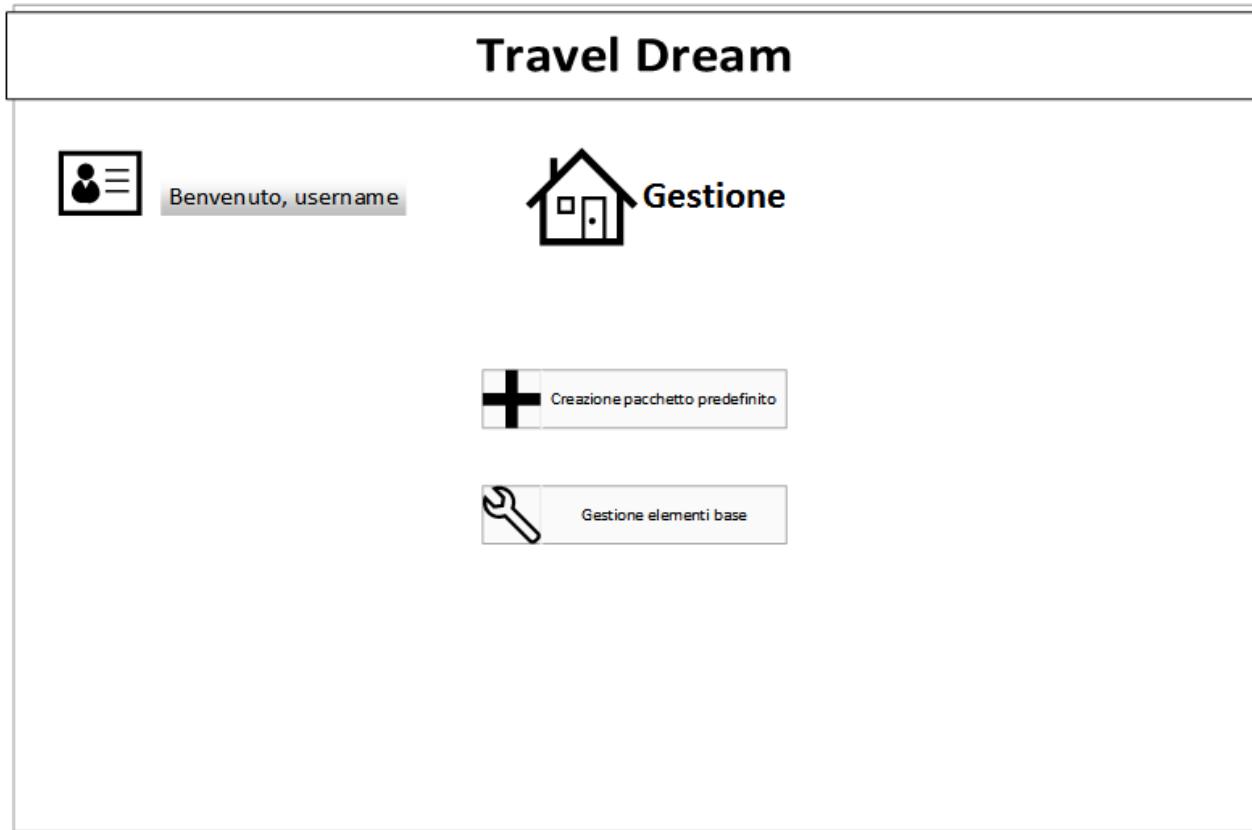
#### 3.1.1 User interfaces

Per quanto riguarda l'interfaccia utente, abbiamo realizzato alcuni mock ups per evidenziare gli aspetti fondamentali del sistema dal punto di vista grafico. Ci siamo concentrati su un aspetto minimalista, chiaro e moderno in modo che gli utenti si concentrino sull'esperienza di utilizzo senza particolari problemi legati alla navigazione.

Come primo punto, sia l'utente registrato sia l'impiegato si troverà di fronte alla pagina di login. Non abbiamo considerato il caso dell'utente non registrato in quanto all'accesso, dopo un invito a viaggio o giftlist, riceverà un semplice pop up che lo invita alla registrazione.



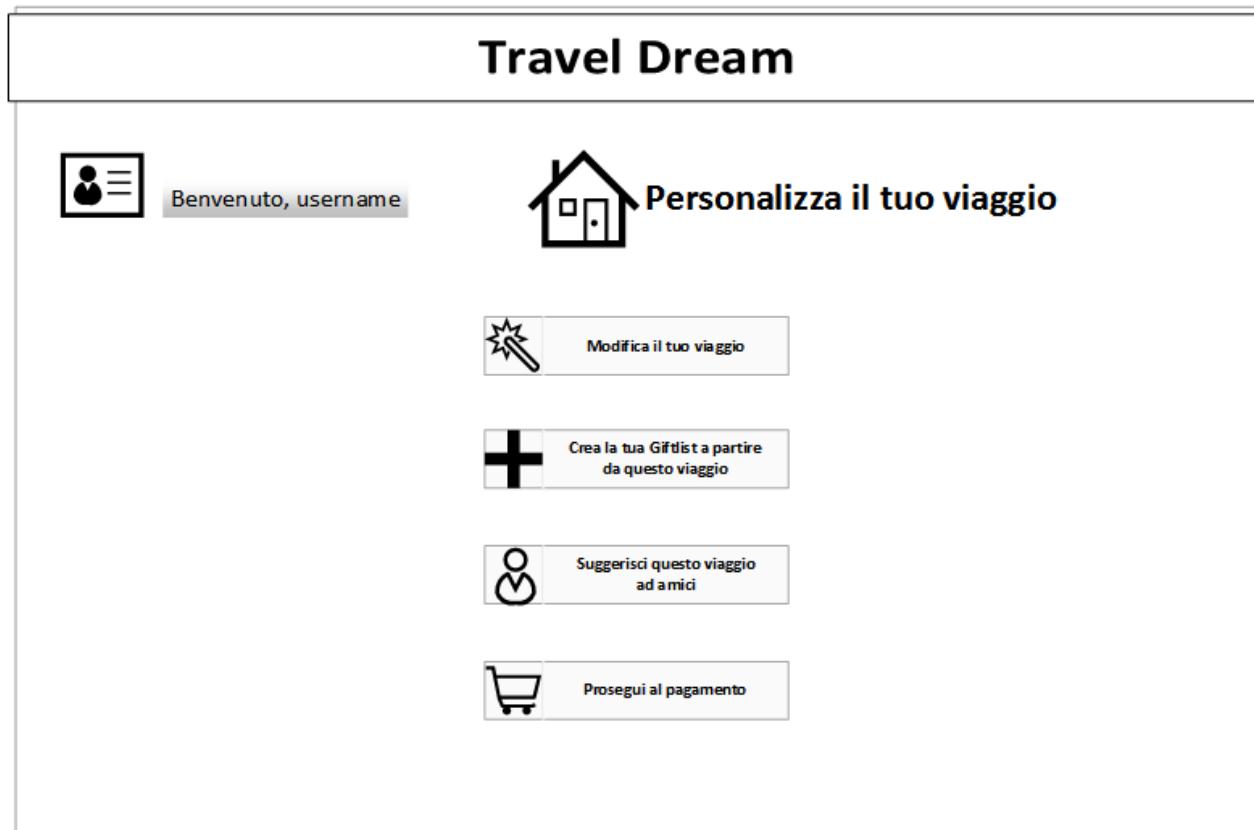
Dopo il login, l'impiegato accede alla sua pagina di gestione degli elementi base e dei pacchetti.



Invece per quanto riguarda l'utente registrato, dopo il login, gli verrà restituita la sua home page, con un grande form dove cercare viaggi e un menù a sinistra con le funzionalità a sua disposizione.



Abbiamo in fine, deciso di inserire anche un mock up per la pagina di personalizzazione del viaggio per evidenziare le possibilità che un utente possiede dopo la selezione del viaggio desiderato, e come egli possa creare una giflist, oppure suggerire il viaggio ad amici.



Ovviamente durante lo sviluppo l'interfaccia utente può subire variazioni maggiori, però lo scopo di questi mock up è dare un'idea di insieme per la futura implementazione.

## 3.2 FUNCTIONAL REQUIREMENTS

### 3.2.1 Attori

Come precisato in precedenza, gli attori del sistema sono:

1. Utente registrato
2. Utente non registrato
3. Impiegato

### 3.2.2 Scenari

Seguendo il documento della specifica del progetto, abbiamo analizzato ed esposto seguenti scenari:

#### 3.2.2.1 Un utente registrato crea una giftlist

Alessandro, utente registrato del sito Travel Dream, residente a Magenta (Mi), desidera creare una Giftlist. Per questo motivo, inizia ad inserire i dati personali di login per autenticarsi. Il suo login è avvenuto con successo e quindi Alessandro inizia a pensare al suo viaggio. Siccome ha da sempre sognato di visitare Istanbul, la sceglie come destinazione, impostando la partenza da Milano (tutti gli aeroporti). Aggiunge la data di partenza 1 dicembre 2013 e la data di ritorno 10 dicembre 2013. Guardando i pacchetti viaggio proposti dal sistema, Alessandro sceglie il pacchetto che contiene i seguenti prodotti: volo di andata il 1 di dicembre 2013 da Milano Malpensa alle 12:00 per Istanbul Sabiha Gokcen 16:00, 10 notti nell'albergo Sura Design, volo di ritorno il 10 di dicembre 2013 da Istanbul Sabiha Gokcen alle 13:00 per Milano Malpensa 15:00, e in più una escursione sullo stretto di Bosforo da 5 ore il

giorno 4 di dicembre. Alessandro conferma la sua scelta e sceglie di personalizzare il viaggio, decidendo di creare una Giftlist. A questo punto, Alessandro inserisce nella Giftlist tutti i prodotti scelti prima e clicca su conferma. Il sistema lo avvisa della creazione della Giftlist, che in totale costa 600 €.

### 3.2.2.2 Un utente registrato suggerisce il viaggio ad amici e poi paga il pacchetto

Bob, utente registrato di Travel Dream, residente a Milano, decide di andare a Londra a visitare alcuni parenti. Siccome non vuole andare da solo, ha intenzione di invitare anche sua sorella Diana e la sua miglior amica Elena a venire con lui. Una volta collegato al sito, inizia ad inserire le sue credenziali di accesso e si logga. Subito gli compare la pagina di ricerca viaggio ed inserisce i seguenti dati: partenza da Milano (tutti gli aeroporti) il 1 di febbraio 2014, ritorno da Londra (tutti gli aeroporti) il 5 di febbraio. Il sistema elabora i dati e gli ritorna la lista dei pacchetti che soddisfano la sua ricerca. A questo punto Bob sceglie il seguente pacchetto: partenza da Milano Bergamo il 1 di febbraio 2014 alle ore 8:00, destinazione Londra Stansted alle ore 9:00, ritorno da Londra Stansted il 5 di febbraio 2014 alle ore 22:00, destinazione Milano Bergamo alle ore 23:55. L'albergo dove andrà ad alloggiare: NH Hotels. Il pacchetto in totale, costa 430 €. Bob conferma la sua scelta, decide di personalizzare il pacchetto in modo da suggerire il viaggio ai suoi amici e quindi clicca sull'apposito tasto. A Bob viene chiesto nella prossima schermata di inserire gli indirizzi e-mail dei suoi amici per cui Bob inizia a scrivere le e-mail di Diana ed Elena. Finito l'inserimento, conferma e il sistema gli notifica che i suoi amici sono stati avvisati. A questo punto Bob decide di pagare il suo viaggio sperando che a Diana ed Elena piacerà il viaggio scelto e che potrà viaggiare in compagnia. Bob clicca sul tasto pagamento e inizia ad inserire i dati relativi al pagamento nella schermata propostagli. Bob clicca su conferma e il sistema gli restituisce una notifica di pagamento andato a buon fine. Contentissimo, Bob chiama i suoi parenti avvisando il suo arrivo.

### 3.2.2.3 Un utente registrato non trova nessun pacchetto soddisfacente, cambia i dati ed acquista un altro pacchetto

Francesco, utente registrato del sito Travel Dream, residente a Milano, desidera andare in un viaggio con la moglie Cristina a San Valentino, in una città romantica. Dopo essersi loggato al sito con i suoi credenziali, inizia a cercare un pacchetto inserendo i seguenti dati: partenza da Milano il 13 di febbraio 2014, destinazione Vienna con ritorno il 16 di febbraio 2014. Inviando i dati, riceve sfortunatamente la notifica di errore da parte del sistema: non abbiamo trovato nessun pacchetto soddisfacente, La preghiamo di cercare un'altra partenza /destinazione oppure di cambiare le date. Visto che Francesco non ha intenzione di cambiare le date, decide a questo punto di cambiare destinazione. Inizia da capo ed inserisce come destinazione Parigi, le date sempre le stesse. Inviando i dati, il sistema gli ritorna la lista con i pacchetti Milano Parigi. Francesco sceglie il seguente pacchetto: partenza da Milano Malpensa alle 10:35 il giorno 13 febbraio, arrivo a Parigi Charles de Gaulle alle 11:30; alloggio nell'albergo Hilton per 3 notti; ritorno da Parigi Charles de Gaulle alle 18:15 per Milano Malpensa, 19:10. Il totale è di 400 €. Francesco è contentissimo del viaggio scelto perché Cristina non ha mai visto Parigi. Francesco conferma la sua scelta e procede al pagamento. Nella schermata successiva inserisce i dati della carta di credito e il sistema gli ritorna la conferma di ricezione pagamento. Francesco terrà tutto segreto fino al 10 di febbraio, compleanno di Cristina, quando le dirà che stanno per andare a Parigi finalmente.

### 3.2.2.4 Un utente registrato visualizza un viaggio a cui è stato invitato e decide di acquistarlo

Carlo, utente registrato di Travel Dream, loggandosi al sito con i suoi credenziali e visualizzando il suo profilo, osserva la notifica di un viaggio a cui è stato invitato dal suo amico Mario, anche esse utente registrato di Travel Dream. Il viaggio è il seguente: viaggio a Barcellona da Milano con 3 notti nell'albergo Prince Hotels, dal 15 al 18 di dicembre 2013, andata Milano Bergamo(10:00) Barcellona El Prat(11:05), ritorno Barcellona El Prat(22:30) Milano Bergamo(23:35), e un'escursione che comprende ingresso a Sagrada Familia e due musei di grande interesse, per un totale di 200 €. Carlo e Mario sono vecchi amici di università e quindi Carlo si fida delle scelta fatta dal suo amico, gli piace l'idea di viaggiare insieme e quindi decide di acquistare anche lui il pacchetto. Clicca dunque sull'apposito tasto ed inizia a compilare il form di pagamento. Alla fine di tutto ciò, invia i dati e riceve la notifica di fine pagamento. Carlo chiama subito Mario per pianificare i loro giorni a Barcellona.

### 3.2.2.5 Un utente non registrato cerca un viaggio, deve effettuare registrazione, si registra, cerca viaggio e non trova nessun risultato

Luca, un utente non registrato di Travel Dream, vuole andare a Madrid con la sua ragazza Maria in un fine settimana. Consigliato da un amico, sceglie il sito di Travel Dream. Iniziando a navigare sul sito, capisce che non può cercare un viaggio senza essersi registrato e quindi clicca sull'apposito link per la registrazione. Inizia a compilare i form di registrazione con i dati richiesti e, una volta inviati, riceve la notifica dell'avvenuta registrazione. Il sistema lo porta alla sezione login e Luca inserisce le credenziali dichiarate prima. A questo punto, una volta loggato, inizia veramente la sua ricerca. Chiama la sua ragazza e decidono che il weekend dal 7 al 10 marzo 2014 sarebbe un'ottima scelta per tutti e due e quindi Luca inizia a cercare. Inserendo le date e la destinazione, il sistema purtroppo lo notifica del fatto che non esistono pacchetti soddisfacenti. Luca non vuole provare altre date o destinazioni, e per questo abbandona la sua ricerca.

### 3.2.2.6 Un utente registrato personalizza un viaggio individuato, con esito positivo, e lo paga

Riccardo, utente registrato del sito Travel Dream, residente a Busto Arsizio, vuole andare con i suoi 3 amici in un breve viaggio ad Atene. Hanno deciso che il periodo migliore per tutti e 4 sarà dal 1 al 4 di dicembre 2013. Loggandosi e cercando un viaggio, riesce a trovare uno che soddisfa a pieno il periodo stabilito, e cioè: andata 1 dicembre 2013 Milano - Atene, ritorno il 4 dello stesso mese, con 4 giorni nell'albergo Star Price. Però il pacchetto non comprende nessuna escursione nelle belle isole greche vicine. Perciò Riccardo inizia a personalizzare il suo viaggio. Nella schermata di personalizzazione sceglie di aggiungere escursione e quindi inserisce la città e il periodo. Il sistema gli ritorna 3 escursioni disponibili ad Atene nell'arco di tempo dal 1 al 4 di dicembre, tra cui anche un'escursione di 8 ore su 3 isole. Il prezzo è conveniente (1100 € per tutti 4) e Riccardo aggiunge l'escursione per 4 persone al pacchetto. Una volta finito, si dirige verso la pagina del pagamento e là inserisce i dati richiesti per pagare. Riccardo conferma i dati e riceve la notifica di pagamento avvenuto correttamente.

### 3.2.2.7 Un utente non registrato visualizza una GiftList a cui è stato invitato da un utente registrato, decide di pagarli i restanti prodotti, si registra e li acquista

Nicola è un utente non registrato di Travel Dream. Egli riceve una mail dal suo amico Daniele, utente registrato di Travel Dream, che gli chiede di pagare per lui alcuni componenti di un pacchetto viaggio. Nicola conosce bene Daniele e visto che sono vecchi amici, decide di pagarli i restanti prodotti. Quindi cliccando sul link viene portato alla registrazione, in quanto non registrato. Inserisce tutti i dati personali necessari e li invia. A questo punto si logga e accede alla sezione dove visualizza GiftList a cui è stato invitato. Lì seleziona il pacchetto di Daniele, si tratta di un viaggio da Roma a Lisbona dove manca da pagare l'albergo LisboaDesign, 3 notti, per un totale di 250 €, tutto il resto Daniele aveva già pagato prima. Nicola seleziona questo prodotto e prosegue con il pagamento, inserendo i dati relativi alla paga. Subito dopo l'invio di questi dati, riceverà la conferma del pagamento e annuncia Daniele che la sua GiftList è completa.

### 3.2.2.8 Un utente registrato visualizza la cronologia dei suoi viaggi dopo essersi loggato

Andrea, utente registrato di Travel Dream, vuole raccontare ai suoi genitori di tutti i viaggi fatti in questi due anni da quando vive a Milano. Sapendo che ha sempre viaggiato con Travel Dream, si logga con i suoi credenziali al sito, e sul suo profilo personale trova il link della cronologia dei suoi viaggi, cronologia che comprende un totale di 5 viaggi fatti a Zurigo, Monaco, Berlino, Budapest e Zagabria.

### 3.2.2.9 Un utente non registrato visualizza una GiftList a cui è stato invitato da un utente registrato e decide di non pagarli niente

Jacopo, utente non registrato di Travel Dream, riceve una mail con un riassunto di una GiftList e i suoi componenti da pagare, dal suo amico Michele, utente registrato di Travel Dream. Si tratta di un viaggio ad Amsterdam a Natale dove rimane da pagare il volo andata e ritorno da Milano, per un valore di 150 €. Jacopo non vuole pagare niente a Michele, cancella la mail subito. La GiftList andrà in timeout.

10. Un utente non registrato visualizza un viaggio a cui è stato invitato da un utente registrato e decide di non aderire al viaggio

Gabriele, utente non registrato di Travel Dream, riceve una mail con un breve riassunto di un viaggio da parte del suo amico Manuel, utente registrato di Travel Dream. Si tratta di un viaggio a Dublino, 3 giorni dal 1 al 3 di marzo 2014 con volo da Milano Malpensa e ritorno nello stesso aeroporto. Il viaggio comprende anche un'escursione nei numerosi castelli medievali nelle vicinanze, per un totale di 450 €, comprese 2 notti all'albergo Dublin Hotel. Però Gabriele rifiuta in quanto ha già preso un impegno in quel periodo; Manuel andrà comunque a Dublino.

### 3.2.2.11 Un impiegato, dopo il login, crea un elemento base e lo inserisce nel sistema

Stefano, impiegato di Travel Dream, accede alla pagina di gestione degli elementi base, inserendo i suoi credenziali nella schermata di login. Una volta dentro, accede alla sezione per inserire un nuovo elemento e inizia a compilare il form che gli è stato ritornato dal sistema. Si tratta di un'escursione alla Città del Vaticano il giorno 15 di dicembre 2013, per i viaggi fatti a Roma in quel periodo. Il valore dell'escursione è di 55 €. Stefano invia i dati inseriti al sistema che gli ritorna la notifica positiva della creazione.

#### 12. Un impiegato, dopo il login, modifica un elemento base nel sistema

Dan, impiegato di Travel Dream, accede alla pagina di gestione degli elementi per modificare un elemento, dopo aver effettuato con successo il login. Dall'elenco degli elementi, Dan sceglie il volo Roma Fiumicino (17:00) - Parigi Charles de Gaulle (19:10) del giorno 16 dicembre 2013. Nel form di modifica, Dan aggiorna il vecchio prezzo di 90 € al nuovo prezzo di 95 €, dopodiché conferma la modifica dei dati. Il sistema lo avvisa, dopo la conferma, che le modifiche portate all'elemento siano valide e per ciò, l'elemento è stato modificato con successo.

#### 3.2.2.12 Un impiegato sbaglia il login, riprova ed entra nel sistema, eliminando un elemento base

Giorgio, impiegato di Travel Dream, vuole accedere alla pagina di gestione degli elementi per eliminare un elemento base. Però nella schermata di login inserisce, per errore, dei dati sbagliati e quindi il sistema non lo logga e lo fa ritornare alla schermata di login. Alla seconda volta Giorgio non fa nessun errore e si logga con successo. Quindi dall'elenco degli elementi base, sceglie l'albergo Sole di Istanbul e opta per la sua eliminazione, visto che l'albergo è stato venduto due giorni fa. Una volta confermata la scelta, Giorgio riceve la notifica dell'eliminazione dell'albergo.

#### 3.2.2.13 Un impiegato, dopo il login, crea un pacchetto predefinito

Claudio, impiegato di Travel Dream, deve creare un pacchetto predefinito ed inserirlo nel sistema. Nella schermata di login, inserisce i suoi credenziali ed entra nel sistema, dove sceglie di creare un pacchetto. Il sistema gli ritorna un form dove Claudio dovrà scegliere il luogo o la città del viaggio. Claudio inserisce Stoccolma e clicca per proseguire. Il sistema gli ritorna tutti gli alberghi della città, tutti i trasporti con destinazione Stoccolma, e tutte le escursioni a Stoccolma. A questo punto Claudio crea il pacchetto: volo andata Milano Malpensa (9:00) - Stoccolma Bromma (10:35) il 3 di marzo 2014, volo ritorno Stoccolma Bromma (14:50) - Milano Malpensa (15:50) il 9 di marzo 2014, albergo Stockholm City Hotel, escursione nei palazzi della monarchia svedese il 7 di marzo 2014, per un totale di 500 €. Claudio conferma gli inserimenti fatti e subito riceve la notifica da parte del sistema della creazione di un nuovo pacchetto.

#### 3.2.2.14 Un impiegato sbaglia il login, riprova ed entra nel sistema con l'intenzione di creare un elemento base ma sbaglia ad inserire alcuni dati, il sistema lo avvisa, l'impiegato riprova la seconda volta con successo

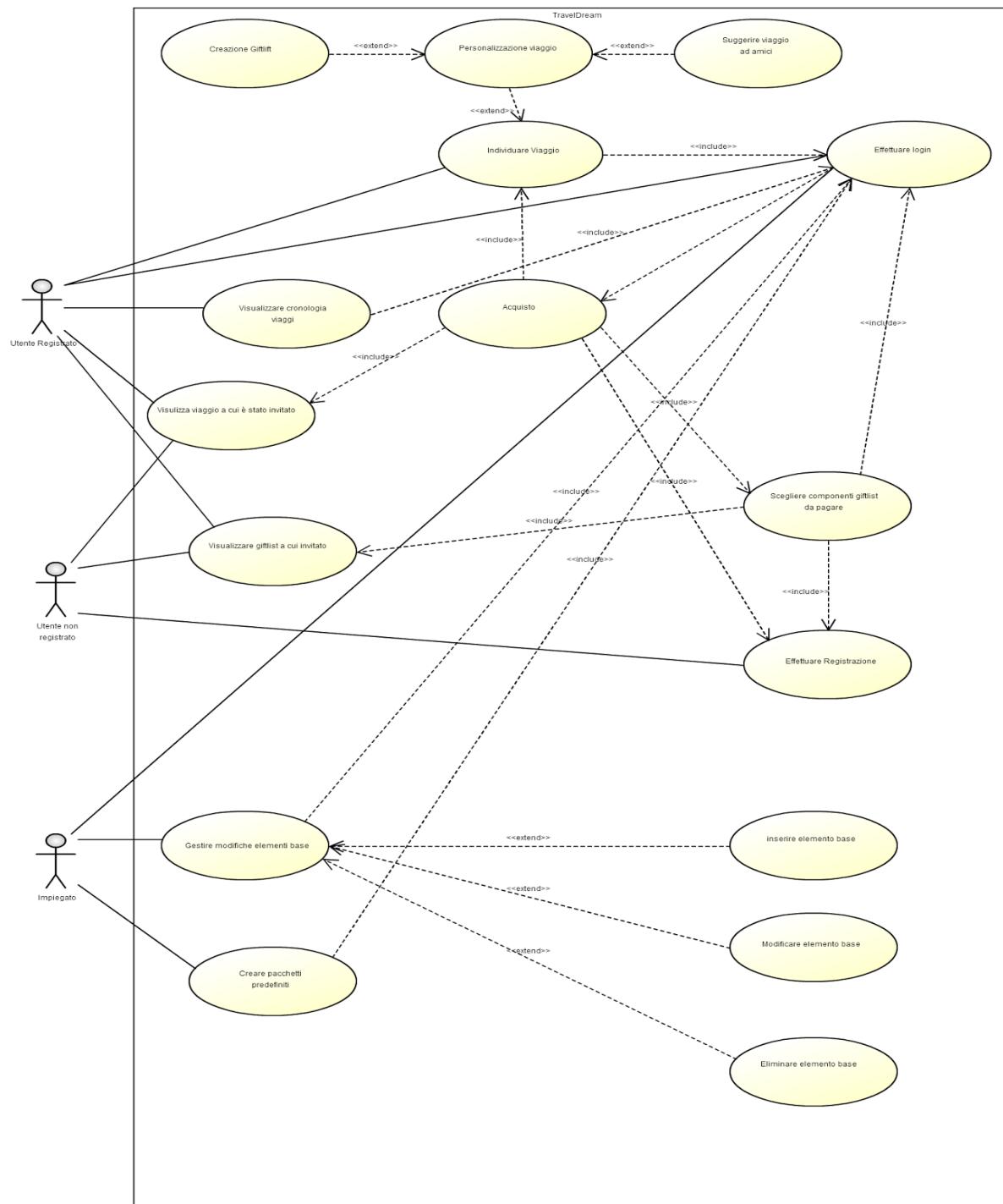
Mario, impiegato di Travel Dream, deve inserire nel sistema un nuovo volo da Milano ad Amsterdam. Per fare ciò, accede alla schermata di login ed inserisce i suoi credenziali, ma sbaglia ad inserirli per disattenzione. Il sistema lo fa

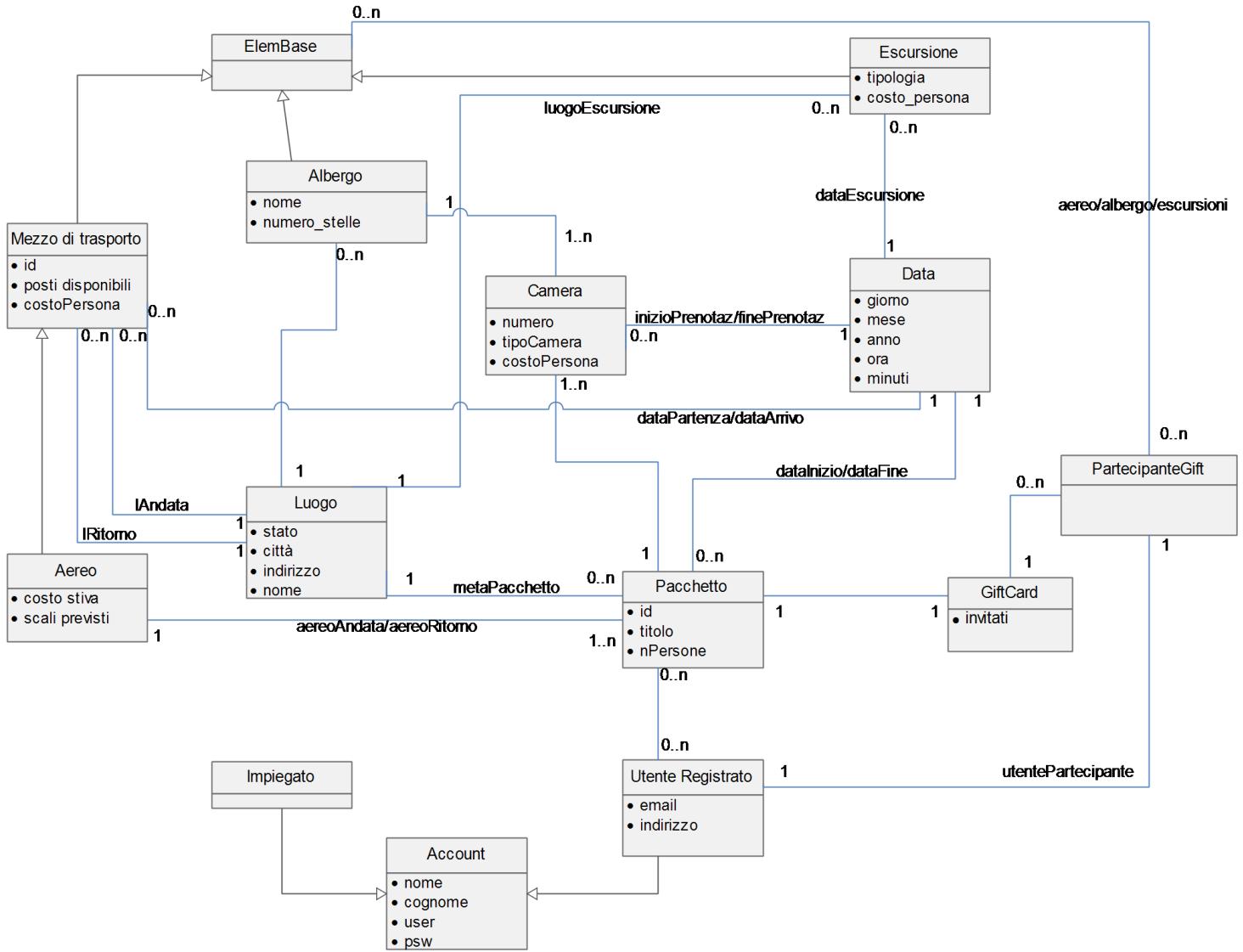
riprovare e questa volta Mario non sbaglia più. Una volta loggato, accede alla sezione per creare un elemento base e il sistema gli ritorna il form di creazione nuovo elemento. Nel form, Mario inserisce i seguenti dati: volo da Milano Bergamo (8:00) per Amsterdam (9:05), il giorno 20 di dicembre 2013. Però Mario dimentica di inserire il prezzo di questo volo e clicca sull'invio dei dati. Il sistema rileva l'errore e Mario si vede costretto a ripetere la procedura. Questa volta si ricorda di inserire il prezzo (50 €) e confermando i dati, riceve la notifica da parte del sistema della creazione del nuovo volo.

### 3.2.3 Use case e modellazione UML

A partire dagli scenari elencati sopra, sono stati definiti alcuni casi d'uso in modo da definire le funzioni implementate nel sistema, e di seguito, il diagramma UML delle classi, che identifica le entità modelizzate nel sistema:

## USE CASE DIAGRAM:

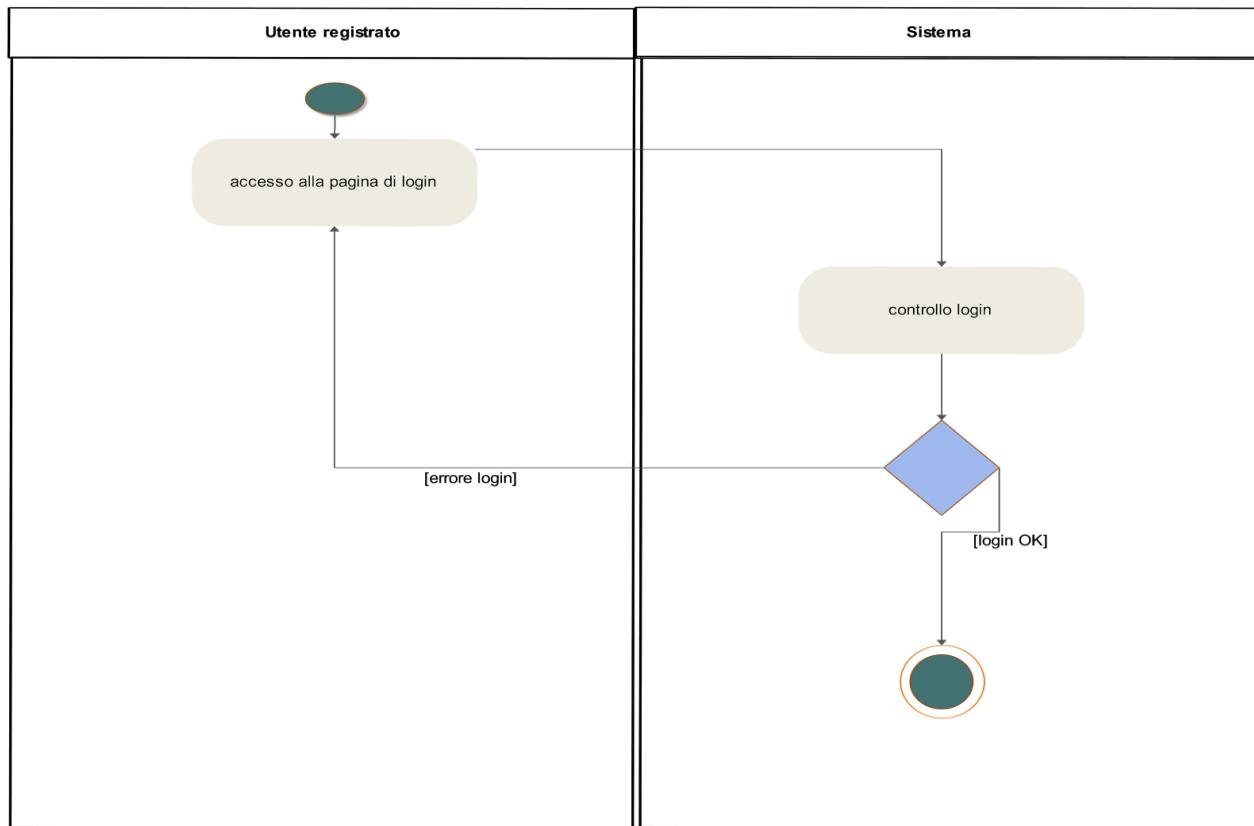


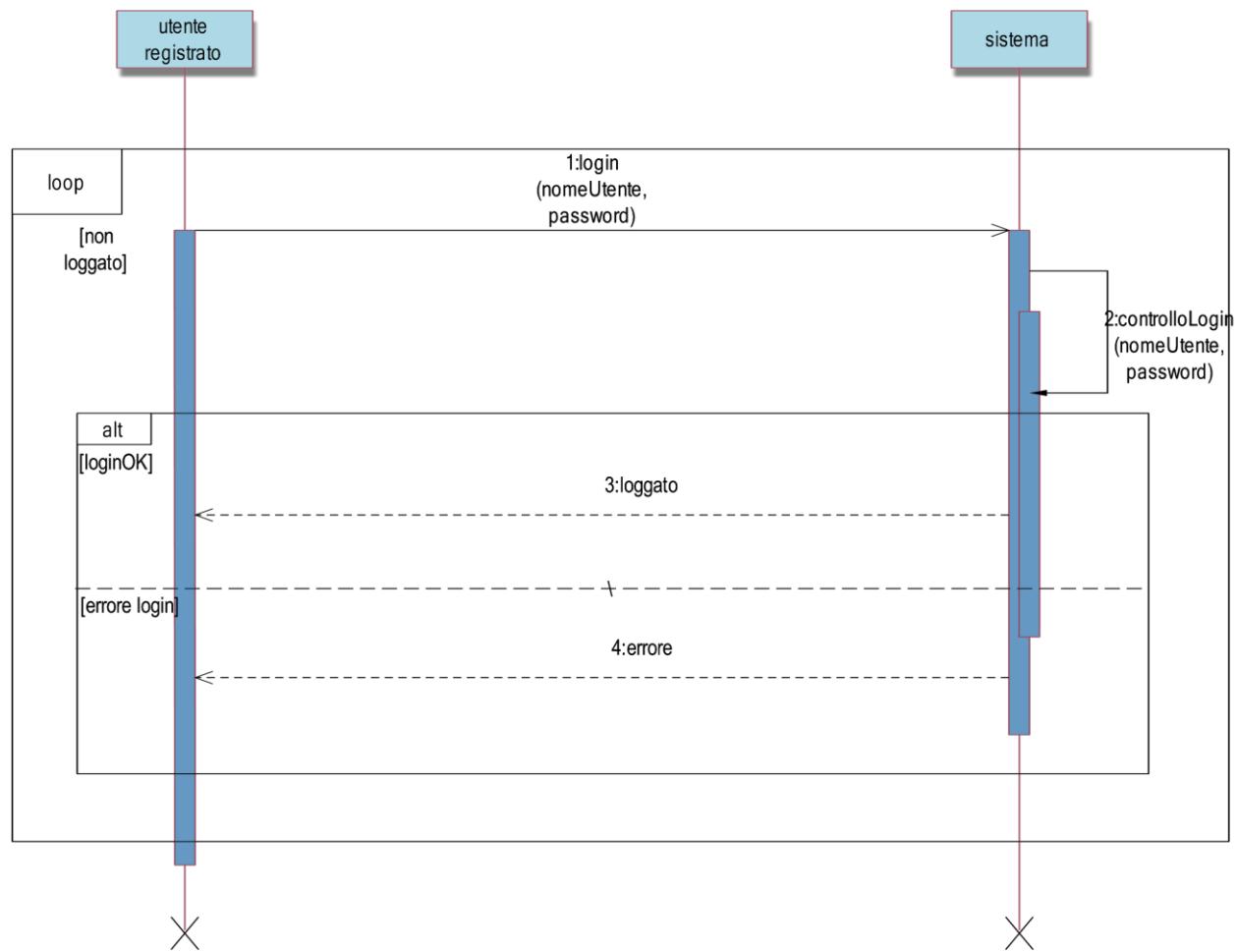
**Class diagram:**

### 3.2.3.1 Effettuare login

Seppur semplice, questo caso d'uso è fondamentale per l'utente registrato e per l'impiegato. Per questo abbiamo deciso di trattarlo separatamente e non inserirlo in altri casi di uso, e proseguendo con gli altri casi che caratterizzano azioni effettuate dall'utente registrato o dall'impiegato, di considerare il login avvenuto con successo e quindi non inserirlo più nei diagrammi. Il caso del login da parte dell'impiegato è banalmente uguale all'utente registrato.

<b>Titolo caso d'uso</b>	Utente registrato effettua login
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	-
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente accede alla pagina di login e inserisce i dati richiesti nel form login</li> <li>2. il sistema verifica i dati e manda notifica dell'esito all'utente</li> </ol>
<b>Condizione di uscita</b>	login avvenuto con successo
<b>Eccezioni</b>	dati inseriti sbagliati, l'utente viene rimandato al login



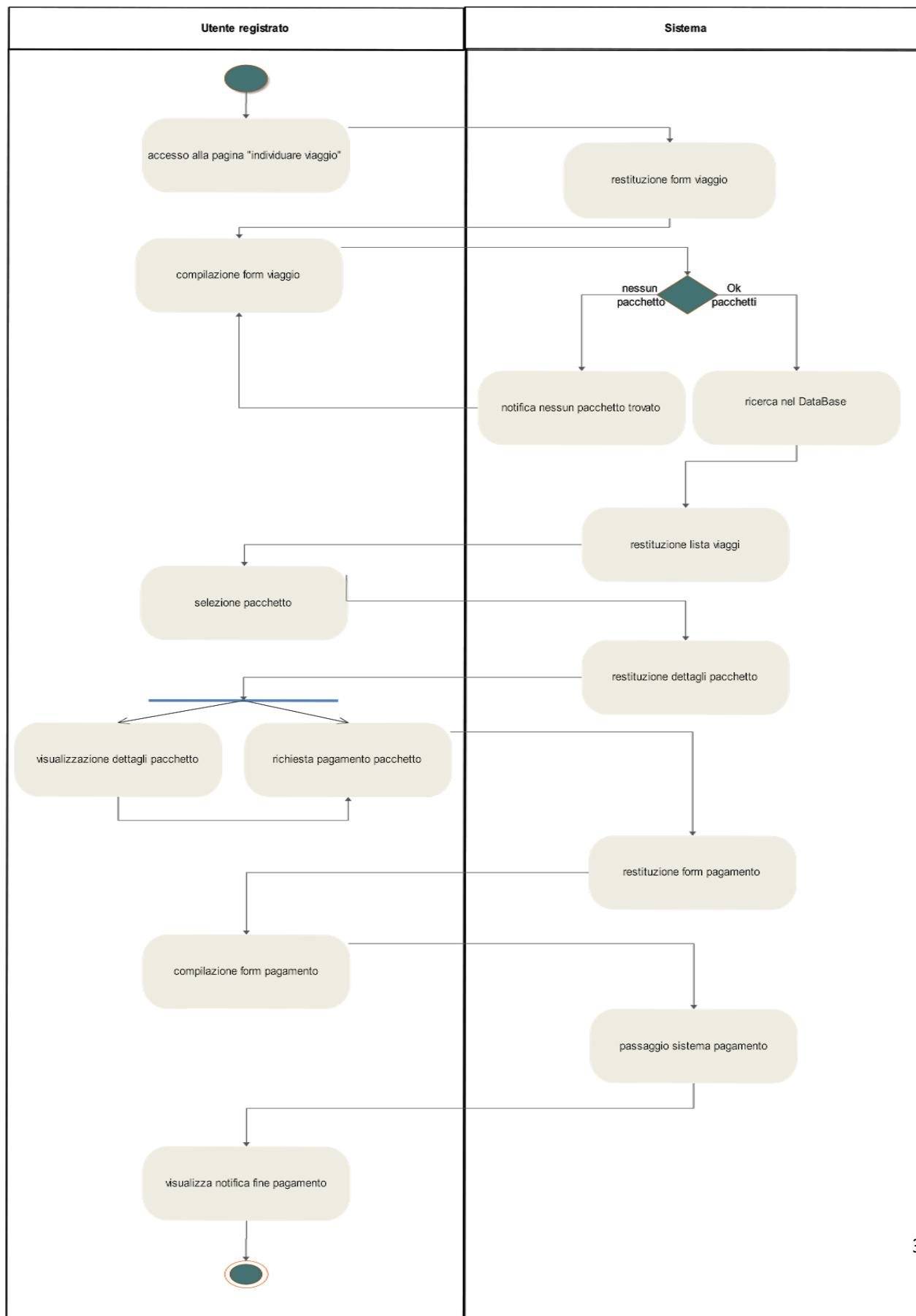


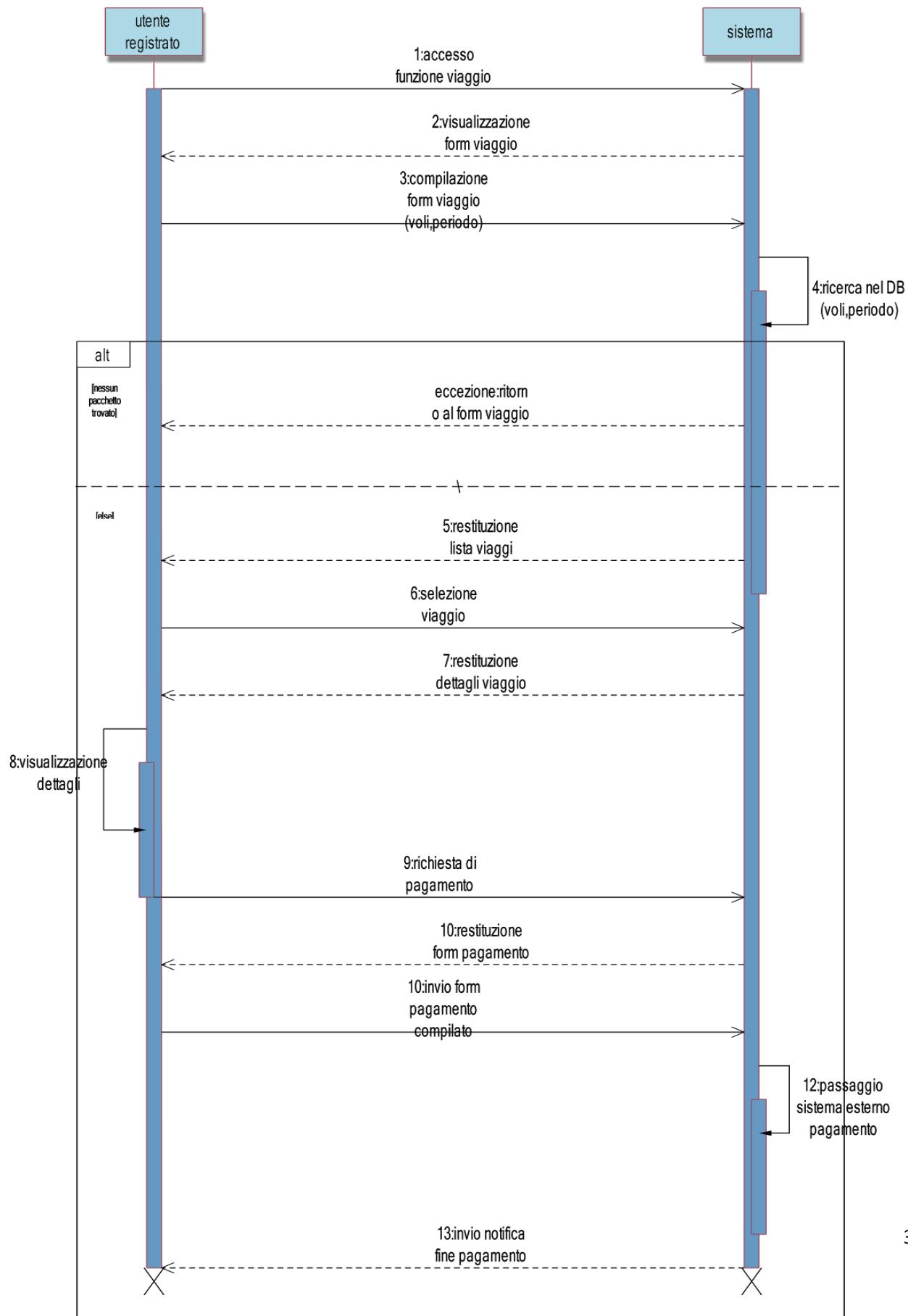
### 3.2.3.2 Individuare viaggio ed acquisto

Il seguente caso d'uso comprende anche l'acquisto da parte dell'utente registrato.

<b>Titolo caso d'uso</b>	l'utente registrato individua un viaggio e lo paga direttamente
<b>Attori coinvolti</b>	l'utente registrato
<b>Condizione d'ingresso</b>	l'utente registrato deve essere loggato
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente registrato e loggato accede alla pagina "individuare il viaggio" e chiede di cercare un viaggio</li> <li>2. il sistema restituisce il form destinazione-data</li> <li>3. l'utente inserisce le informazioni necessarie (destinazione e periodo)</li> <li>4. il sistema cerca nel DataBase i pacchetti in funzione delle informazioni inserite</li> <li>5. il sistema restituisce all'utente una lista che contiene i pacchetti trovati</li> <li>6. l'utente visualizza la lista e seleziona un pacchetto</li> <li>7. il sistema gli restituisce la pagina con i dettagli del pacchetto (che include la possibilità di personalizzarlo) oppure proseguire con il pagamento</li> </ol>

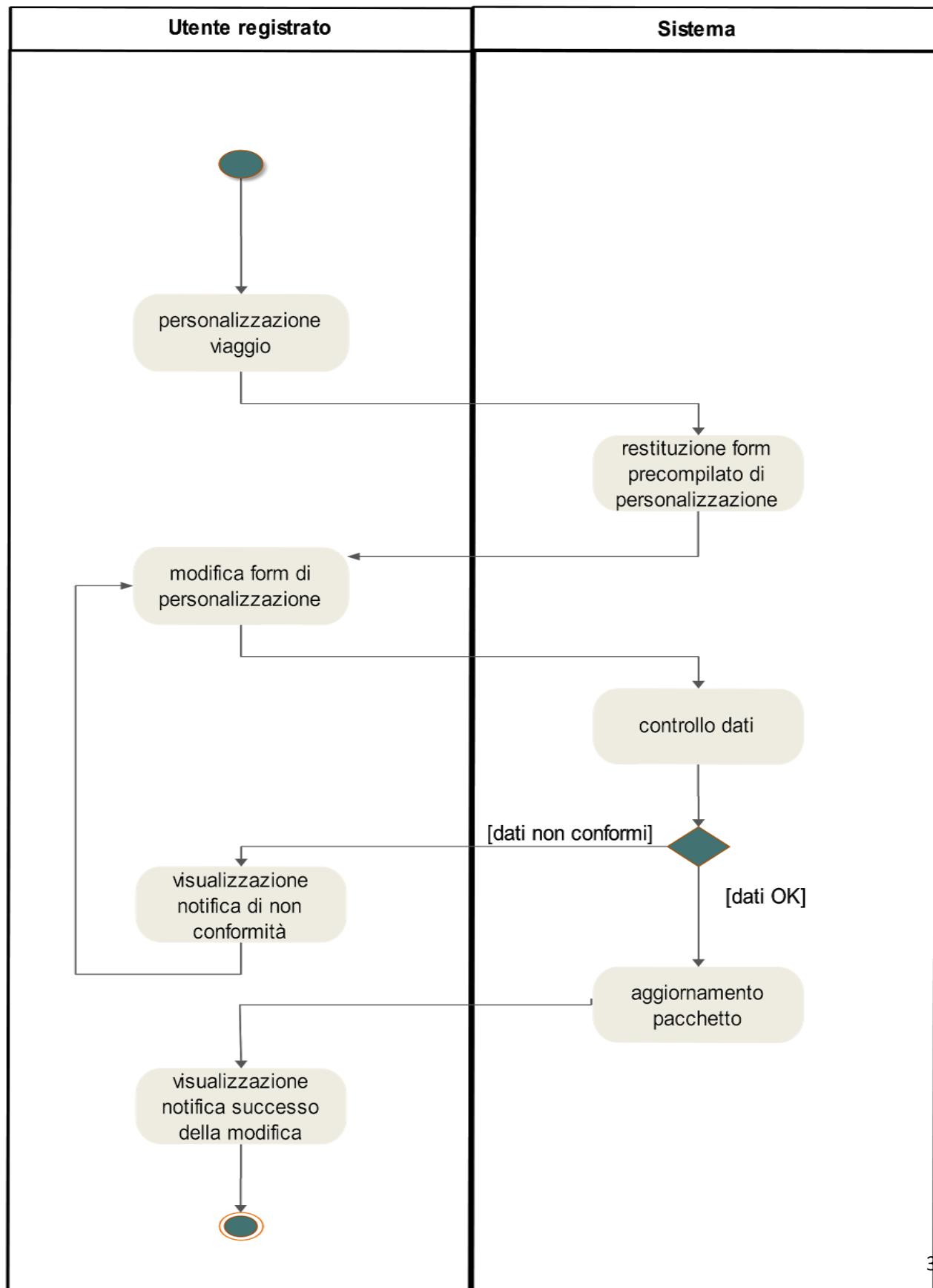
	<ol style="list-style-type: none"><li>8. l'utente richiede di proseguire con il pagamento normalmente</li><li>9. il sistema gli restituisce un form da completare con i dati relativi al pagamento</li><li>10. l'utente inserisce i dati del pagamento e invia il form</li><li>11. il sistema passa i dati al sistema esterno per il pagamento</li></ol>
<b>Condizione di uscita</b>	invio messaggio pagamento andato a buon fine
<b>Eccezioni</b>	non esiste un pacchetto che rispetti le richieste, il sistema ritorna un messaggio di errore (roll back compilazione form viaggio)

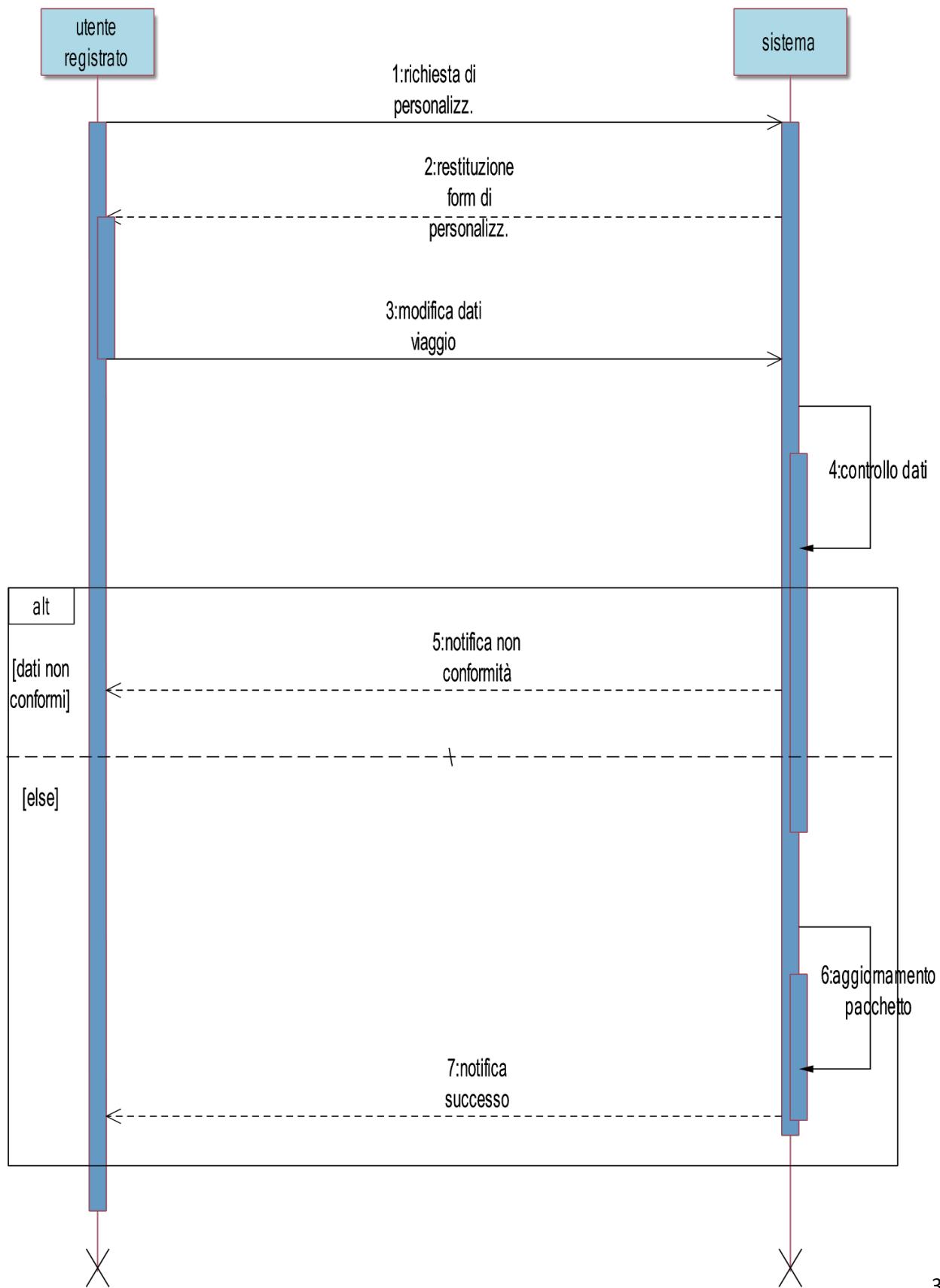




### 3.2.3.3 Personalizzazione viaggio

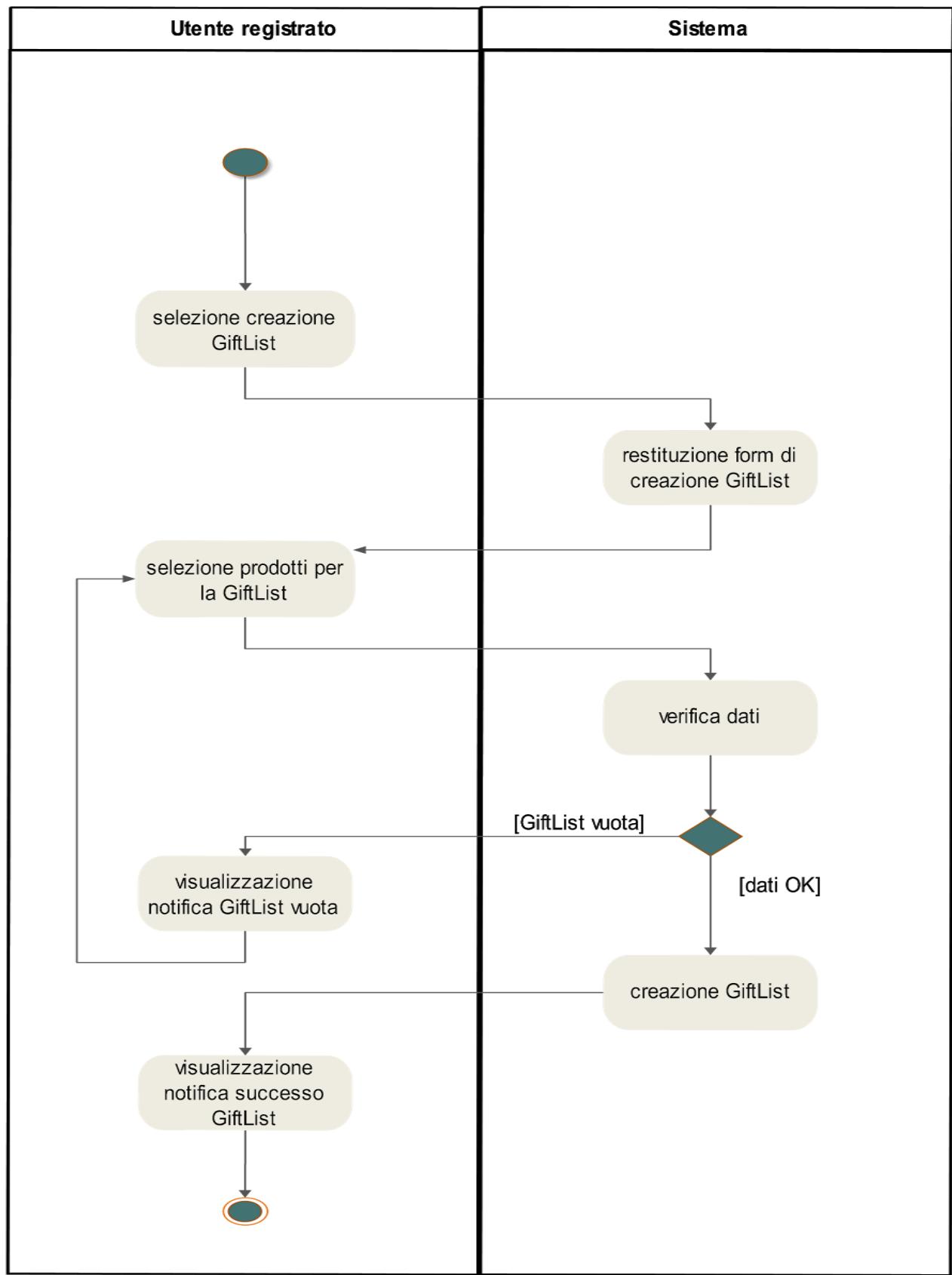
<b>Titolo caso d'uso</b>	L'utente registrato decide di personalizzare il viaggio scelto
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	L'utente deve aver individuato il viaggio
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente sceglie di personalizzare il viaggio, cliccando sull'apposito tasto</li> <li>2. il sistema gli restituisce il form di personalizzazione precompilato con i dati del pacchetto già esistenti</li> <li>3. l'utente inserisce le sue modifiche e finalmente le conferma</li> <li>4. il sistema verifica la conformità dei dati modificati dall'utente</li> <li>5. il sistema aggiorna il pacchetto viaggio</li> </ol>
<b>Condizione di uscita</b>	Il sistema notifica il successo dell'operazione all'utente
<b>Eccezioni</b>	L'utente inserisce dati non conformi (non consistenti, non validi, non esistenti); il sistema notifica che la procedura deve essere ripetuta

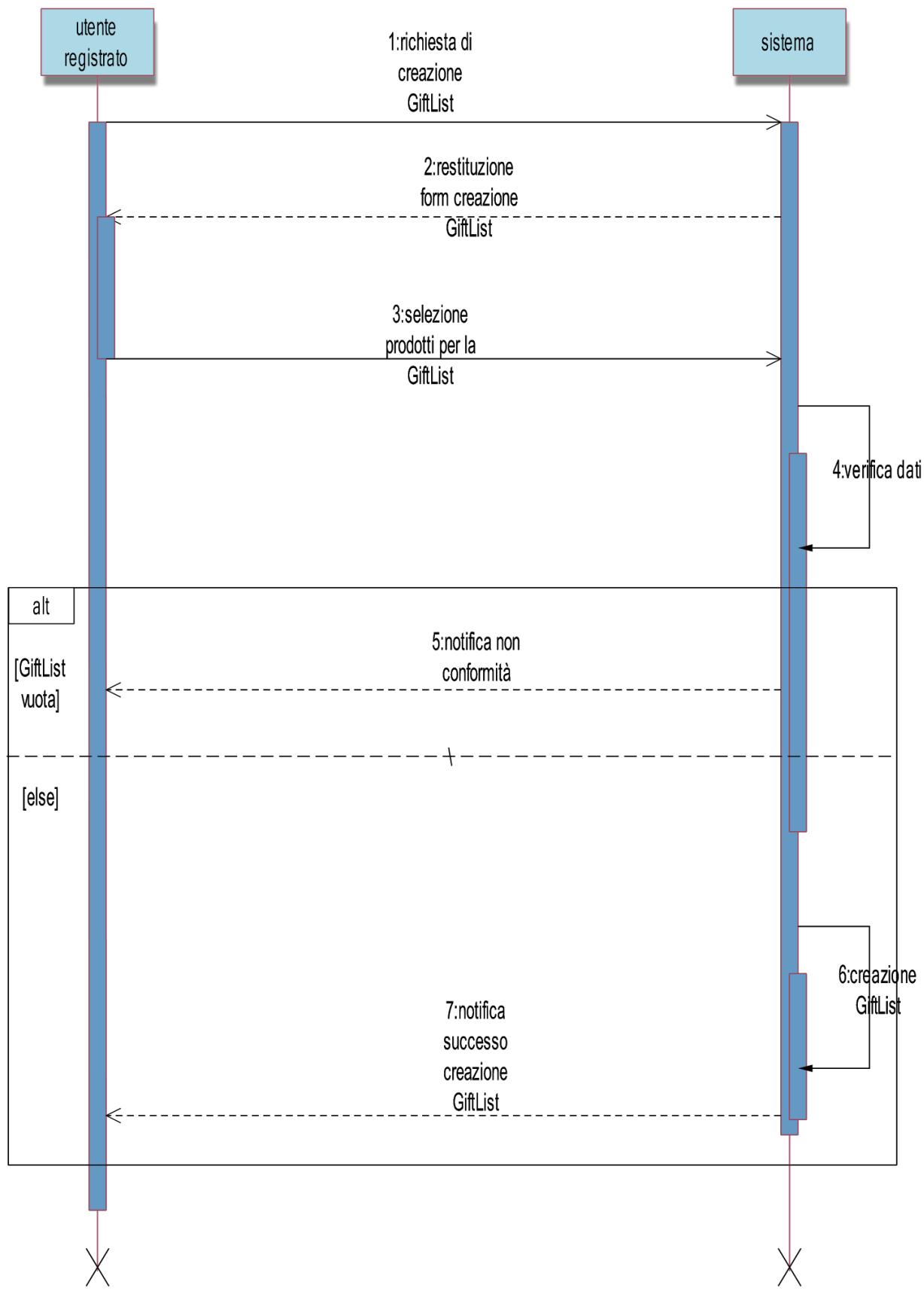




### 3.2.3.4 Creazione giftlist

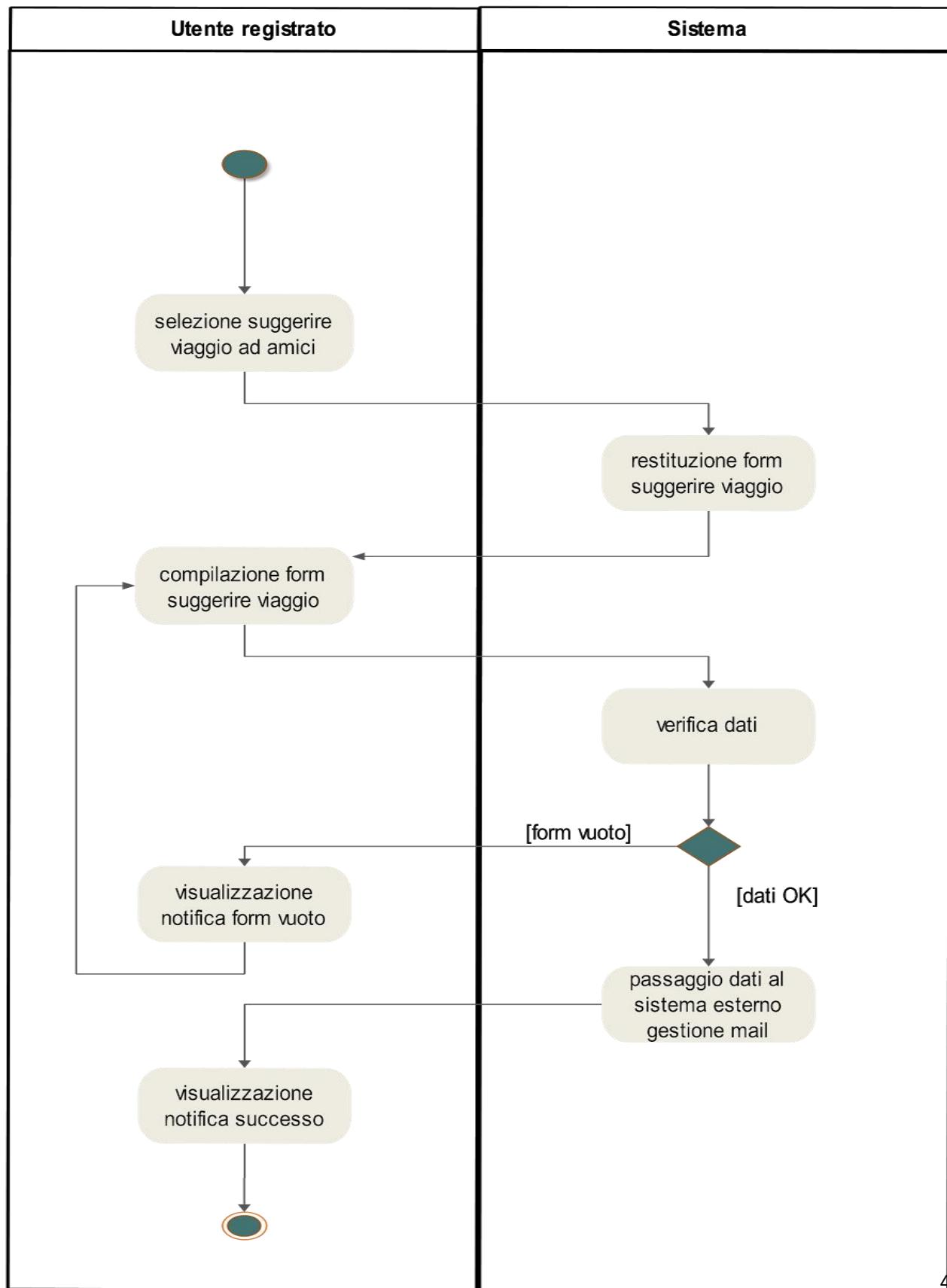
<b>Titolo caso d'uso</b>	L'utente registrato crea una GiftList
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	L'utente sta personalizzando il viaggio
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente sceglie di creare una GiftList, cliccando sull'apposito tasto</li> <li>2. il sistema gli restituisce il form di creazione GiftList</li> <li>3. l'utente decide i componenti da inserire e finalmente conferma</li> <li>4. il sistema verifica il form</li> <li>5. il sistema crea la GiftList</li> </ol>
<b>Condizione di uscita</b>	Il sistema notifica l'utente della creazione della giftlist avvenuta con successo
<b>Eccezioni</b>	L'utente conferma il form senza aver scelto nessun prodotto da inserire nella GiftList; il sistema manda notifica all'utente di scegliere almeno un prodotto

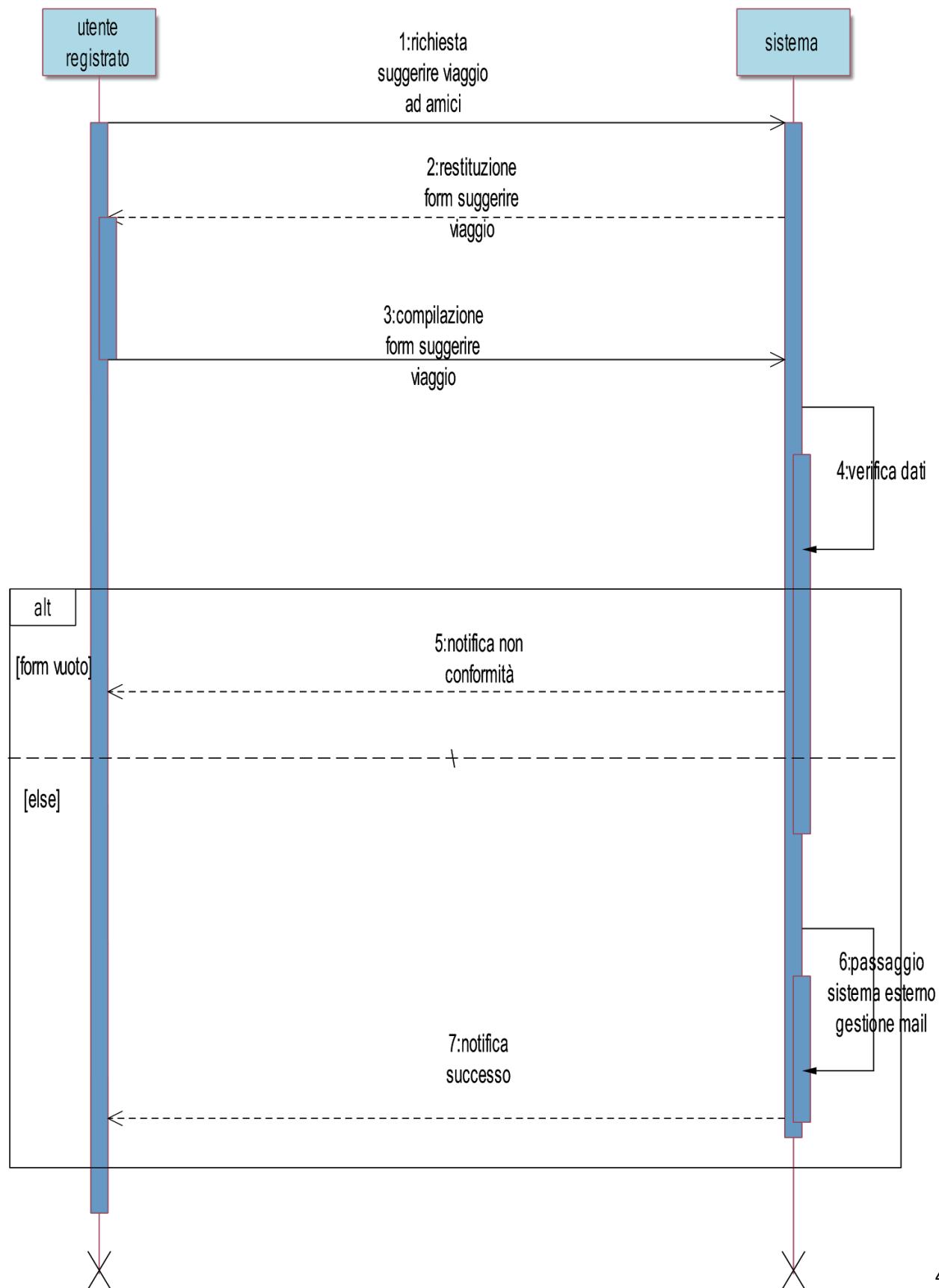




### 3.2.3.5 Suggerire viaggio amici

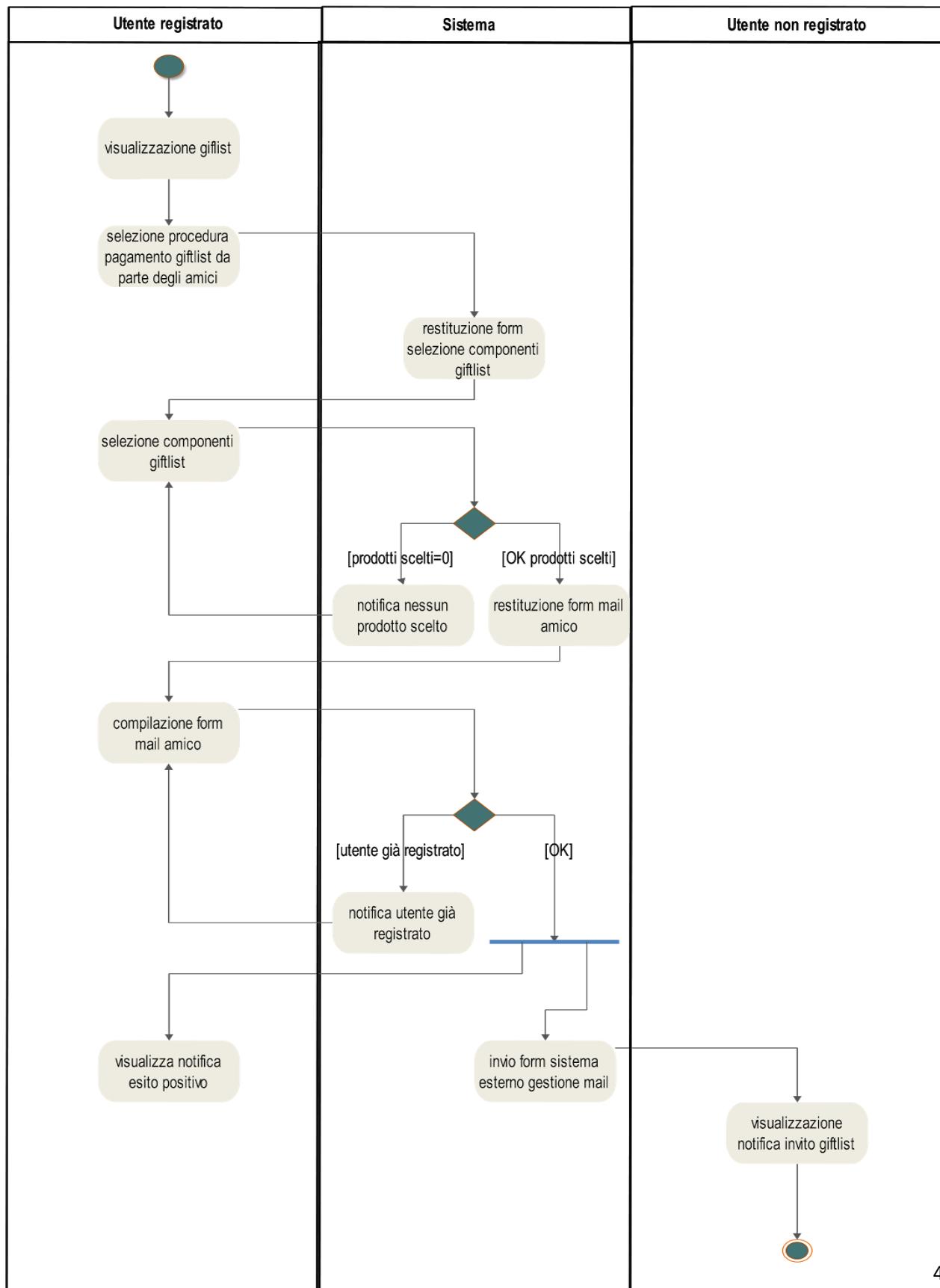
<b>Titolo caso d'uso</b>	L'utente suggerisce il viaggio scelto a degli amici, via mail
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	L'utente sta personalizzando il viaggio
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente sceglie di suggerire il viaggio ad amici, cliccando sull'apposito tasto</li> <li>2. il sistema gli restituisce il form da riempire con mail di amici</li> <li>3. l'utente compila il form e finalmente lo conferma inviandolo al sistema</li> <li>4. il sistema verifica il form</li> <li>5. il sistema passa le mail inserite al sistema esterno gestione mail</li> </ol>
<b>Condizione di uscita</b>	Il sistema notifica l'utente del successo dell'operazione
<b>Eccezioni</b>	L'utente conferma il form senza aver introdotto nessuna mail; il sistema manda notifica all'utente di inserire almeno una mail

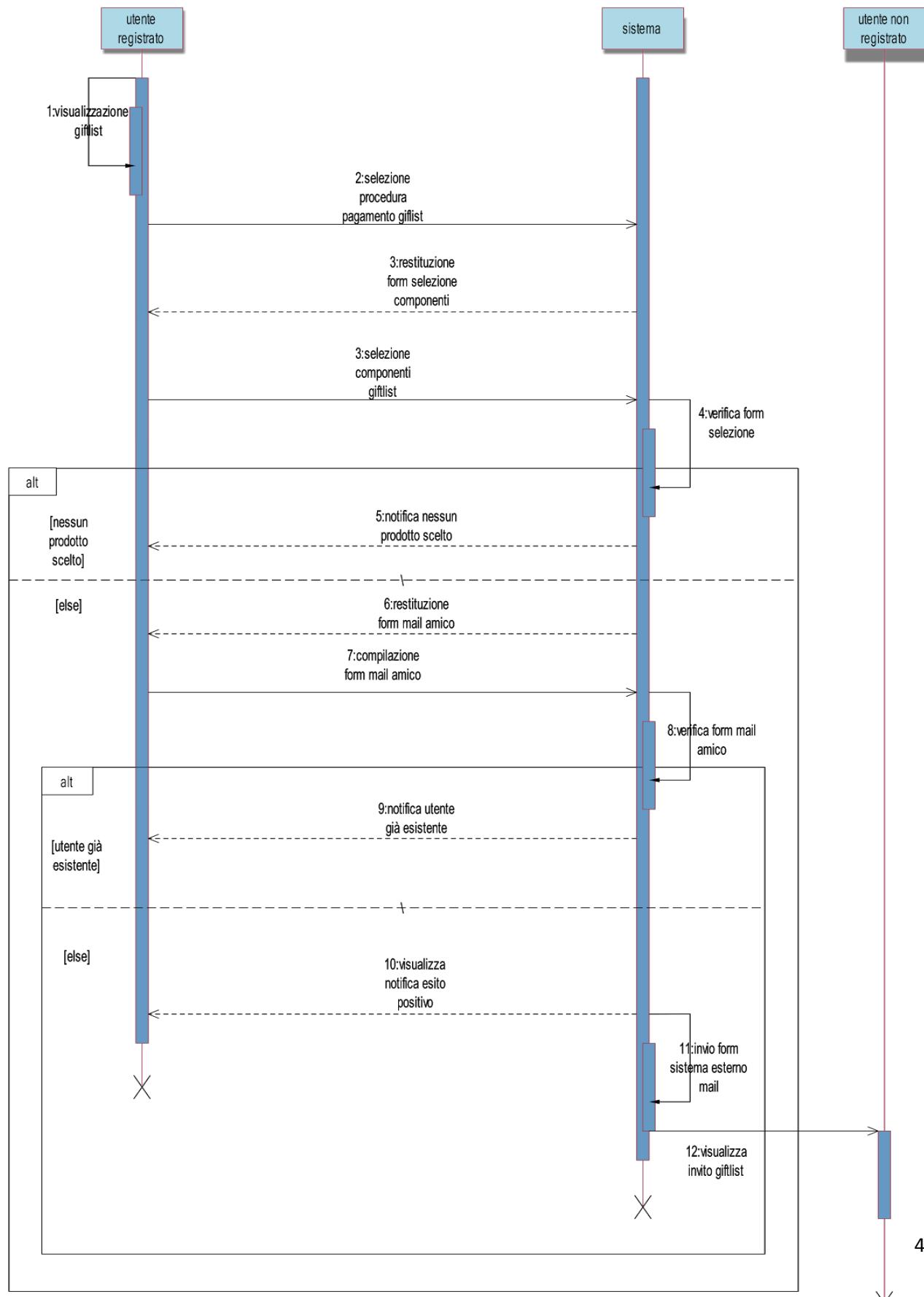




### 3.2.3.6 Selezione prodotti giftlist per pagamento amico

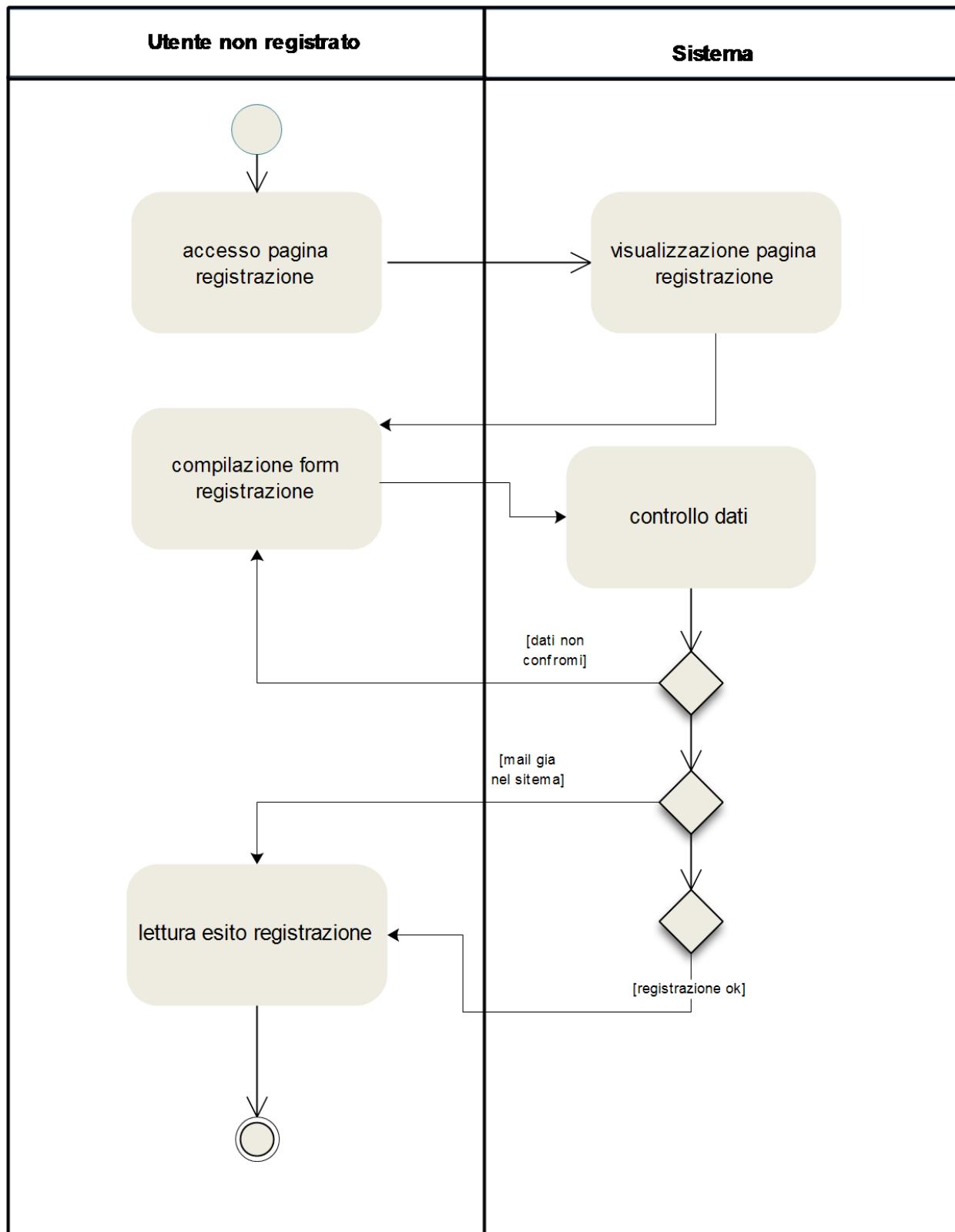
<b>Titolo caso d'uso</b>	L'utente registrato seleziona prodotti della giftlist per essere pagati da un amico, l'utente non registrato li visualizza
<b>Attori coinvolti</b>	Utente registrato, utente non registrato
<b>Condizione d'ingresso</b>	L'utente registrato deve aver creato una giftlist
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente registrato visualizza la giftlist creata in precedenza</li> <li>2. l'utente clicca su selezione prodotti per la giftlist, per pagamento agli amici</li> <li>3. il sistema gli restituisce il form di selezionare i prodotti</li> <li>4. l'utente sceglie i prodotti e poi conferma la selezione</li> <li>5. il sistema conferma la selezione e ritorna all'utente il form mail amico</li> <li>6. l'utente compila il form mail amico e lo conferma</li> <li>7. il sistema verifica il form e notifica all'utente l'esito</li> <li>8. l'utente registrato visualizza l'esito</li> <li>9. il sistema manda richiesta al sistema esterno per invio mail all'utente non registrato</li> </ol>
<b>Condizione di uscita</b>	I'utente non registrato visualizza la giftlist a cui è stato invitato
<b>Eccezioni</b>	<ol style="list-style-type: none"> <li>1. l'utente registrato non seleziona nessun prodotto; il sistema notifica l'errore e lo riporta alla giftlist</li> <li>2. l'utente registrato inserisce mail di un utente già registrato; il sistema notifica l'errore e lo riporta al form mail amico</li> </ol>

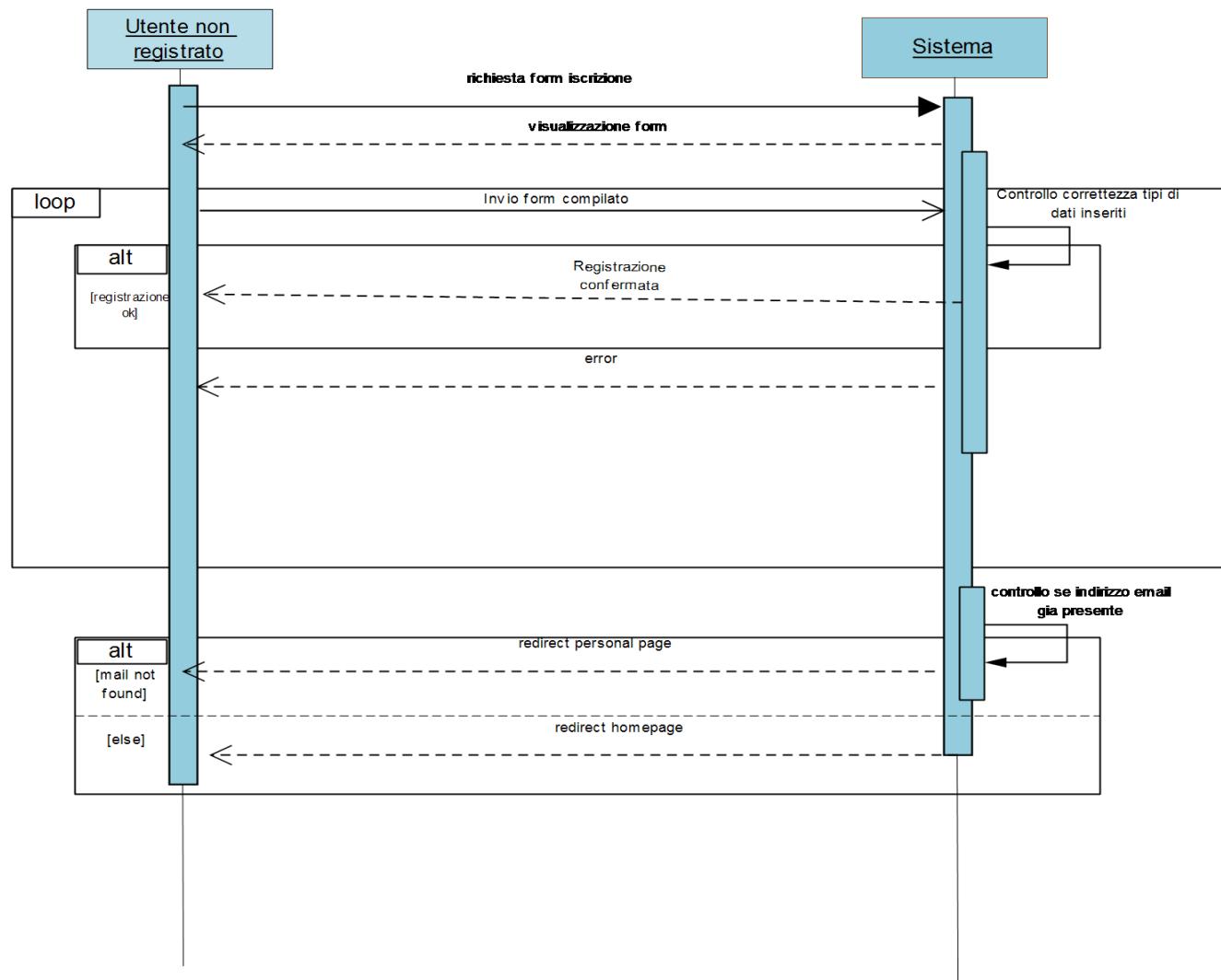




### 3.2.3.7 Effettuare registrazione

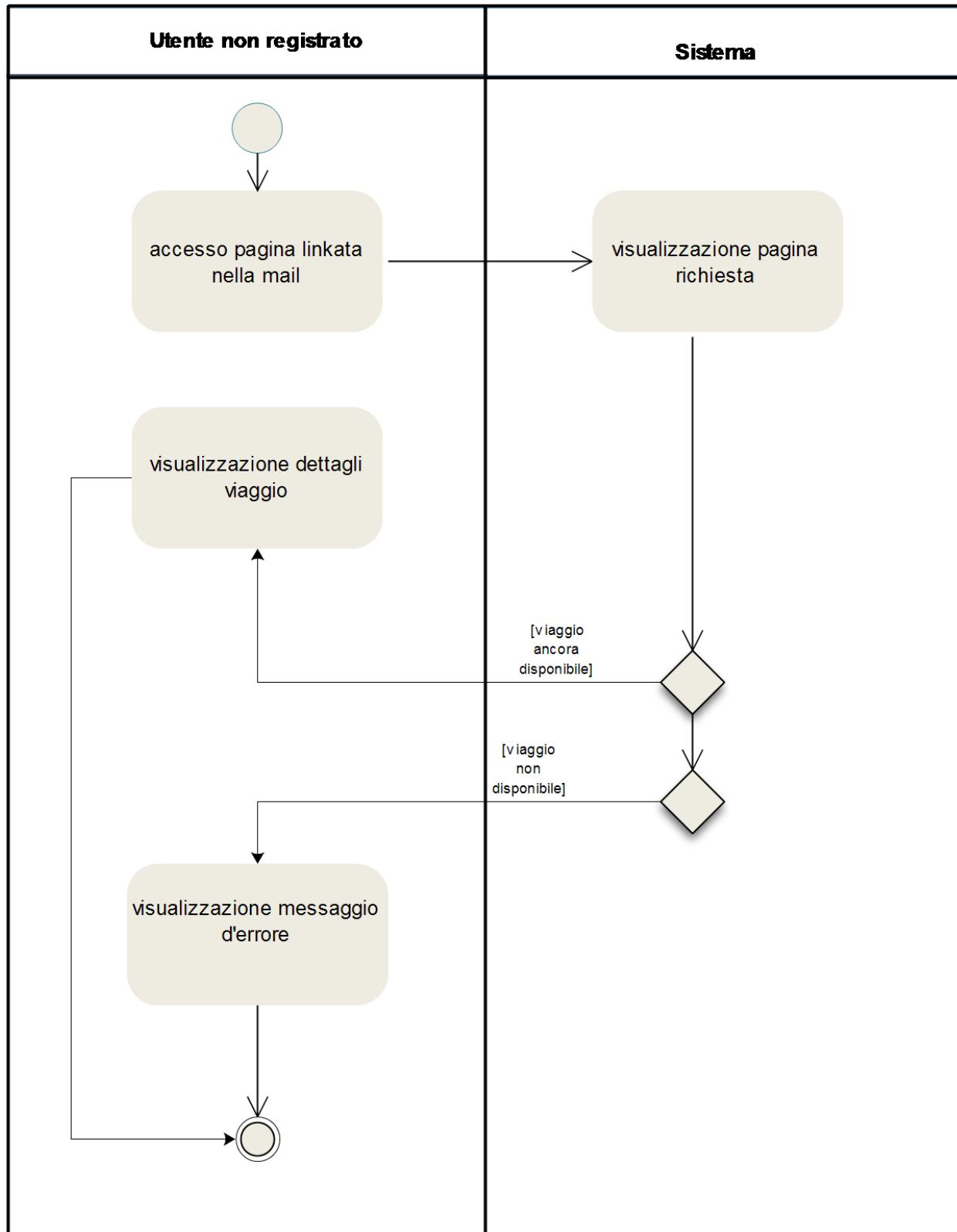
<b>Titolo caso d'uso</b>	Registrazione utente non registrato
<b>Attori coinvolti</b>	Utente non registrato
<b>Condizione d'ingresso</b>	Un utente non registrato deve effettuare la registrazione al sito
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente si dirige alla pagina di registrazione del sito</li> <li>2. il sistema restituisce il form da compilare</li> <li>3. L'utente inserisce i dati richiesti per la registrazione nell'apposito form</li> <li>4. L'utente visualizza il risultato della sua registrazione</li> </ol>
<b>Condizione di uscita</b>	L'utente viene reindirizzato alla propria pagina profilo.
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>● La registrazione non va a buon fine causa dati inseriti non conformi, il sistema segnala l'errore e ne chiede la reimmissione all'utente.</li> <li>● L'utente registrato inserisce una mail già nel sistema: il sistema comunica all'utente che la sua mail fa parte di un profilo già creato, viene reindirizzato alla pagina di login.</li> </ul>

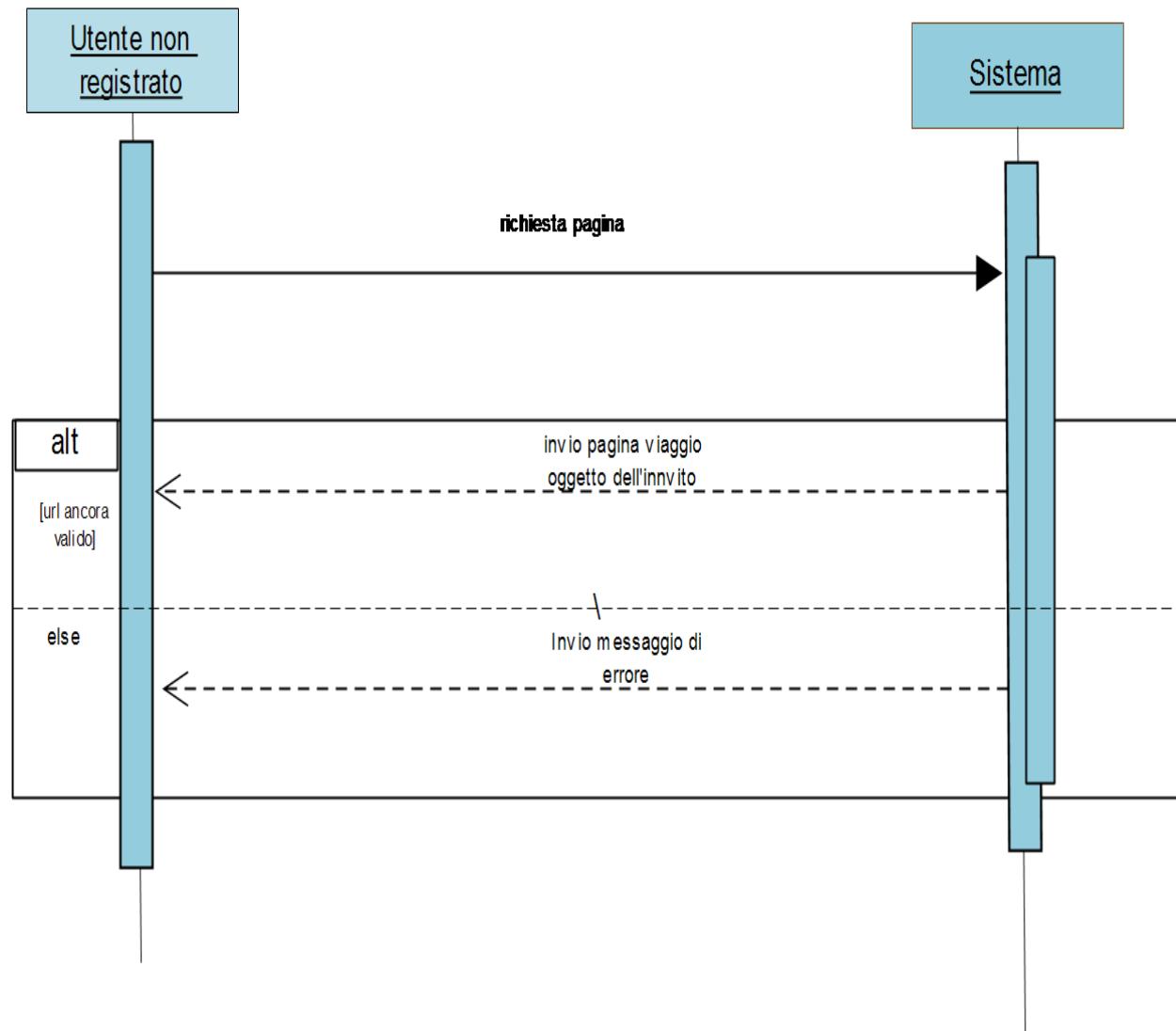




### 3.2.3.8 Visualizza viaggio a cui è stato invitato (utente non registrato)

<b>Titolo caso d'uso</b>	Utente non registrato visualizza un viaggio a cui è stato invitato a partecipare
<b>Attori coinvolti</b>	Utente non registrato
<b>Condizione d'ingresso</b>	L'utente non registrato accede al sito tramite il link mandatogli per mail
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente accede alla pagina linkata nella mail</li> <li>2. il sistema gli propone la soluzione di viaggio condivisagli dall'amico</li> <li>3. l'utente visualizza i dettagli del viaggio</li> </ol>
<b>Condizione di uscita</b>	I'utente prende visione del viaggio condivisogli dall'amico
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>● il viaggio condiviso non è più disponibile causa scadenza del tempo massimo per la visualizzazione:il sistema risponde con messaggio di errore.</li> </ul>



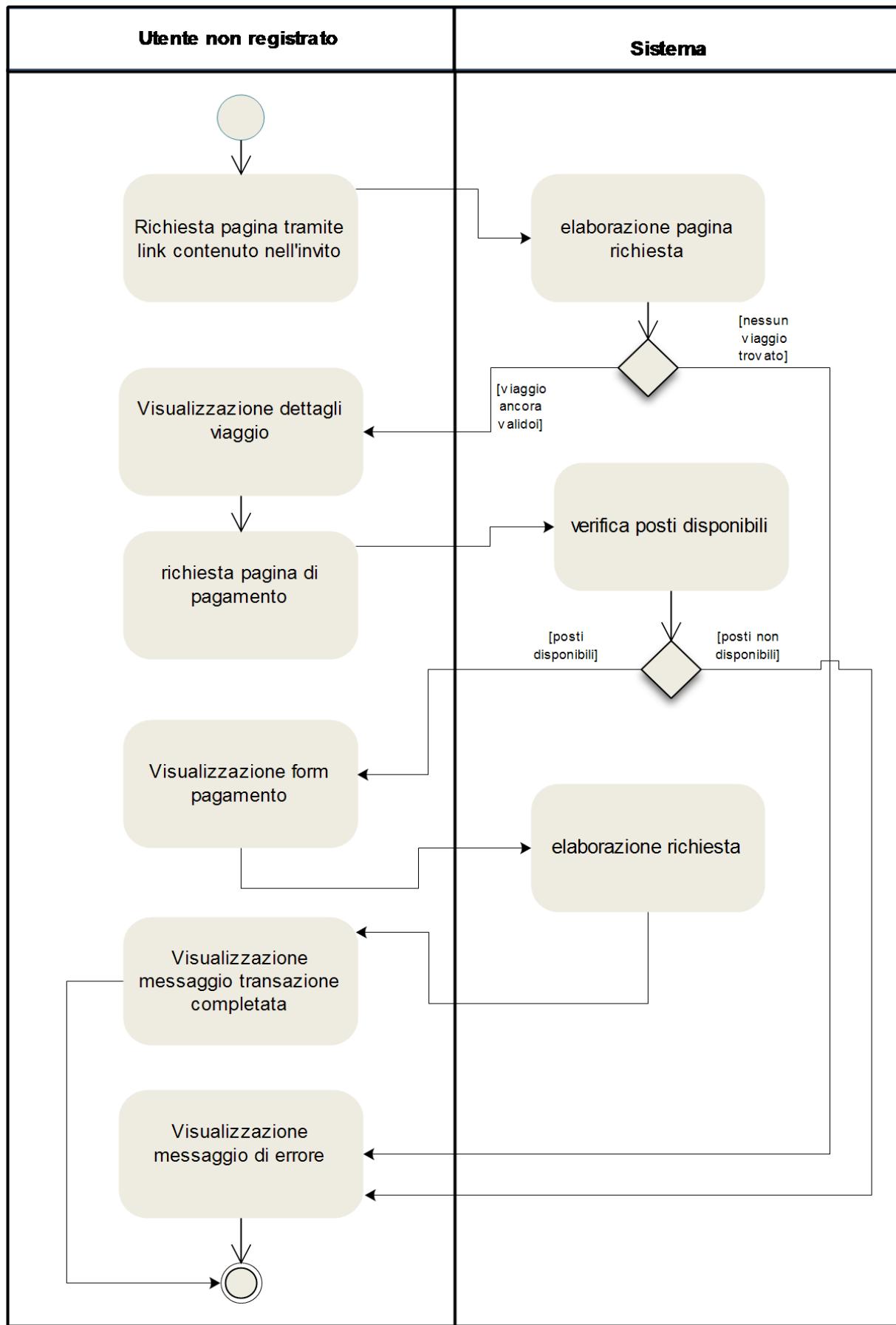


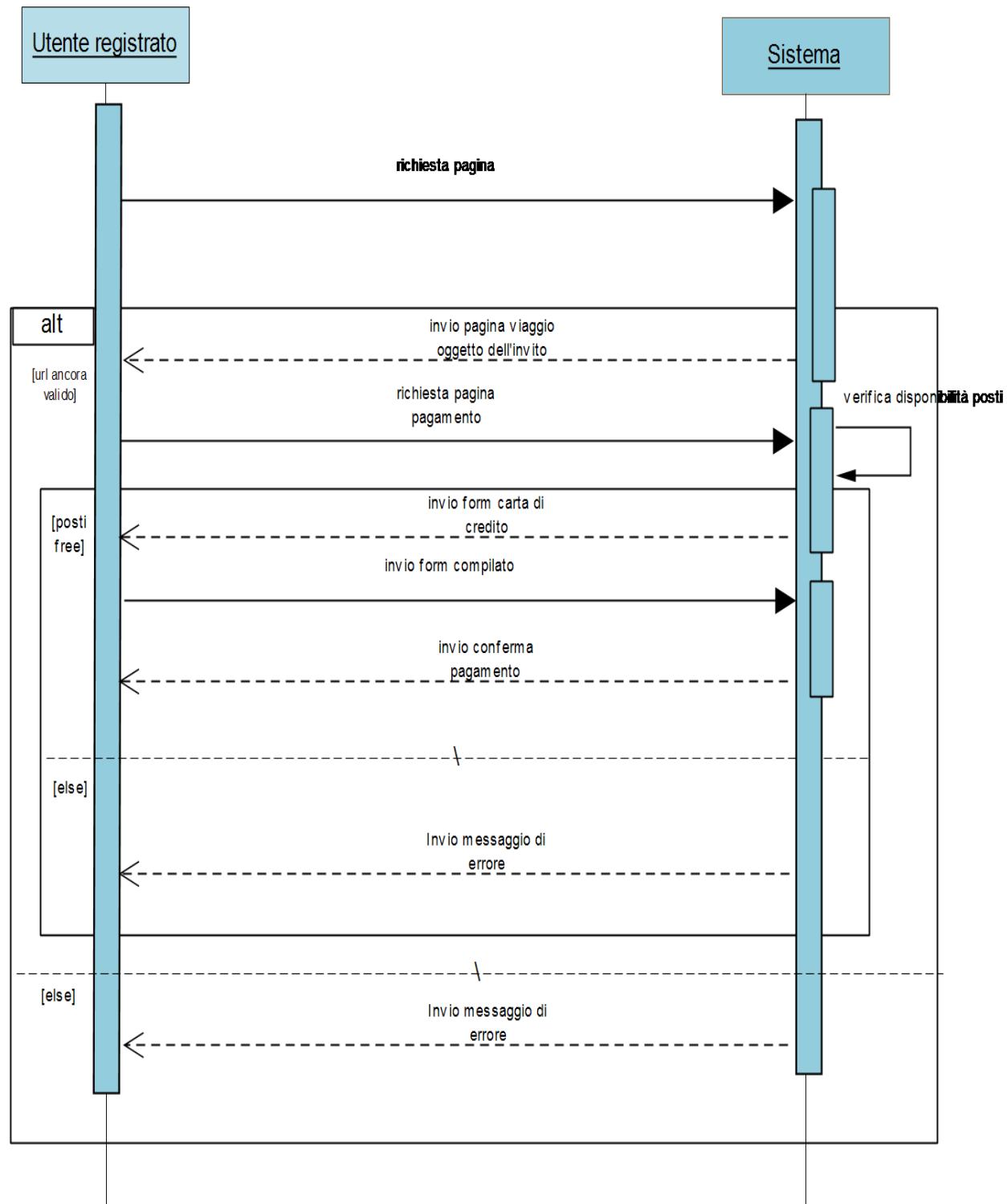
### 3.2.3.9 Visualizza viaggio a cui è stato invitato (utente registrato)

Il seguente caso d'uso comprende anche l'acquisto da parte dell'utente registrato

<b>Titolo caso d'uso</b>	Utente registrato e loggato visualizza un viaggio a cui è stato invitato a partecipare e lo acquista
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	L'utente non registrato accede al sito tramite il link mandatogli per mail
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente accede alla pagina</li> <li>2. il sistema gli propone la soluzione di viaggio condivisagli dall'amico</li> <li>3. l'utente visualizza i dettagli del viaggio</li> <li>1. l'utente accede alla pagina di pagamento</li> <li>2. il sistema verifica la disponibilità dei posti per il viaggio</li> <li>3. il sistema ritorna un messaggio di esito positivo della transazione e un form dove inserire i dati della carta di credito dell'utente</li> <li>4. l'utente inserisce i dati richiesti e invia il form</li> <li>5. il sistema passa i dati al sistema esterno per il pagamento e ritorna un messaggio di conferma dell'acquisto</li> </ol>
<b>Condizione di uscita</b>	Il pagamento va a buon fine

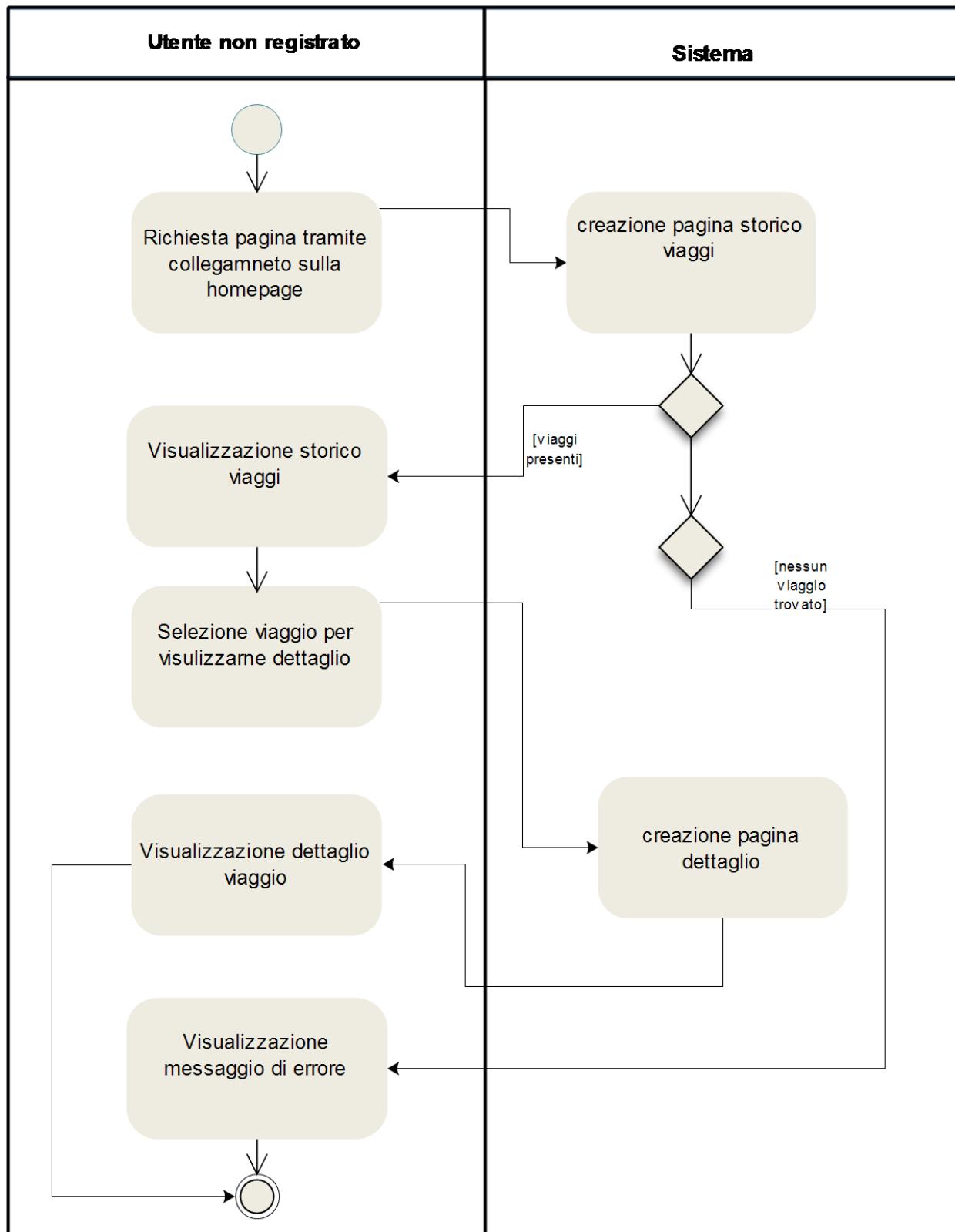
<b>Eccezioni</b>	<ul style="list-style-type: none"><li>• posti non disponibili per il viaggio consigliato dall'amico:il sistema ritorna un messaggio di errore (rollback della transazione)</li><li>• il viaggio condiviso non è più disponibile causa scadenza del tempo massimo per la visualizzazione:il sistema risponde con messaggio di errore.</li></ul>
------------------	--

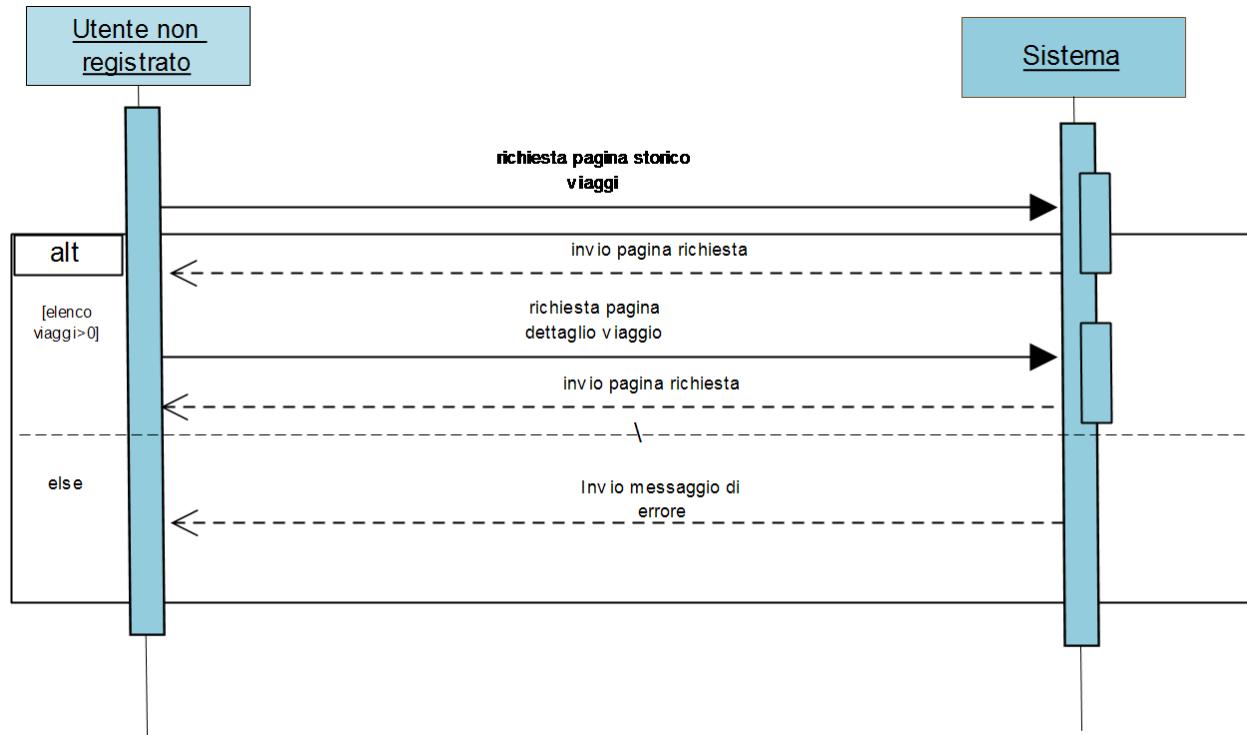




### 3.2.3.10 Visualizza cronologia viaggi

<b>Titolo caso d'uso</b>	Utente registrato e loggato visualizza i dettagli di un suo viaggio prenotato
<b>Attori coinvolti</b>	Utente registrato
<b>Condizione d'ingresso</b>	Utente registrato accede alla pagina "storico viaggi"
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. l'utente accede alla pagina tramite apposito link</li> <li>2. il sistema risponde con l'elenco degli ultimi viaggi prenotati</li> <li>3. l'utente seleziona il viaggio del quale vuole visualizzare i dettagli</li> <li>4. il sistema risponde con la pagina del dettaglio del viaggio richiesto</li> </ol>
<b>Condizione di uscita</b>	visualizzazione della pagina da parte dell'utente
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>● nessun viaggio trovato: il sistema mostra all'utente un messaggio.</li> </ul>

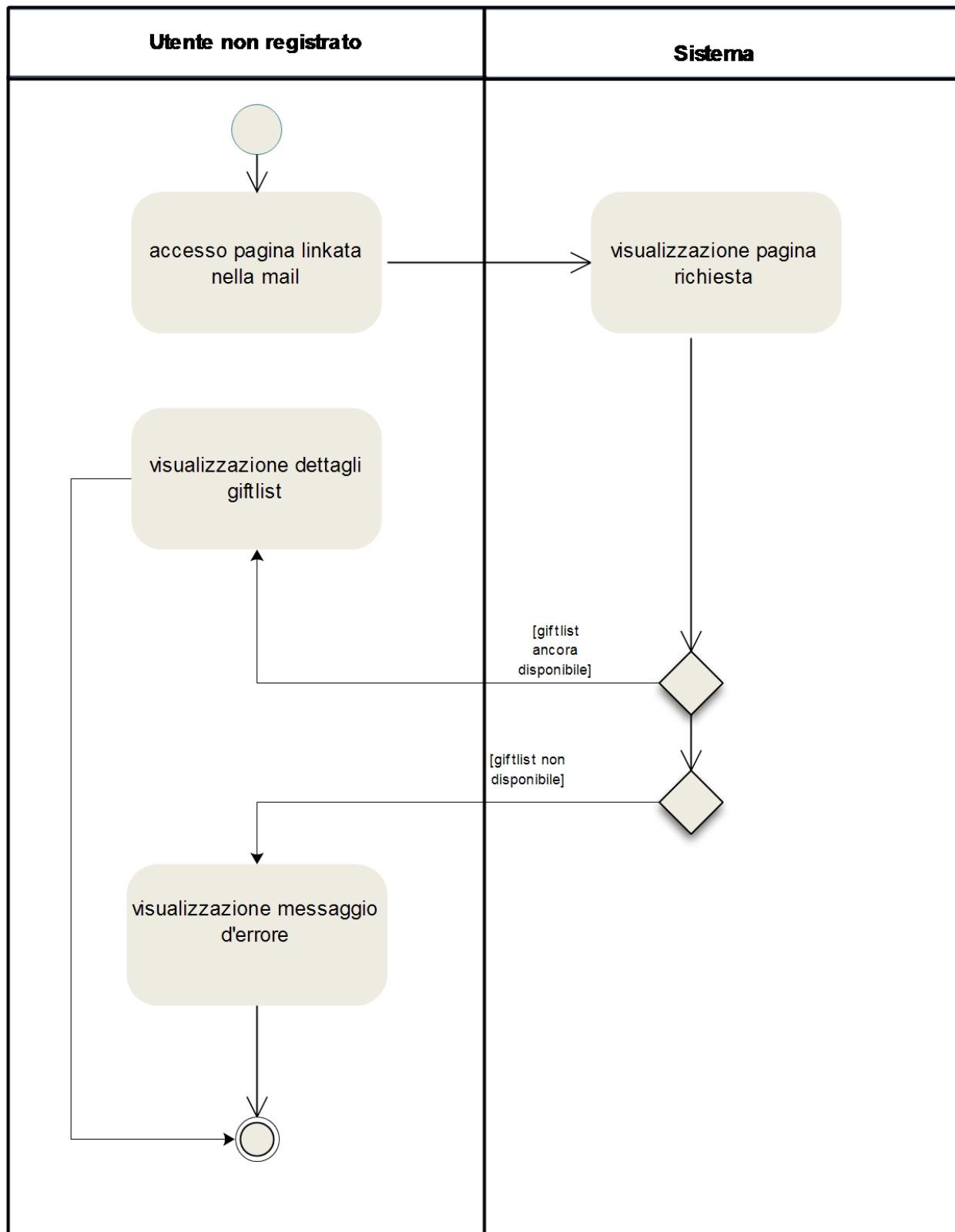


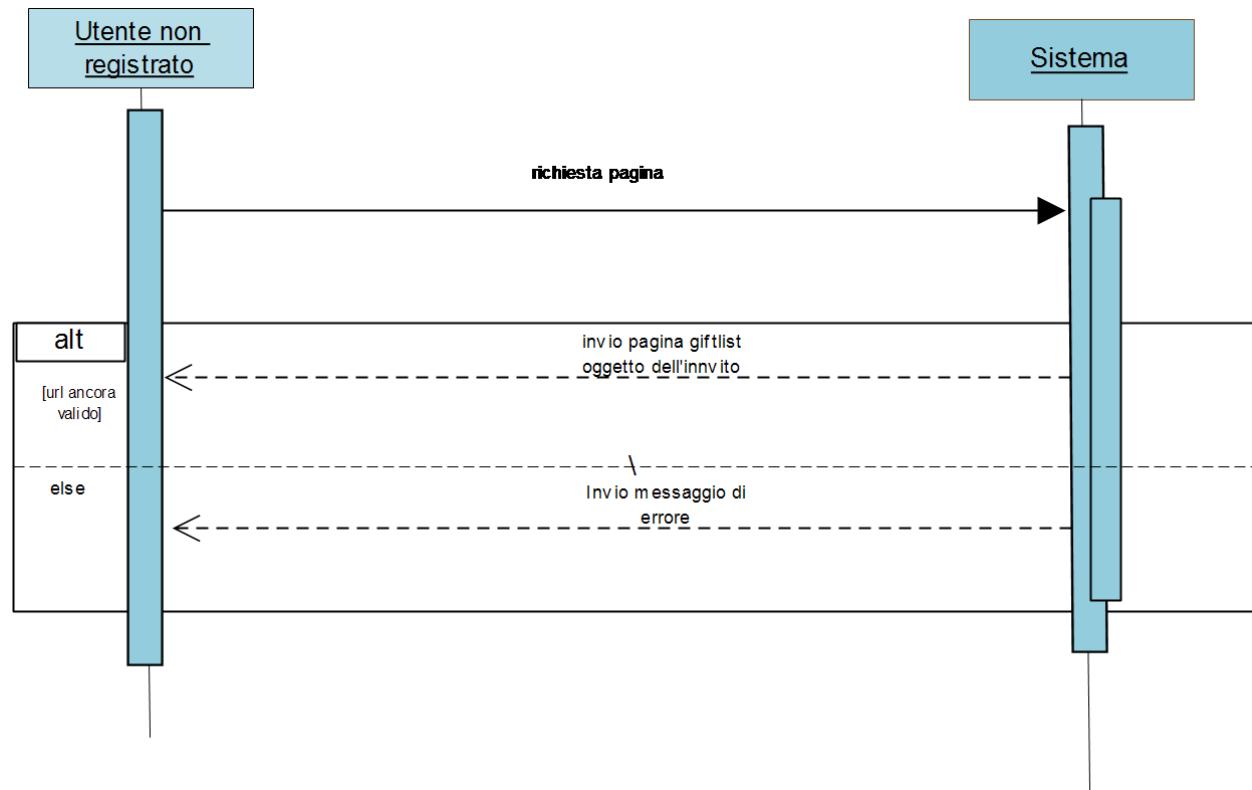


### 3.2.3.11 Visualizza giftlist a cui è stato invitato (utente non registrato)

<b>Titolo caso d'uso</b>	Utente non registrato visualizza una giftlist al quale è stato invitato a partecipare
<b>Attori coinvolti</b>	Utente non registrato
<b>Condizione d'ingresso</b>	L'utente non registrato accede al sito tramite il link mandatogli per mail
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L'utente accede alla pagina</li> <li>2. il sistema gli propone la giftlist condivisagli dall'amico</li> <li>3. l'utente visualizza i dettagli della giftlist</li> </ol>
<b>Condizione di uscita</b>	I'utente prende visione della giftlist condivisagli dall'amico
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>• la giftlist condivisa non è più disponibile</li> </ul>

	<p>causa scadenza del tempo massimo per la visualizzazione: il sistema risponde con messaggio di errore.</p>
--	--

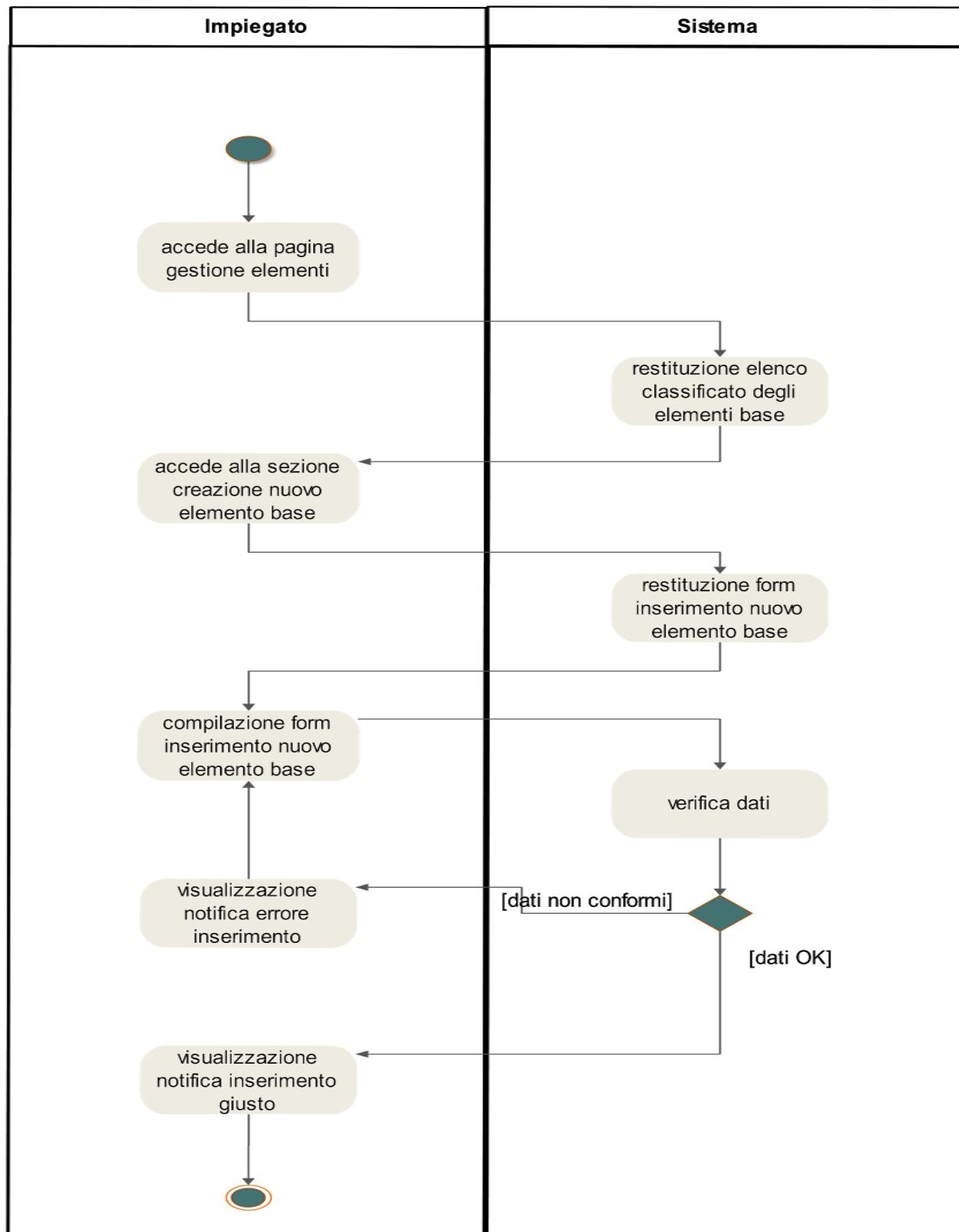


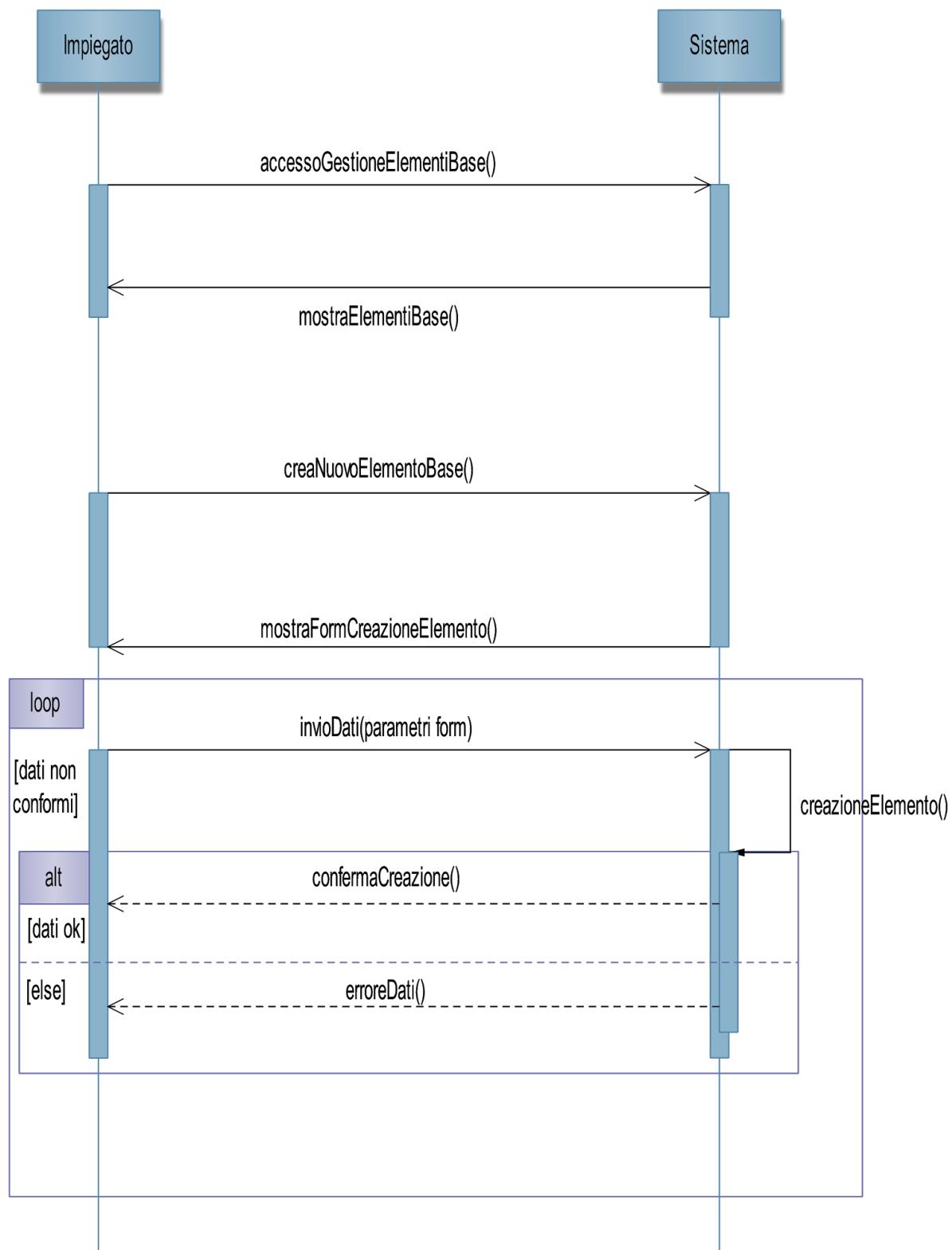


### 3.2.3.12 Creazione di un elemento base

Il caso d'uso comprende anche “Gestire un elemento base” da parte dell’impiegato.

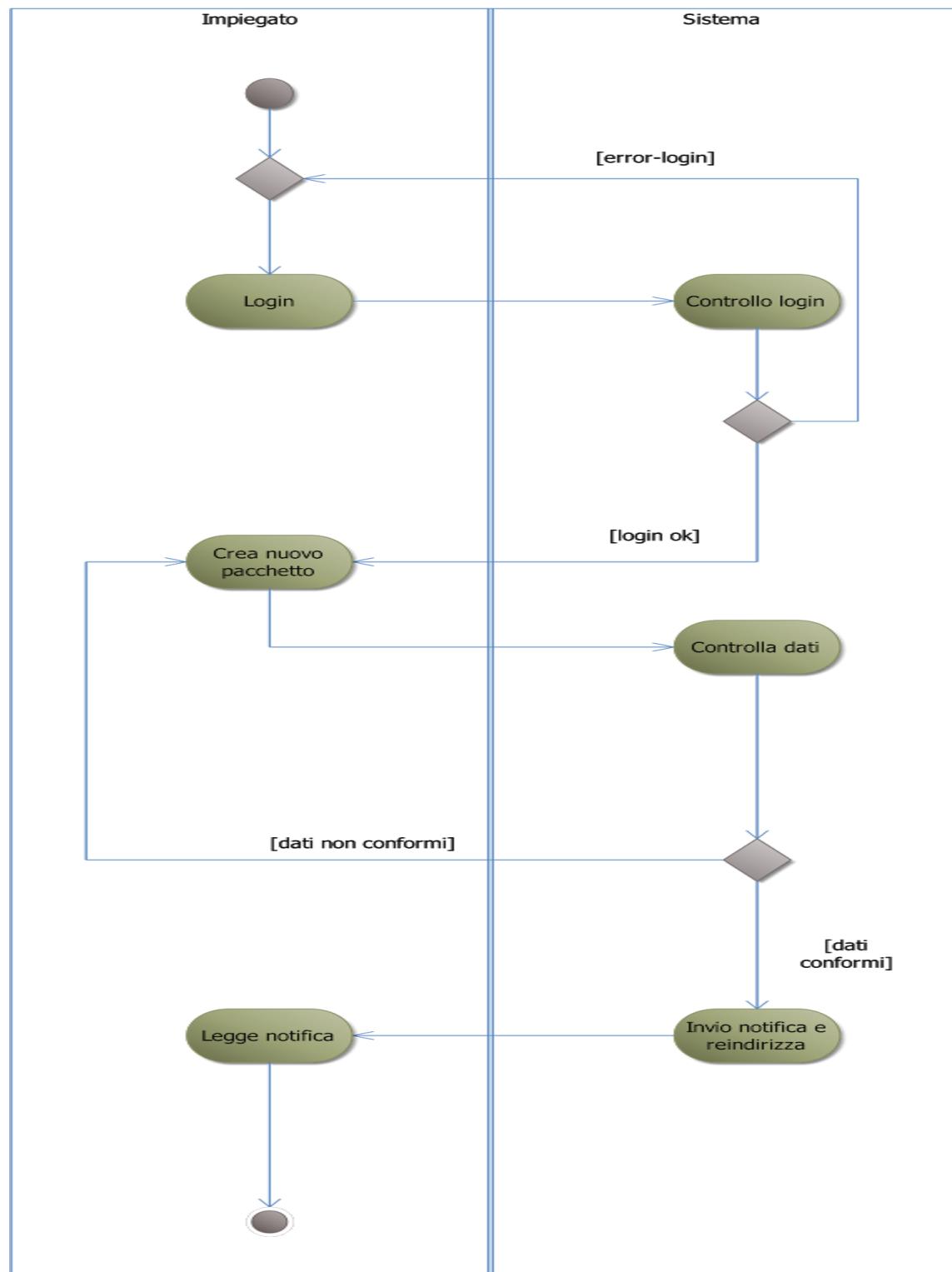
<b>Titolo caso d'uso</b>	Inserire elemento base
<b>Attori coinvolti</b>	Impiegato
<b>Condizione d'ingresso</b>	L’impiegato autenticato accede alla pagina di gestione degli elementi base.
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L’impiegato accede alla pagina di gestione degli elementi base.</li> <li>2. Il sistema restituisce l’elenco classificato di tutti gli elementi base presenti.</li> <li>3. L’impiegato accede alla sezione per inserire un nuovo elemento base.</li> <li>4. Il sistema mostra il form per l’inserimento dei dati.</li> <li>5. L’impiegato inserisce e invia al sistema i dati del nuovo elemento.</li> <li>6. Il sistema risponde con l’esito dell’operazione.</li> </ol>
<b>Condizione di uscita</b>	L’impiegato riceve notifica di avvenuto inserimento e viene reindirizzato nella pagina di gestione degli elementi base.
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>● L’impiegato inserisce dati non conformi (es. campi obbligatori lasciati vuoti, numeri). Il sistema notifica l’errore e fa ripetere la procedura.</li> </ul>

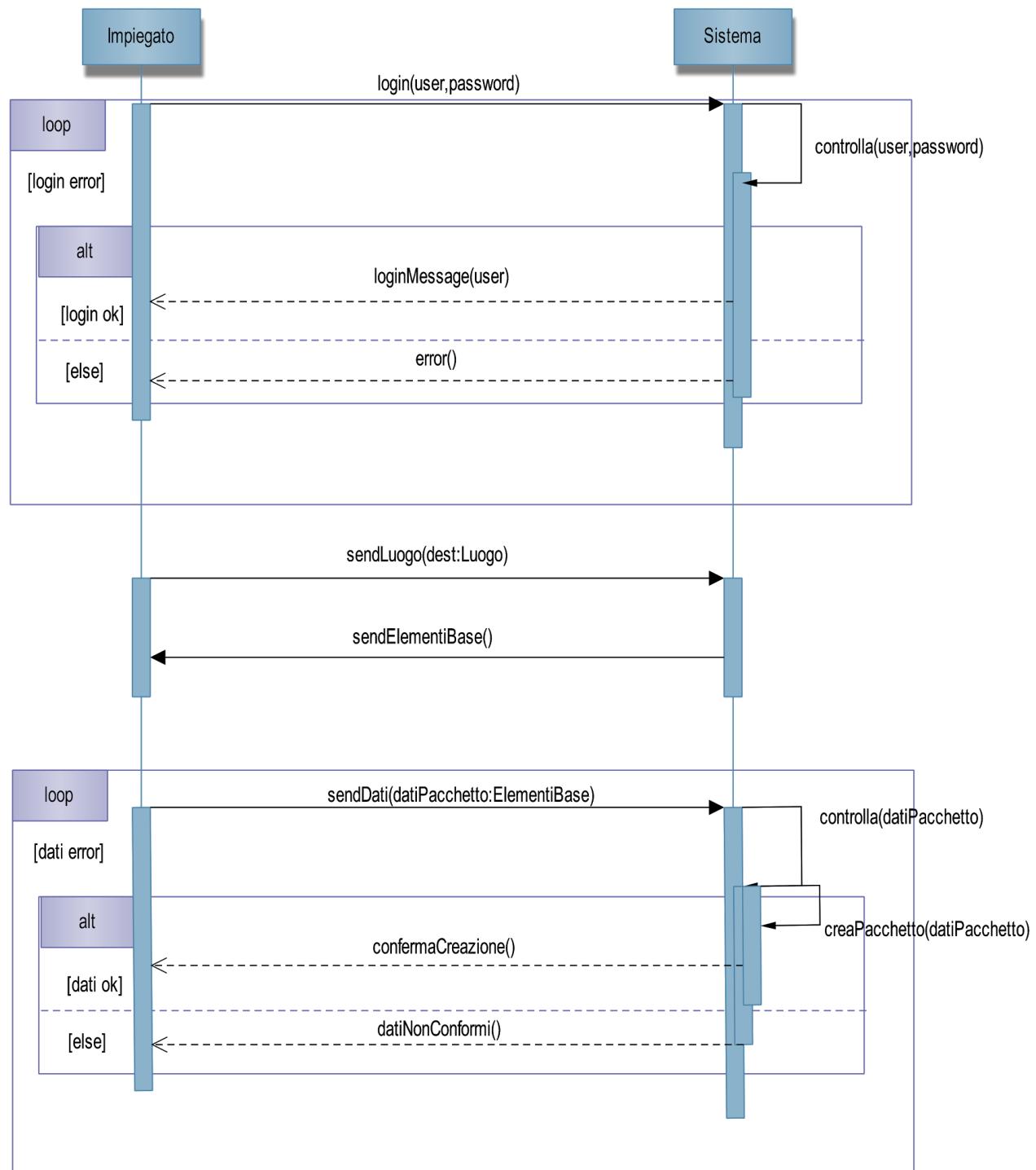




### 3.2.3.13 Creazione pacchetto predefinito

<b>Titolo caso d'uso</b>	Creazione pacchetto predefinito
<b>Attori coinvolti</b>	Impiegato
<b>Condizione d'ingresso</b>	L'impiegato deve essere loggato al sito con le proprie credenziali.
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L'impiegato accede alla sezione per creare un nuovo pacchetto.</li> <li>2. Il sistema mostra il form per l'inserimento dei dati del pacchetto.</li> <li>3. L'impiegato inserisce il luogo di destinazione del viaggio.</li> <li>4. Il sistema restituisce alberghi, mezzi di trasporto ed escursioni corrispondenti.</li> <li>5. L'impiegato sceglie albergo, trasporto, periodo ed escursioni.</li> <li>6. Il sistema risponde con l'esito dell'operazione.</li> </ol>
<b>Condizione di uscita</b>	L'impiegato riceve notifica di avvenuta creazione del pacchetto predefinito e viene reindirizzato nella pagina principale di amministrazione.
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>• L'impiegato inserisce dati non conformi (es. giorni e posti-persona disponibili, campi obbligatori lasciati vuoti). Il sistema notifica l'errore e fa ripetere la procedura.</li> </ul>

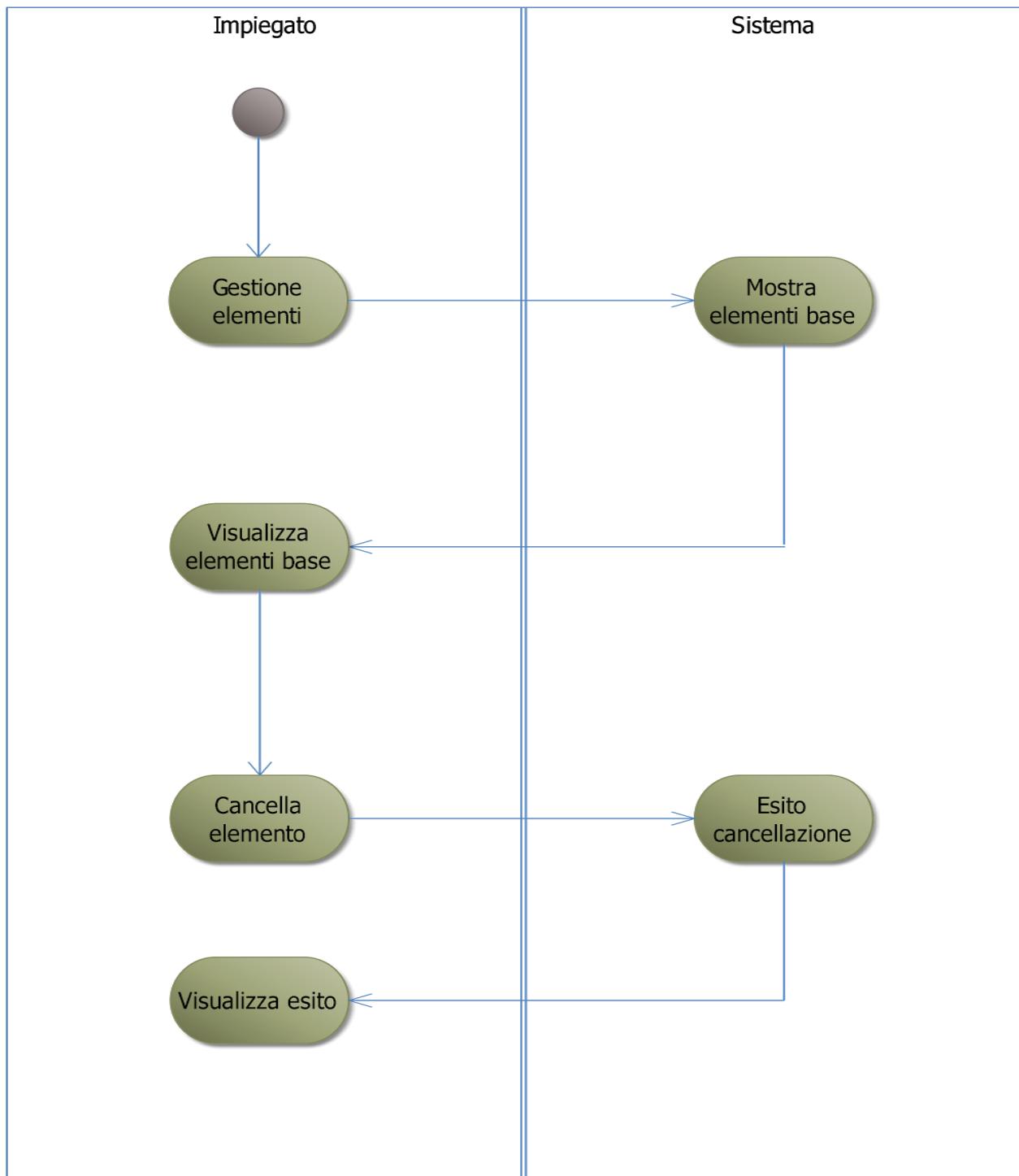


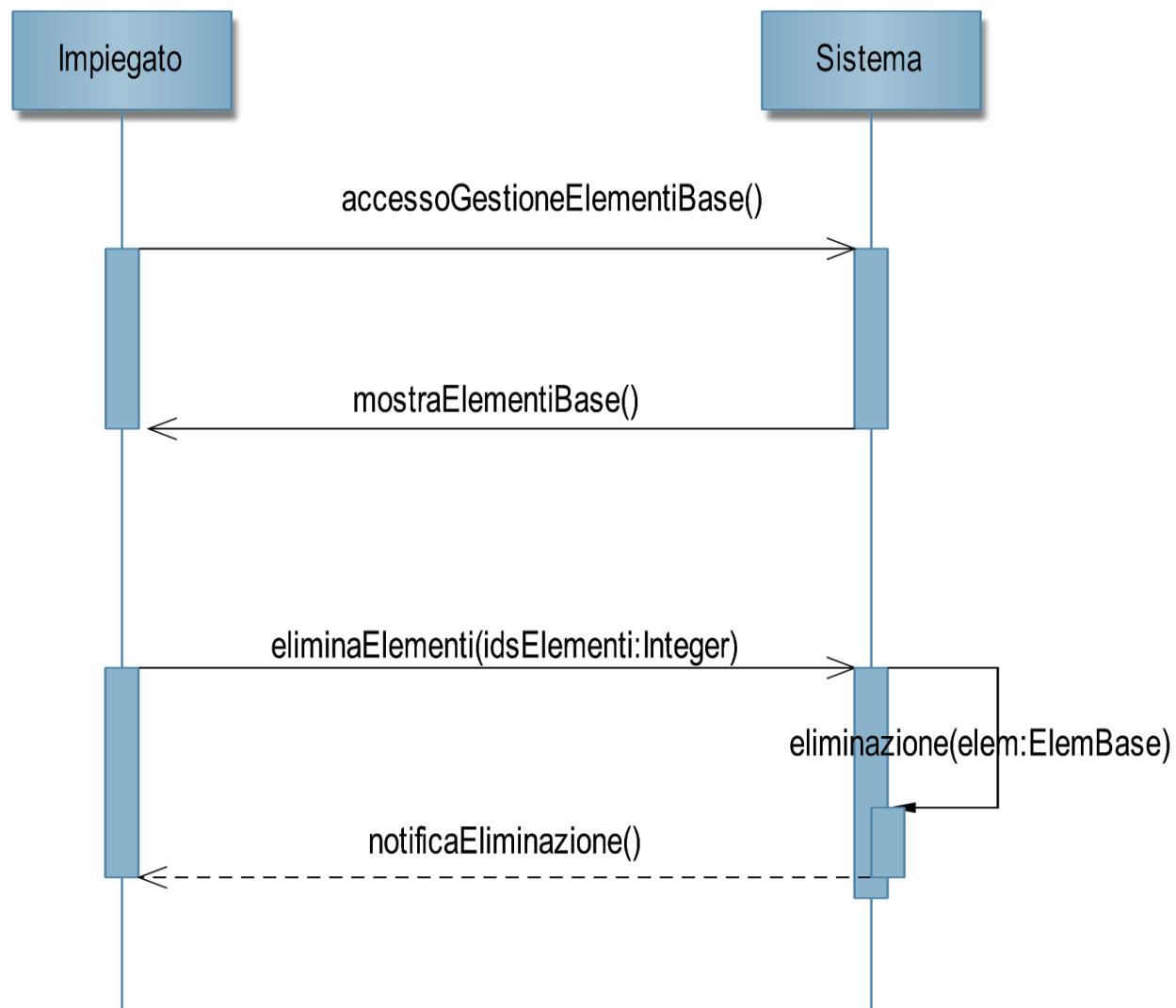


### 3.2.3.14 Elimina elemento base

Il caso d'uso comprende anche “Gestire un elemento base” da parte dell’impiegato.

<b>Titolo caso d'uso</b>	Eliminare elemento base
<b>Attori coinvolti</b>	Impiegato
<b>Condizione d'ingresso</b>	L’impiegato autenticato accede alla pagina di gestione degli elementi base.
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L’impiegato accede alla pagina di gestione degli elementi base.</li> <li>2. Il sistema restituisce l’elenco, categorizzato per elemento, di tutti gli elementi base presenti.</li> <li>3. L’impiegato seleziona l’elemento base da eliminare</li> <li>4. L’impiegato elimina l’elemento.</li> <li>5. Il sistema rimuove dall’elenco l’elemento base eliminato.</li> </ol>
<b>Condizione di uscita</b>	L’impiegato riceve notifica di avvenuta eliminazione e viene reindirizzato nella pagina di gestione degli elementi base.
<b>Eccezioni</b>	-



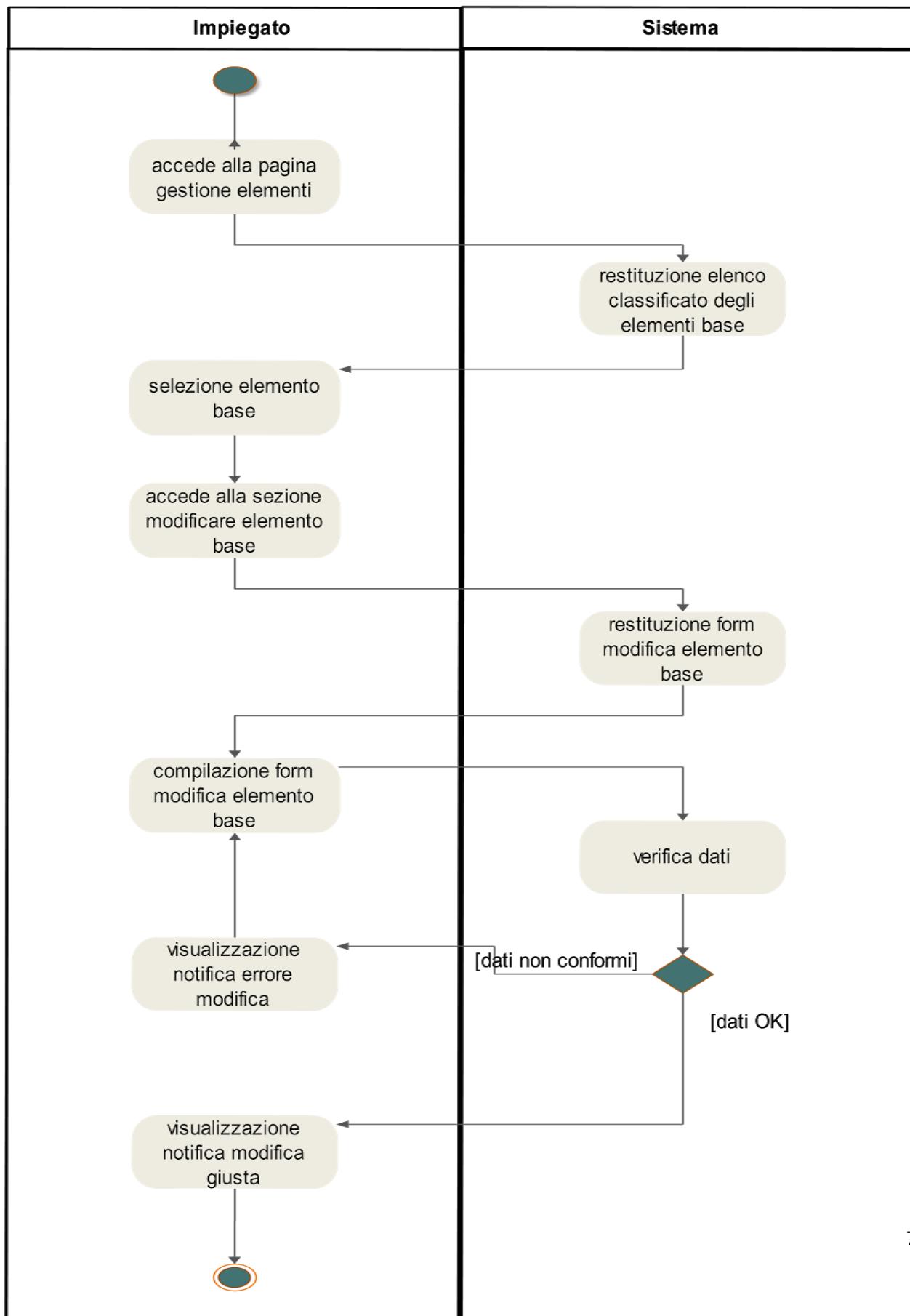


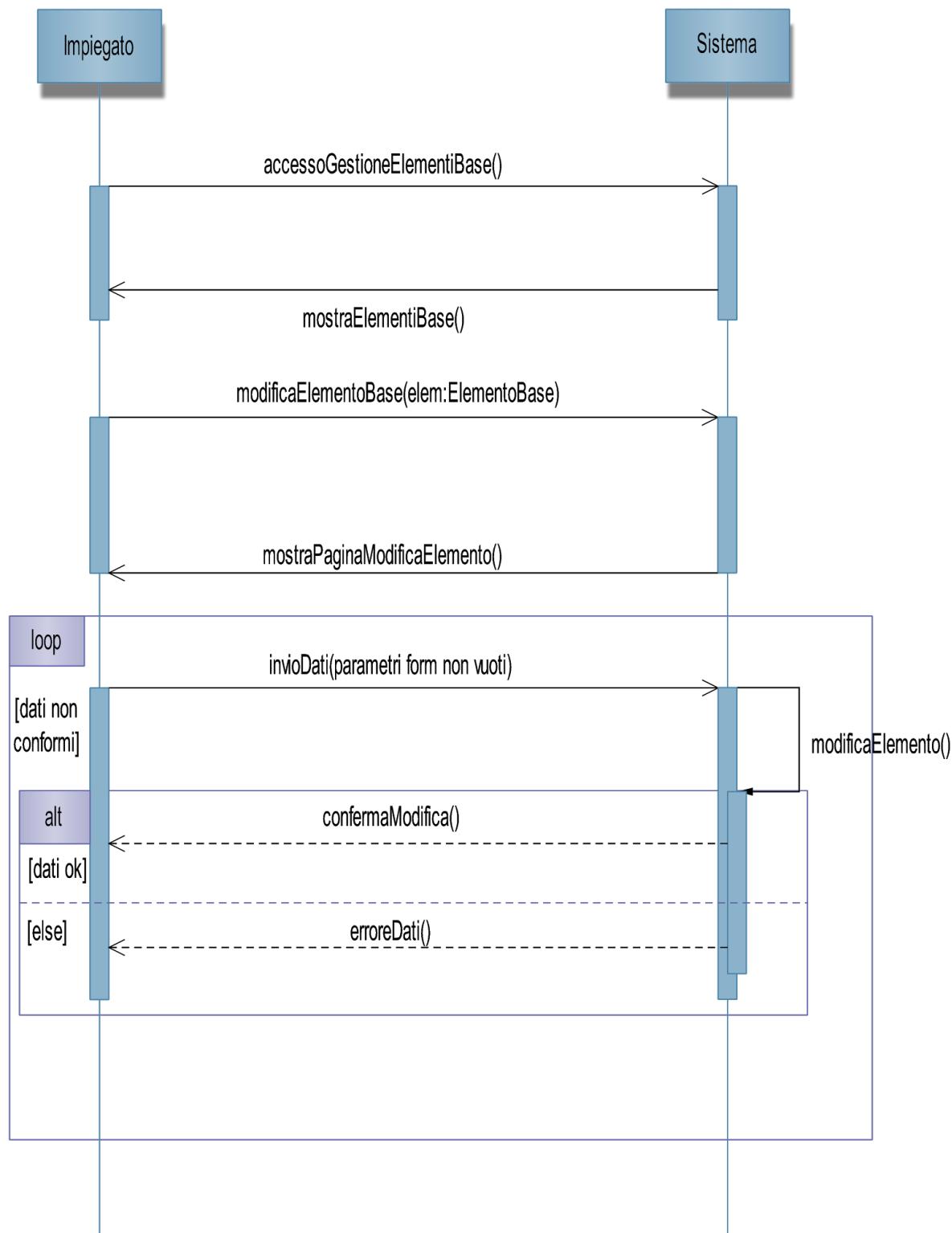
### 3.2.3.15 Modifica elemento base

Il caso d'uso comprende anche “Gestire un elemento base” da parte dell’impiegato.

<b>Titolo caso d'uso</b>	Modificare elemento base
<b>Attori coinvolti</b>	Impiegato
<b>Condizione d'ingresso</b>	L’impiegato autenticato accede alla pagina di gestione degli elementi base.
<b>Flusso degli eventi</b>	<ol style="list-style-type: none"> <li>1. L’impiegato accede alla pagina di gestione degli elementi base.</li> <li>2. Il sistema restituisce l’elenco, categorizzato per elemento, di tutti gli elementi base presenti.</li> <li>3. L’impiegato sceglie l’elemento e accede alla sezione per modificare l’elemento base in questione.</li> <li>4. Il sistema mostra la pagina di modifica dell’elemento con form per l’inserimento dei nuovi dati.</li> <li>5. L’impiegato inserisce e conferma i nuovi dati inseriti riferiti al campo selezionato.</li> <li>6. Il sistema risponde con l’esito dell’operazione.</li> </ol>
<b>Condizione di uscita</b>	L’impiegato riceve notifica di avvenuta modifica e viene reindirizzato nella pagina di gestione degli elementi base.
<b>Eccezioni</b>	<ul style="list-style-type: none"> <li>● L’impiegato, nel momento di modificare un campo, inserisce dati non conformi (es. numeri). il sistema notifica l’errore e fa ripetere la procedura di modifica di quel determinato attributo.</li> <li>● L’impiegato lascia dei campi vuoti. Il sistema intende tali campi come non</li> </ul>

	modificati e mantiene inalterato il dato precedente alla modifica.
--	--





## 3.3 NON FUNCTIONAL REQUIREMENTS

### 3.3.1 Performance requirements

Poiché il sistema dovrà gestire un elevato numero di utenti e quindi allo stesso tempo un elevato numero di accesso contemporaneo alle sue funzioni, è richiesto che il server sia collegato ad una connessione a banda larga che soddisfi il traffico di dati generati.

Le prestazioni percepite dagli utenti dovranno essere esclusivamente influenzate dalla qualità delle connessioni internet di cui dispongono.

### 3.3.2 Design constraints

Il vincolo identificato nella fase di design è l'utilizzo delle tecnologie J2EE.

### 3.3.3 Requisiti legati alla base dati

La base dati sarà progettata al fine di memorizzare tutte le informazioni rilevanti per il sistema identificate nel dettaglio nel diagramma delle classi presente in questo documento. Si userà un database MySql.

### 3.2.3 Vincoli generali di progetto

Il sistema software sarà progettato ed implementato seguendo le caratteristiche della piattaforma Java Enterprise Edition (JEE).

### 3.2.4 Standard adottati

I documenti prodotti rispecchieranno, per quanto possibile, gli standard dell'ingegneria del software.

L'interfaccia utente dovrà essere sviluppata tenendo in considerazione alcune raccomandazioni derivanti dalla buona progettazione di interfacce utenti. Tra queste direttive rientrano la profondità di navigazione mentre si interagisce con il sistema; la possibilità di accedere nel modo più rapido possibile a tutte le funzionalità disponibili; l'utilizzo di collegamenti chiari sia dal punto di vista del linguaggio che della "comunicazione visiva" per accedere alle varie sezioni del sistema.

Il codice software che verrà prodotto dovrà rispettare le migliori regole di programmazione al fine di renderlo leggibile e facilmente manutenibile.

### 3.3 SYSTEM ATTRIBUTES

#### 3.4.1 Reliability

Il sistema deve garantire il funzionamento dei servizi offerti dal sistema tramite la persistenza dei dati contenuti nel database.

#### 3.4.2 Availability

Il sistema dovrà essere accessibile sempre, tranne in casi di manutenzione straordinaria.

#### 3.4.3 Security

La fase di login dovrà utilizzare protocolli di comunicazione criptati in modo da garantire l'invio sicuro delle credenziali di accesso. La fase di pagamento e la fase di invio/ricezione mail saranno gestite da sistemi esterni.

#### 3.4.4 Maintainability

Si consiglia di ottimizzare e deframmentare il database una volta all'anno. In particolare, ripulire le tabelle corrispondenti agli utenti non più attivi da almeno un anno, deframmentare gli indici e mantenere sempre alta l'efficienza del sistema. Insieme alla pulizia, si consiglia anche un backup periodico. Il backup è in generale qualunque operazione di manutenzione del db, saranno effettuati in ore notturne (secondo il nostro fuso orario) in modo da non sovraccaricare troppo il sistema o evitare di avere il sistema inutilizzabile da parte degli utenti.

#### 3.4.5 Portability

Il sistema può essere installato su macchine dotate di Java Virtual Machine. Grazie alla JVM la portabilità è garantita, il che significa che può essere usato su ogni sistema operativo.

## 4. APPENDICES

### 4.1 ALLOY CODE

La presente sezione illustra la modellazione del diagramma delle classi, presentato in precedenza, utilizzando il linguaggio Alloy. Tale linguaggio consente di verificare la consistenza del diagramma delle classi prodotto, ricorrendo ad una rappresentazione formale delle singole entità e delle rispettive relazioni. In particolare permette di stabilire la consistenza di tutte le assunzioni fatte andando a verificare alcune asserzioni e cercando di generare delle istanze di "mondi" che le rispettino. L'analizzatore utilizzato, nel generare i "mondi", va a creare delle istanze all'interno di un dominio limitato e specificato; tali risultati, quindi, garantiscono la consistenza della modellazione realizzata per la fascia di dominio scelta e sono indizio di una sua probabile correttezza anche sull'intero dominio. Sono riportati di seguito signatures, fact, assertions e predicates opportunamente selezionati e ritenuti indispensabili. Sono stati tralasciati futili vincoli già esplicitati nel testo e ribaditi più volte nel documento.

#### 4.1.2 Signatures

```

abstract sig Account {
    user: one Stringa,
    psw: one Stringa
}

sig UtenteNonRegistrato{}

abstract sig ElementoBase{}

sig UtenteRegistrato extends Account{
    email:one Stringa,
    nome: one Stringa,
    cognome: one Stringa,
    pacchetti: set Pacchetto
}
{
    #pacchetti>0
}
sig Impiegato extends Account{}

sig Stringa{}

sig Data{
    timestamp: one Int,
    /*anno: one Int, //non mettiamo i componenti della data per rendere il diagramma più leggibile|
    mese: one Int,
    giorno: one Int,
    ora: one Int,
    minuti: one Int*/
}

```

```

} {
    timestamp>0
    //non mettiamo vincoli sui componenti data per facilitare l'esecuzione del codice e visualizzazione del
    //risultato
    /*anno>2013
    mese > 0
    mese<13
    giorno > 0
    giorno<30*/
}

sig Luogo{ //attributi non necessari nel documento
}

sig PartecipanteGift{
    utente: one UtenteRegistrato,
    elementiPagati: set ElementoBase
}

}

sig GiftCard{
    pacchetto:one Pacchetto,
    invitati:set Stringa,
    partecipanti: set PartecipanteGift
}

}

sig Escursioni extends ElementoBase{
    luogo:one Luogo,
    data:one Data,
    costoPersona:one Int,
}

{
    costoPersona>0
}

sig Pacchetto{
    id:one Int,
    dataInizio:one Data,
    dataFine:one Data,
    numPartecipanti:one Int,
    albergo:one Albergo,
    meta:one Luogo,
    mezzoUsatoAndata:one MezzoDiTrasporto,
    mezzoUsatoRitorno:one MezzoDiTrasporto,
    escursioni: set Escursioni,
    utentiInvitatiRegistrati: set UtenteRegistrato,
    utentiInvitatiNonRegistrati: set UtenteNonRegistrato,
    camerePrenotate: set Camera
}

{
    id>0
    numPartecipanti > 0
    dataInizio.timestamp<dataFine.timestamp
}

```

```

sig MezzoDiTrasporto extends ElementoBase{
    id:one Int,
    postiDisponibili:one Int,
    dataPartenza:one Data,
    partenza:one Luogo,
    destinazione: one Luogo,
    dataArrivo: one Data,
}

}{  

    postiDisponibili>=0  

    dataArrivo.timestamp>dataPartenza.timestamp  

    id>0
}

sig Albergo extends ElementoBase{
    codice:one Int,
    nome:some Stringa,
    locazione:one Luogo,
    camereLibere:set Camera,
    camereOccupate:set Camera
}
{  

    codice>0  

    #camereLibere + #camereOccupate > 0
}

sig Camera{
    numero: one Int,
    nPosti: one Int,
    costoPersona: one Int,
    inizioPren: one Data,
    finePren: one Data
}
{  

    numero > 0  

    nPosti > 0  

    costoPersona > 0
}

```

---

### 4.1.3 Fact

```

//un volo ha date concordi con quelle del pacchetto

fact stessoGiornoArrivoDellInizioPacchetto{
  all p:Pacchetto,m:MezzoDiTrasporto,d,d':Data|(p.mezzoUsatoAndata=m and m.dataArrivo=d and
  p.dataInizio=d')implies d=d'
}

fact stessoGiornoPartenzaDellaFine{
  all p:Pacchetto,m:MezzoDiTrasporto,d,d':Data|(p.mezzoUsatoRitorno=m and m.dataPartenza=d and
  p.dataFine=d')implies d=d'
}

fact aereoAndataDiversoAereoRitorno{
  all p:Pacchetto,m,m':MezzoDiTrasporto|(p.mezzoUsatoAndata=m and p.mezzoUsatoRitorno=m') implies m!=m'
}

//escursioni e camere hanno date concordi con quelle del pacchetto

fact periodoPrenotazioneCamereCoerente{
  all p:Pacchetto,a:Albergo,c:Camera|p.albergo = a and c in p.camerePrenotate implies (c.inizioPren = p.dataInizio and c.finePren = p.dataFine)
}

fact dataEscursioneCoerente{
  all p:Pacchetto,e:Escursioni|e in p.escursioni implies (e.data.timestamp >= p.dataInizio.timestamp and e.data.timestamp <= p.dataFine.timestamp)
}

//in un volo luogo di partenza e arrivo non possono coincidere

fact noSameDepartenceAndArrived{
  all m:MezzoDiTrasporto|not m.partenza=m.destinazione
}

//stesso luogo per albergo di un pacchetto

fact stessoLuogoDiPacchettoEscursioni{
  all e:Albergo,p:Pacchetto|p.meta=e.locazione
}

//stesso luogo per escursioni di un pacchetto

fact stessoLuogoDiPacchettoEscursioni{
  all e:Escursioni,p:Pacchetto|p.meta=e.luogo
}

//il volo mi deve portare nel luogo in cui c'è il pacchetto

fact destinazioneVoloAndataComePacchetto{
  all e:MezzoDiTrasporto,p:Pacchetto| p.mezzoUsatoAndata=e implies p.meta=e.destinazione
}

//il volo di ritorno deve partire dal luogo in cui c'è il pacchetto e ha come destinazione la partenza del volo d'andata

fact ilVoloDiRitornoMiRiportaDaLuogoPacchettoACasa{
  all e,e':MezzoDiTrasporto,p:Pacchetto|p.mezzoUsatoRitorno=e and p.mezzoUsatoAndata=e' implies p.meta=e.partenza
  and e.destinazione=e'.partenza and p.meta=e'.destinazione
}

//l'albergo è situato nello stesso luogo previsto dal pacchetto

fact albergoRisiedeInDestinazione{
  all h:Albergo,p:Pacchetto|p.meta=h.locazione
}

```

```

//un elemento base della gift acquistato da un utente non può essere riacquistato

fact noAcquistiDoppi{
all p:PartecipanteGift, g:GiftCard|p in g.partecipanti implies
    (all p':PartecipanteGift, e:ElementoBase|p' != p and p' in g.partecipanti and e in p'.elementiPagati implies
        e not in p.elementiPagati)
}

//gli elementi base pagati dai partecipanti ad una gift appartengono al pacchetto cui appartiene la gift

fact stessoAlbergoPartecipanteEPacchetto{
all a:Albergo, p:Pacchetto, pt:PartecipanteGift, g:GiftCard|g.pacchetto = p and pt in g.partecipanti and a in pt.elementiPagati implies p.albergo = a
}

fact stessoMezzoPartecipanteEPacchetto{
all m:MezzoDiTrasporto, p:Pacchetto, pt:PartecipanteGift, g:GiftCard|g.pacchetto = p and pt in g.partecipanti and m in pt.elementiPagati implies
    (p.mezzoUsatoAndata = m or p.mezzoUsatoRitorno = m)
}

fact stesseEscursioniPartecipanteEPacchetto{
all e:Escursioni, p:Pacchetto, pt:PartecipanteGift, g:GiftCard|g.pacchetto = p and pt in g.partecipanti and e in pt.elementiPagati implies
    e in p.escursioni
}

//un utente non si può invitare alla sua giftcard

fact nonInvitoMeStessoGiftcard{
not some g:GiftCard, p:Pacchetto, u:UtenteRegistrato|(p in u.pacchetti and g.pacchetto=p and u.user in g.invitati)
}

//un utente non si può invitare al suo viaggio

fact nonInvitoMestessoalViaggio{
all p:Pacchetto|some disj u,u':UtenteRegistrato|p in u.pacchetti and u' in p.utentiInvitatiRegistrati
}

//non posso invitare a un viaggio due volte lo stesso user, ciò non vale per l'user non registrato
fact noInvitiViaggioMultipli {
no u: UtenteRegistrato, p: Pacchetto | u in p.utentiInvitatiRegistrati implies
    (not some u': UtenteRegistrato | u=u' and u' in p.utentiInvitatiRegistrati)
}

//non posso invitare a una gitcard due volte lo stesso user, ciò non vale per l'user non registrato
fact noInvitiViaggioMultipli {
all u,u': UtenteRegistrato, g: GiftCard | (u.user + u'.user in g.invitati) implies not (u= u')
}

//il numero di partecipanti a un pacchetto non può superare il numero di posti disponibili nel volo

fact partecipantiNonMaggioriDiPostiDisponibili{
all p:Pacchetto,e,e':MezzoDiTrasporto|p.mezzoUsatoRitorno=e and p.mezzoUsatoAndata=e' implies
    p.numPartecipanti<=e.postiDisponibili and p.numPartecipanti<=e'.postiDisponibili
}

//il numero di partecipanti al pacchetto non può superare i posti disponibili nelle camere

fact partecipantiMinoriAPostiCamere{
all p:Pacchetto,c:Camera|c in p.camerePrenotate and p.numPartecipanti <= mul[#p.camerePrenotate,c.nPosti]
}

//il numero di camere prenotate non deve superare il numero di camere libere dell'albergo

fact camerePrenotateMinoriCamereLibere{
all p:Pacchetto,a:Albergo|p.albergo = a implies #p.camerePrenotate <= #a.camereLibere
}

//le camere prenotate in un pacchetto devono essere dell'albergo specificato nel pacchetto stesso

fact camereInStessoAlbergo{
all p:Pacchetto,a:Albergo,c:Camera|p.albergo = a and c in p.camerePrenotate implies (c in a.camereLibere or c in a.camereOccupate)
}

```

```
//una camera non può far parte di due alberghi

fact camereInStessoAlbergo{
  all a,a':Albergo,c:Camera|a' != a and (c in a.camereLibere or c in a.camereOccupate) implies
    not(c in a'.camereLibere or c in a'.camereOccupate)
}

//una camera occupata non è anche libera

fact noCameraSiaLiberaCheOccupata{
  all a:Albergo,c:Camera|(c in a.camereLibere implies not(c in a.camereOccupate)) and
    (c in a.camereOccupate implies not(c in a.camereLibere))
}

//Garantire che le chiavi delle singole entità siano uniche

fact unicitaChiaviEntita {
  //due alberghi non possono avere lo stesso codice
  no disj h, h': Albergo | h.codice = h'.codice
  //due pacchetti non possono avere lo stesso id
  no disj p, p': Pacchetto | p.id = p'.id
  //due mezzi di trasporto non possono avere lo stesso id
  no disj m, m': MezzoDiTrasporto | m.id = m'.id
  //due utenti non possono avere lo stesso nickname
  no disj a, a': Account | a.user = a'.user
  //due date non possono avere lo stesso timestamp
  no disj d, d': Data | d.timestamp = d'.timestamp
  //due camere non possono avere lo stesso numero
  no disj c, c': Camera | c.numero = c'.numero
}
```

#### 4.1.4 Assertion

```
//un utente non può comprare un pacchetto per più persone di quanti posti nelle camere sono disponibili
assert noPersonesuperioriAiPosti {
all p:Pacchetto,c:Camera|c in p.camerePrenotate and p.numPartecipanti <= mul[#p.camerePrenotate,c.nPosti]
}
check noPersonesuperioriAiPosti for 6

//non ci sono voli che portano in destinazioni diverse dal luogo in cui parto a quello stabilito nel pacchetto
assert noVoliNonCoerenti{
    not some e,e':MezzoDiTrasporto,p:Pacchetto|p.mezzoUsatoRitorno=e and p.mezzoUsatoAndata=e' and
        (p.meta!=e.partenza or e.destinazione!=e'.partenza or p.meta!=e'.destinazione)
}
check noVoliNonCoerenti for 6

//un utente non può creare una giftcard e invitare se stesso
assert noUtenteProprietarioInSuaGiftcard{
    not some g:GiftCard, u,u':UtenteRegistrato, p:Pacchetto|(p in u.pacchetti and g.pacchetto=p and u.user in g.invitati and u=u')
}
check noUtenteProprietarioInSuaGiftcard for 6

//un utente può scegliere un pacchetto ma ciò non implica che debba creare per forza una giftCard
assert noUtenteObbligatoaFareGiftcard{
    all p:Pacchetto,u:UtenteRegistrato| p in u.pacchetti implies
        (no g: GiftCard | g.pacchetto=p)
        or
        (some g: GiftCard | g.pacchetto=p)
}
check noUtenteObbligatoaFareGiftcard for 6

//un utente non può invitare un altro utente più volte ad un viaggio

assert noInvitiMultiplitraUtenti{
    no u: UtenteRegistrato, p: Pacchetto | u in p.utentiInvitatiRegistrati implies
        (not some u': UtenteRegistrato | u=u' and u' in p.utentiInvitatiRegistrati)
}
check noInvitiMultiplitraUtenti for 6
```

#### 4.1.5 Verifiche Assertions

**Executing "Check noPersoneSuperioriAiPosti for 6"**

Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20  
71952 vars. 2616 primary vars. 148652 clauses. 1134ms.  
No counterexample found. Assertion may be valid. 357ms.

**Executing "Check noVoliNonCoerenti for 6"**

Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20  
71124 vars. 2610 primary vars. 146057 clauses. 1138ms.  
No counterexample found. Assertion may be valid. 229ms.

**Executing "Check noUtenteProprietarioInSuaGiftcard for 6"**

Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20  
71224 vars. 2616 primary vars. 146061 clauses. 1137ms.  
No counterexample found. Assertion may be valid. 131ms.

**Executing "Check noUtenteObbligatoaFareGiftcard for 6"**

Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20  
71206 vars. 2610 primary vars. 145960 clauses. 1135ms.  
No counterexample found. Assertion may be valid. 144ms.

#### 4.1.6 Predicates

```

//-----
//PREDICATI
//-----

//Operazione di invito di un amico in un pacchetto

pred InvioInvito(u: UtenteRegistrato, un: UtenteNonRegistrato, p: Pacchetto) {
    p.utentiInvitatiRegistrati = p.utentiInvitatiRegistrati + u
    p.utentiInvitatiNonRegistrati = p.utentiInvitatiNonRegistrati + un
}
run InvioInvito for 4

pred InvioInvitoGift(u: UtenteRegistrato, un: UtenteNonRegistrato, g: GiftCard) {
    g.elencoInvitati = g.elencoInvitati + u
    g.elencoInvitatiNonRegistrati = g.elencoInvitatiNonRegistrati + un
}
run InvioInvitoGift for 4
pred PrenotazioneCamera(u: UtenteRegistrato, p: Pacchetto, a:Albergo, c:Camera, e:Escursioni) {
    p.albergo = a
    p.camerePrenotate = p.camerePrenotate + c
    u.pacchetti = u.pacchetti + p
}
run PrenotazioneCamera for 5

pred show(){}
run show for 8

```

#### 4.1.7 Verifica predicates

**Executing "Run InvioInvitoGift for 4"**  
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20  
 30293 vars. 1500 primary vars. 71216 clauses. 418ms.  
**Instance** found. Predicate is consistent. 876ms.

**Executing "Run InvioInvito for 4"**  
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20  
 31445 vars. 1500 primary vars. 72593 clauses. 339ms.  
**Instance** found. Predicate is consistent. 1459ms.

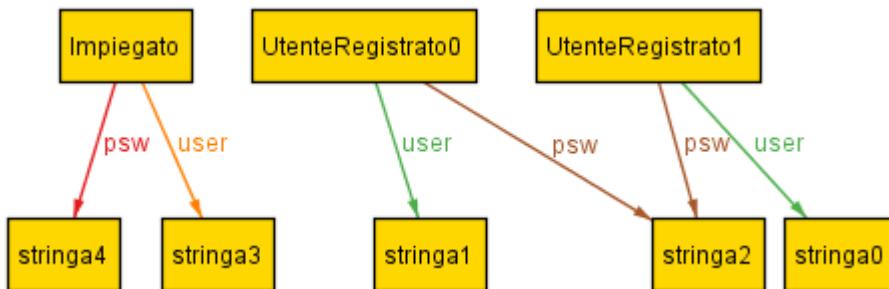
#### 4.1.6 Mondi Generati

Il presente paragrafo mostra una serie di istanze di "mondi" generati attraverso l'analizzatore. Si è scelto di rappresentare alcune delle parti più delicate del sistema. Ma se si vuole tener traccia di tutti i vincoli verificati, è stato, nel capitolo sopra, riportato per intero tutto il codice alloy generato.

##### ★ Mondo 0

Il seguente mondo è stato generato per mostrare l'unicità delle credenziali di accesso per ogni utente registrato e impiegato. In particolare si è scelto di mostrare l'unicità dell'attributo "user" delle entità "UtenteRegistrato" e "Impiegato" tralasciando gli altri attributi e le altre entità.

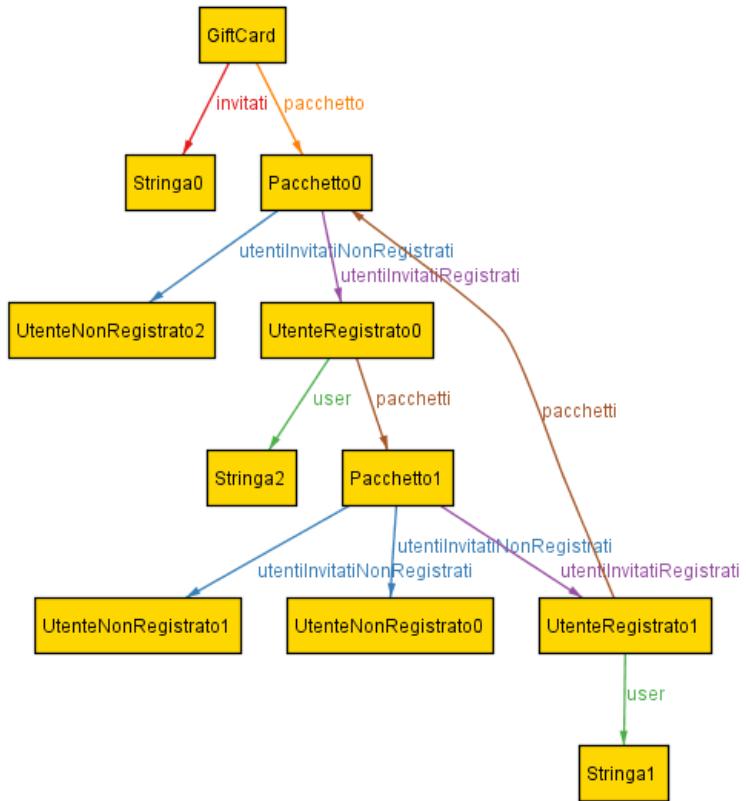
Dalla seguente figura si può notare che ad ogni entità UtenteRegistrato differente corrisponde una stringa diversa che ne rappresenta il proprio "user" e l'impossibilità che questo attributo risulti comune a quello di un impiegato(evitando conflitto di permessi nel sistema), le password sono svincolata dall'unicità per entrambe le entità Impiegato e UtenteRegistrato



##### ★ Mondo 1

Il seguente mondo è stato generato per mostrare la consistenza sugli inviti che un "UtenteRegistrato" può mandare a un "UtenteNonRegistrato" o a un altro "UtenteRegistrato".

Il mondo mostra che un "UtenteRegistrato" non può invitare se stesso alla sua giftcard o al suo stesso viaggio, ovvero non è possibile che l'utente "richiedente" del pacchetto corrisponda a uno degli "utentilInvitatiRegistratiGift" oppure "utentilInvitatiRegistratiViaggio" corrispondano all'utente stesso; il mondo inoltre verifica che non è possibile invitare due volte la stessa persona allo stesso viaggio o alla stessa gift.

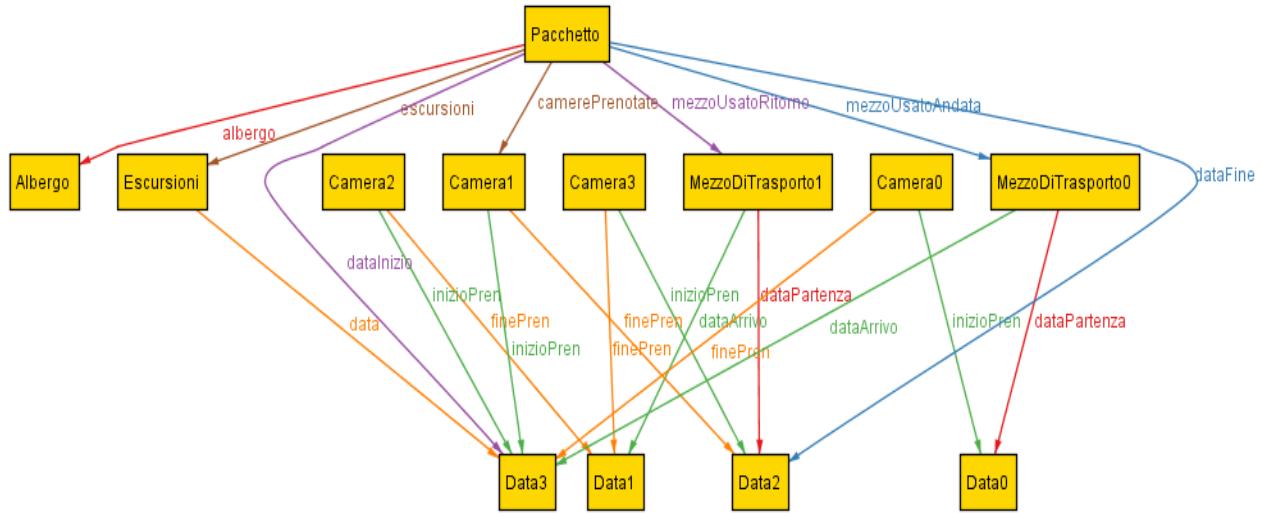


## ★ Mondo 2

Il seguente mondo è stato generato per mostrare la consistenza sulle date e orari di inizio e fine di "Pacchetto" con i relativi elementi collegati("Escursioni" e "MezzoDiTrasporto").

Come si puo notare dal grafico non è possibile che il "MezzoDiTrasporto" all'andata ,arrivi a destinazione in data differente a quella di inizio del pacchetto;stessa cosa per la ripartenza non è possibile che il mezzo di trasporto usato riparta in data differente a quella di fine del pacchetto.

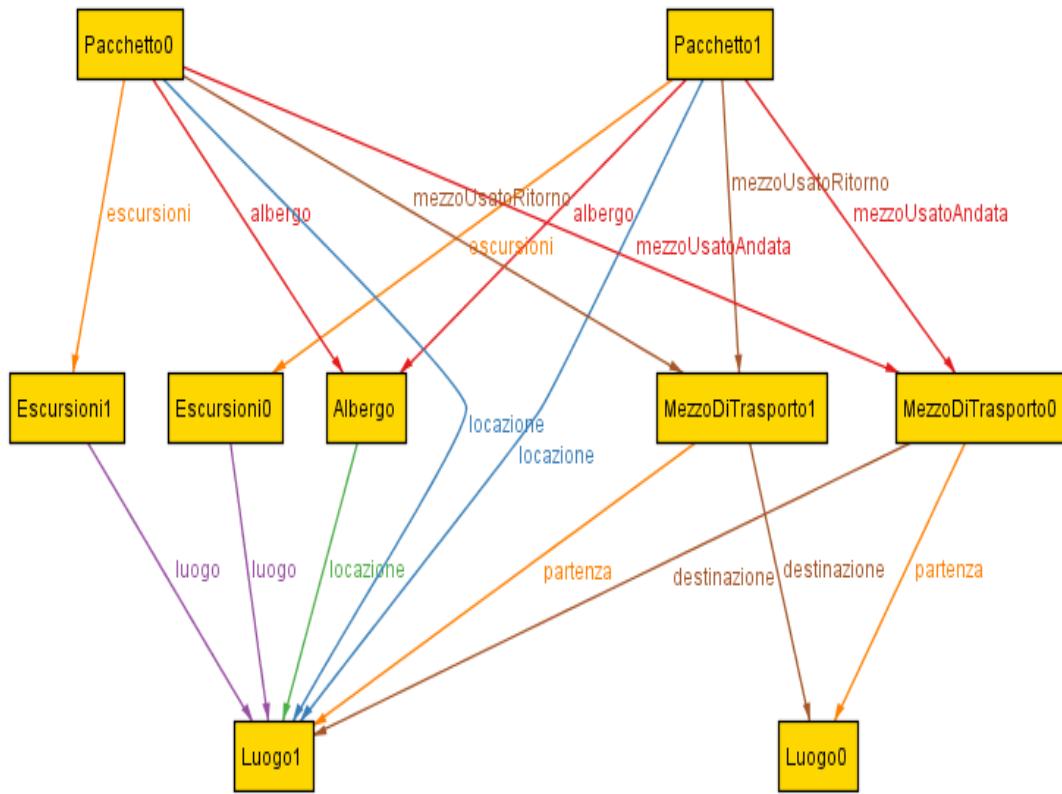
Per lo stesso motivo le “Escursioni” devono avere data concorde con quella di inizio e fine di “Pacchetto”.



### ★ Mondo 3

Il seguente mondo è stato generato per mostrare che i luoghi visitati durante un viaggio sono concordi al luogo di destinazione del volo di andata e al luogo di partenza del “Luogo” di ritorno, inoltre il luogo di destinazione del viaggio di ritorno deve essere concorde con il luogo di partenza del viaggio di andata.

Inoltre le “Escursioni” e la locazione dell “Albergo” devono trovarsi in un “Luogo” concorde con quello visitato nel “Pacchetto”.



### ★ Mondo generale

La seguente figura mostra un'istanza completa di mondo, generata attraverso l'analizzatore Alloy, senza tralasciare alcuna entità e mostrando tutte le relazioni che intercorrono tra di esse. Non è particolarmente utile all'analisi dei vincoli ma rende visibile la moltitudine di oggetti usati nel progetto.

