

# SPRINT REPORT N. 1

## **[Antonio Daniele Enterprise]**

### **Cicerone**

Versione 1.4

Data di rilascio:

30 Dicembre 2019

INGEGNERIA DEL SOFTWARE A.A. 2018-2019

**Realizzato da**

Gialluisi Antonio Daniele

<b>1. SOFTWARE SYSTEM SKELETON.....</b>	<b>4</b>
1.1 SCELTA ARCHITETTURALE.....	4
1.2 DIAGRAMMA DELLE COMPONENTI.....	5
1.2.1 <i>Descrizione delle Componenti</i> .....	6
1.2.2 <i>Descrizione delle Interfacce</i> .....	11
1.3 DIAGRAMMA DI DEPLOY.....	14
<b>2. USER STORIES.....</b>	<b>15</b>
2.1 SPRINT BACKLOG.....	15
2.2 PRODUCT SPECIFICATION.....	17
2.2.1 <i>Diagramma dei casi d'uso</i> .....	17
2.2.2 <i>Attori</i> .....	18
2.2.3 <i>Scenari</i> .....	19
2.2.4 <i>Corrispondenza tra Item funzionali e Casi d'uso</i> .....	33
2.2.5 <i>Interfacce</i> .....	35
2.2.6 <i>Qualità</i> .....	39
2.2.7 <i>Altro</i> .....	42
<b>3. PRODUCT DESIGN.....</b>	<b>43</b>
3.1 DIAGRAMMA DELLE CLASSI.....	43
3.2 SPECIFICHE DELLE CLASSI.....	44
3.2.1 <i>I package</i> .....	44
3.2.2 <i>Le classi/interfacce</i> .....	44
3.3 DIAGRAMMI DI SEQUENZA.....	50
3.3.1 <i>Diagrammi</i> .....	50
3.3.2 <i>Corrispondenza tra Casi d'uso e Diagrammi di sequenza</i> .....	61
<b>4. DATA DESIGN.....</b>	<b>63</b>
4.1 SPECIFICA DEI DATI E DEI VINCOLI INFORMATIVI.....	63
4.1.1 <i>Diagramma delle Dipendenze dei Dati</i> .....	65
4.1.2 <i>Modello del Database</i> .....	66
4.1.3 <i>Lista delle dipendenze</i> .....	67
4.1.4 <i>Dettaglio dei Dati</i> .....	67
4.2 FILE SYSTEM.....	78
4.2.1 <i>Descrizione Struttura del File System</i> .....	80
4.2.2 <i>Altro</i> .....	81
<b>5. GLOSSARIO.....</b>	<b>82</b>

5.1 ACRONIMI.....	82
5.2 DEFINIZIONI.....	82
<b>6. APPENDICE.....</b>	<b>89</b>
6.1 .....	89
6.2 .....	89

# 1. Software System Skeleton

## 1.1 Scelta architetturale

Si è deciso per questo progetto di utilizzare una variante del pattern architetturale MVC che include le nozioni di Boundary, Control, Entity.

La differenza principale sta nel fatto che nell'MVC tradizionale, le View possono richiedere dati direttamente al Model, solitamente, dopo che questi avvisano (mediante pattern Observer) di un cambio del loro stato interno.

In generale comunque, si ha che è possibile avere un'interazione tra Model<>View, View->Controller (quando è necessario leggere input dell'utente), Controller->Model (per tramutare l'input dell'utente in un'operazione da compiere, è da notare che è il Model ad eseguire l'operazione).

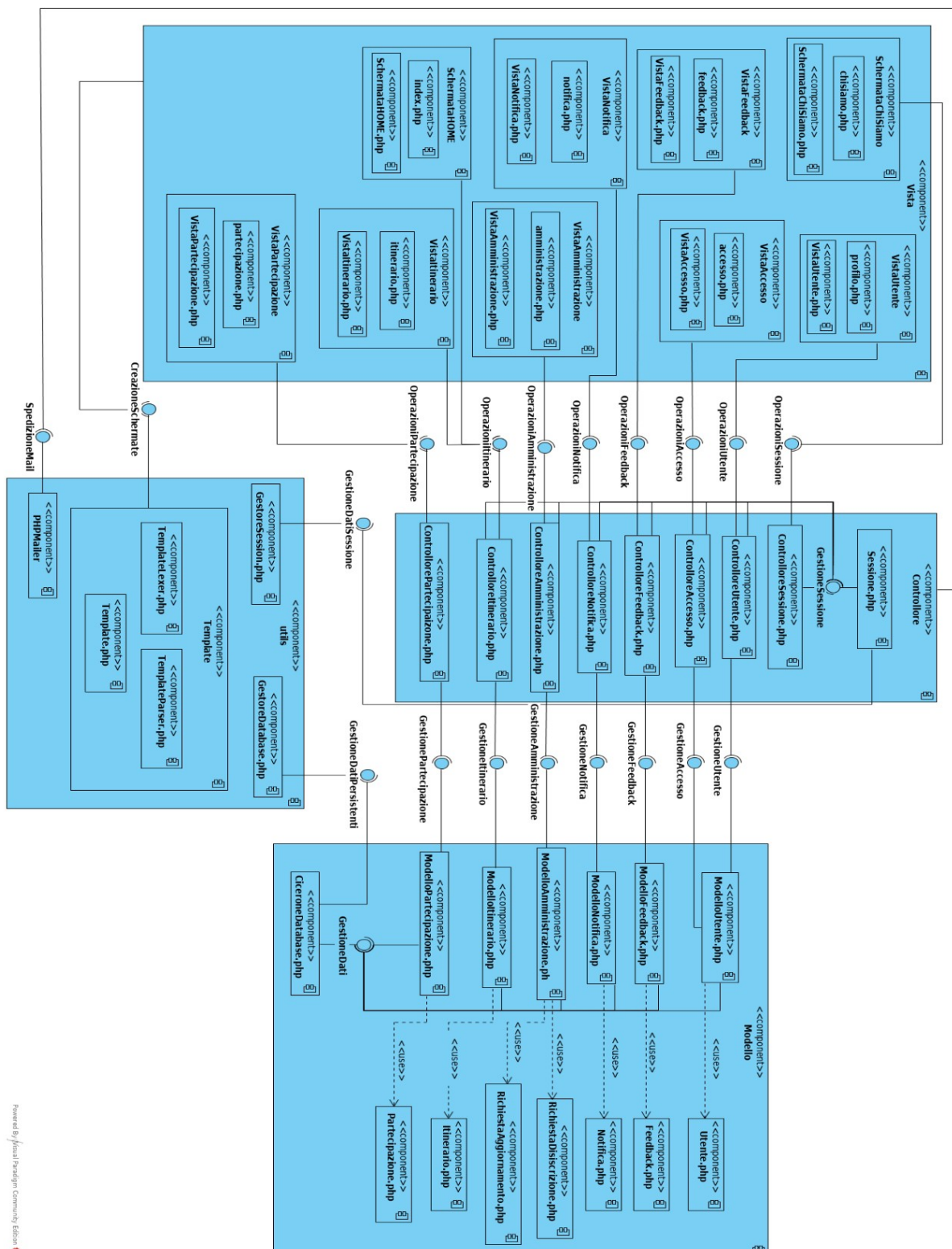
Lo scopo com'è noto, è disaccoppiare l'interfaccia dai dati affinché si possano avere più viste sugli stessi e consentire un facile aggiornamento di tutte le viste, in caso di modifica.

La variante scelta considera le sue componenti nel seguente modo:

- I Controller come classi Control
- I Model:
  - Se eseguono operazioni (incluse lettura/scrittura di dati): come classi Control
  - Se rappresentano entità informative (utenti, partecipazioni, eccetera): come classi Entity
- Le View come classi Boundary

Sebbene si mantenga l'idea dell'MVC, i Controller si trasformano da banali elaboratori di input a veri e propri punti di incontro tra Model e View, rispettando dunque le relazioni possibili tra Boundary<>Control<>Entity.

## 1.2 Diagramma delle Componenti



### 1.2.1 Descrizione delle Componenti

- Vista:  
È la macro-componente che consente le interazioni tra attori e sistema.
- Modello:  
È la macro-componente che esegue la logica di business (ergo, realizza i casi d'uso) e il conseguente caricamento/salvataggio dei dati persistenti sul e dal DB.
- Controllore:  
È la macro-componente che consente l'esecuzione dei vari casi d'uso, in altri termini fa da tramite tra la componente Modello e la componente Vista.
- utils:  
È la macro-componente che fa da supporto alle altre tre, fornendo varie funzionalità.
- VistaAccesso:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di accesso, registrazione, disconnessione, attivazione e recupero dell'accesso.  
È composta dai file `accesso.php` e `VistaAccesso.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaAccesso` che verrà poi utilizzata.
- VistaUtente:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di visualizzazione e modifica dei profili associati agli utenti.  
È composta dai file `profilo.php` e `VistaUtente.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaUtente` che verrà poi utilizzata.
- VistaItinerario:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di visualizzazione/modifica di itinerari.  
È composta dai file `itinerario.php` e `VistaItinerario.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaItinerario` che verrà poi utilizzata.

- VistaAmministrazione:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di creazione e visualizzazione di richieste d'amministrazione. È composta dai file `amministrazione.php` e `VistaAmministrazione.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaAmministrazione` che verrà poi utilizzata.
- VistaNotifica:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di visualizzazione/rimozione di notifiche e cambio delle impostazioni che le riguardano. È composta dai file `notifica.php` e `VistaNotifica.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaNotifica` che verrà poi utilizzata.
- VistaPartecipazione:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di invio/accordo/annullamento/declino di richieste di partecipazione agli itinerari. È composta dai file `partecipazione.php` e `VistaPartecipazione.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaPartecipazione` che verrà poi utilizzata.
- VistaFeedback:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare tutte le interfacce utili ad effettuare operazioni di visualizzazione/modifica di itinerari ed invio di richieste di partecipazione. È composta dai file `feedback.php` e `VistaFeedback.php`. Il primo è l'entrypoint che serve a comunicare l'intenzione di utilizzare questo tipo di Vista, il secondo contiene la classe di tipo *boundary* `VistaFeedback` che verrà poi utilizzata.
- SchermataHOME:  
È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare l'interfaccia utile a mostrare la home del sistema, che consiste nella visualizzazione degli ultimi itinerari organizzati in ordine di tempo. È composta dai file `index.php` e `SchermataHOME.php`.

Il primo è l'entrypoint, il secondo contiene la classe *boundary* SchermataHOME che si occuperà di mostrare la lista di itinerari.

- SchermataChiSiamo:

È la componente che consente l'interfacciamento dell'utente al sistema, occupandosi di mostrare l'interfaccia utile a mostrare le informazioni del sistema.  
È composta dai file *chisiamo.php* e *SchermataChiSiamo.php*.  
Il primo è l'entrypoint, il secondo contiene la classe *boundary* SchermataChiSiamo che si occuperà di mostrare le informazioni.

- Sessione.php:

È la componente che si occuperà di memorizzare e leggere i dati di sessione (quelli che consentono di comprendere se un utente ha effettuato l'accesso, se ci sono stati errori e così via).  
È un file che contiene la classe *control* Sessione.

- ControlloreSessione.php:

È la componente che consentirà alle Viste che lo richiederanno, la sola lettura dei dati di sessione.  
È un file che contiene la classe *control* ControlloreSessione.

- ControlloreAccesso.php:

È la componente che fa da intermediario tra la componente ModelloUtente.php e la componente VistaAccesso.  
Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloUtente.php in base ai parametri ottenuti da VistaAccesso, e viceversa, restituendo a VistaAccesso il risultato dei servizi richiesti a ModelloUtente.php.  
Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore.  
È un file che contiene la classe *control* ControlloreAccesso.

- ControlloreUtente.php:

È la componente che fa da intermediario tra la componente ModelloUtente.php e la componente VistaUtente.  
Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloUtente.php in base ai parametri ottenuti da VistaUtente, e viceversa, restituendo a VistaUtente il risultato dei servizi richiesti a ModelloUtente.php.  
Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore.  
È un file che contiene la classe *control* ControlloreUtente.

- ControlloreItinerario.php:

È la componente che fa da intermediario tra la componente ModelloItinerario.php e la componente VistaItinerario.  
Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloItinerario.php in base ai parametri ottenuti da VistaItinerario, e



viceversa, restituendo a VistaItinerario il risultato dei servizi richiesti a ModelloItinerario.php.

Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore. È un file che contiene la classe *control* *ControlloreItinerario*.

- *ControlloreAmministrazione.php*:

È la componente che fa da intermediario tra la componente ModelloAmministrazione.php e la componente VistaAmministrazione. Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloAmministrazione.php in base ai parametri ottenuti da VistaAmministrazione, e viceversa, restituendo a VistaAmministrazione il risultato dei servizi richiesti a ModelloAmministrazione.php. Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore. È un file che contiene la classe *control* *ControlloreAmministrazione*.

- *ControlloreNotifica.php*:

È la componente che fa da intermediario tra la componente ModelloNotifica.php e la componente VistaNotifica. Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloNotifica.php in base ai parametri ottenuti da VistaNotifica, e viceversa, restituendo a VistaNotifica il risultato dei servizi richiesti a ModelloNotifica.php.

Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore. È un file che contiene la classe *control* *ControlloreNotifica*.

- *ControllorePartecipazione.php*:

È la componente che fa da intermediario tra la componente ModelloPartecipazione.php e la componente VistaPartecipazione. Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloPartecipazione.php in base ai parametri ottenuti da VistaPartecipazione, e viceversa, restituendo a VistaPartecipazione il risultato dei servizi richiesti a ModelloPartecipazione.php. Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore. È un file che contiene la classe *control* *ControllorePartecipazione*.

- *ControlloreFeedback.php*:

È la componente che fa da intermediario tra la componente ModelloFeedback.php e la componente VistaFeedback. Si occuperà di gestire la comunicazione, chiedendo servizi a ModelloFeedback.php in base ai parametri ottenuti da VistaFeedback, e viceversa, restituendo a VistaFeedback il risultato dei servizi richiesti a ModelloFeedback.php.

Tra i compiti assolti abbiamo anche la restituzione di messaggi d'errore. È un file che contiene la classe *control* *ControlloreFeedback*.

- *ModelloUtente.php*:

È la componente che contiene la logica di business e che gestisce i dati

di business riguardanti gli utenti. Tra le operazioni che può svolgere, abbiamo l'accesso al sistema, la registrazione di un nuovo utente e la gestione dei profili. Utilizza principalmente il componente Utente.php. È un file che contiene la classe *control* ModelloUtente.

- ModelloItinerario.php:

È la componente che contiene la logica di business e che gestisce i dati di business riguardanti gli itinerari. Tra le operazioni che può svolgere, abbiamo la ricerca di itinerari e la creazione/modifica/restituzione degli stessi. Utilizza principalmente la componente Itinerario.php. È un file che contiene la classe *control* ModelloItinerario.

- ModelloAmministrazione.php:

È la componente che contiene la logica di business e che gestisce i dati di business riguardanti le richieste d'amministrazione. Tra le operazioni che può svolgere, abbiamo la creazione/accordo/restituzione delle richieste di disiscrizione e d'aggiornamento. Utilizza principalmente le componenti RichiestaDisiscrizione.php e RichiestaAggiornamento.php. È un file che contiene la classe *control* ModelloAmministrazione.

- ModelloNotifica.php:

È la componente che contiene la logica di business e che gestisce i dati di business riguardanti le notifiche. Tra le operazioni che può svolgere, abbiamo la restituzione delle notifiche e la loro rimozione. Utilizza principalmente la componente Notifica.php. È un file che contiene la classe *control* ModelloNotifica.

- ModelloPartecipazione.php:

È la componente che contiene la logica di business e che gestisce i dati di business riguardanti le richieste di partecipazione agli itinerari. Tra le operazioni che può svolgere, abbiamo la restituzione delle richieste di partecipazione e l'invio/accordo/annullamento delle stesse. Utilizza principalmente la componente Partecipazione.php. È un file che contiene la classe *control* ModelloPartecipazione.

- ModelloFeedback.php:

È la componente che contiene la logica di business e che gestisce i dati di business riguardanti i feedback. Tra le operazioni che può svolgere, abbiamo la creazione/restituzione dei feedback. Utilizza principalmente la componente Feedback.php. È un file che contiene la classe *control* ModelloFeedback.

- *Utente.php*:  
È la componente che rappresenta i dati persistenti inerenti gli utenti del sistema.  
È un file che contiene la classe *entity* Utente.
- *Feedback.php*:  
È la componente che rappresenta i dati persistenti inerenti gli i  
feedback rilasciati.  
È un file che contiene la classe *entity* Feedback.
- *Notifica.php*:  
È la componente che rappresenta i dati persistenti inerenti le notifiche.  
È un file che contiene la classe *entity* Notifica.
- *RichiestaDisiscrizione.php*:  
È la componente che rappresenta i dati persistenti inerenti gli utenti del sistema.  
È un file che contiene la classe *entity* RichiestaDisiscrizione.
- *RichiestaAggiornamento.php*:  
È la componente che rappresenta i dati persistenti inerenti le richieste  
d'aggiornamento inviate dai Globetrotter.  
È un file che contiene la classe *entity* RichiestaAggiornamento.
- *Itinerario.php*:  
È la componente che rappresenta i dati persistenti inerenti gli itinerari.  
È un file che contiene la classe *entity* Itinerario.
- *Partecipazione.php*:  
È la componente che rappresenta i dati persistenti inerenti le richieste  
di partecipazione inviate dai fruitori.  
È un file che contiene la classe *entity* Partecipazione.
- *GestoreSession.php*:  
È la componente d'ausilio che consente in maniera semplice di gestire i  
dati di sessione a basso livello (lettura, scrittura, modifica e rimozione).
- *GestoreDatabase.php*:  
È la componente d'ausilio che consente in maniera semplice di gestire i  
dati di un DB a basso livello.
- *PHPMailer*:  
È la componente di tipo grey-box che consente in maniera semplice di  
spedire email sfruttando server SMTP esterni.
- *Template*:  
È la componente d'ausilio che consente in maniera semplice di costruire  
schermate "a modello" utilizzando un apposito linguaggio.  
Per implementare le sue funzionalità, si serve delle sotto-componenti  
*TemplateLexer.php*, *TemplateParser.php* e *Template.php*.

Quest'ultima contiene la classe che realizza le funzionalità per creare le schermate.

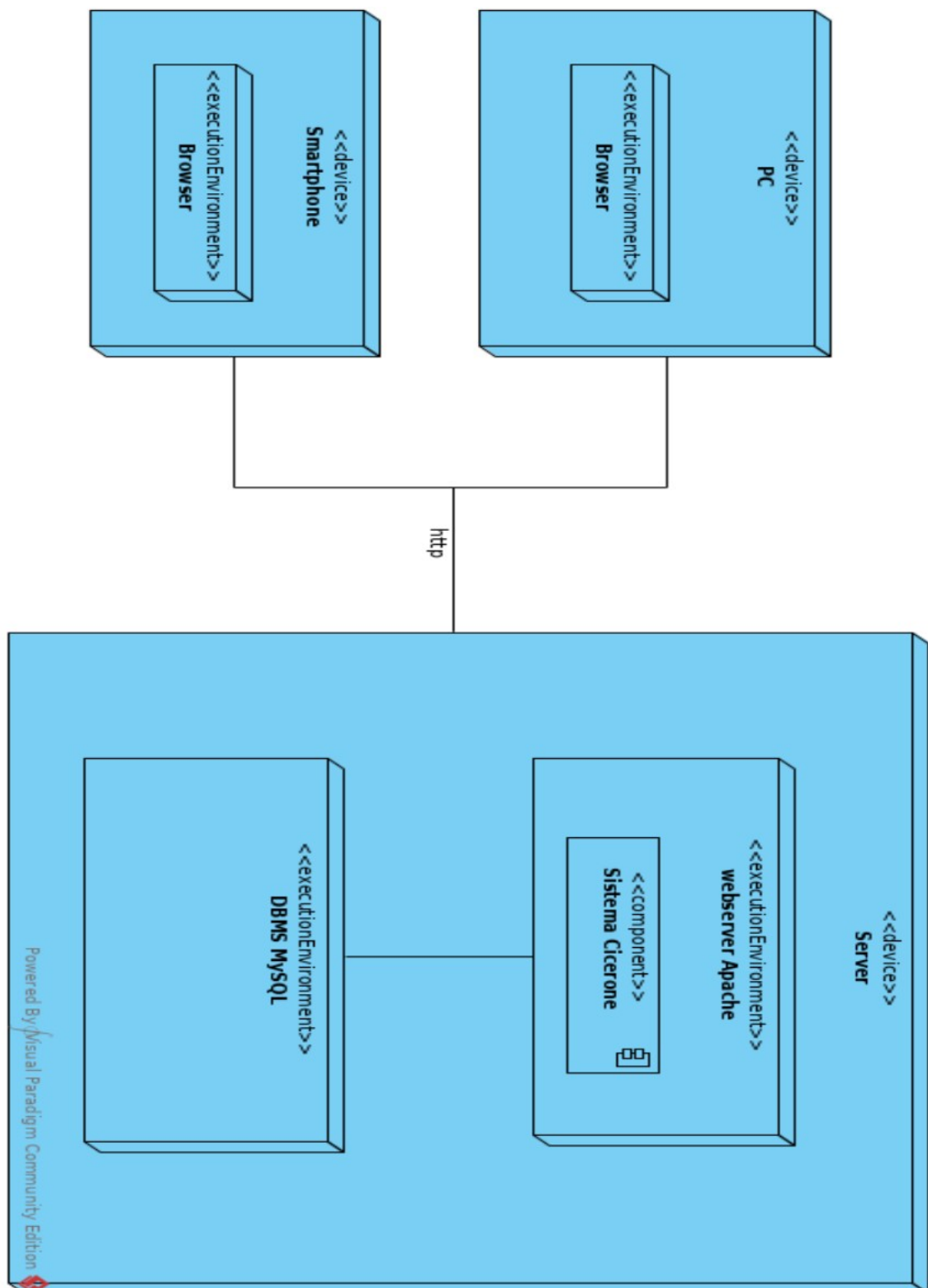
### 1.2.2 Descrizione delle Interfacce

- *GestioneAccesso* (fornita dalla componente *ModelloUtente.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano l'accesso degli utenti registrati, come ad esempio, l'accesso, registrazione, recupero accesso e attivazione degli utenti.
- *GestioneUtente* (fornita dalla componente *ModelloUtente.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano i profili degli utenti registrati, come ad esempio, la modifica del profilo o delle anagrafiche.
- *GestioneItinerario* (fornita dalla componente *ModelloItinerario.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano gli itinerari, come ad esempio, la loro creazione, modifica e rimozione.
- *GestioneAmministrazione* (fornita dalla componente *ModelloAmministrazione.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano le richieste d'amministrazione, come ad esempio, l'invio delle richieste di disiscrizione e d'aggiornamento.
- *GestioneNotifica* (fornita dalla componente *ModelloNotifica.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano le notifiche, come ad esempio, la rimozione e la restituzione delle stesse per essere visualizzate.
- *GestionePartecipazione* (fornita dalla componente *ModelloPartecipazione.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano le richieste di partecipazione, come ad esempio, l'accordo e la restituzione delle stesse per essere visualizzate.
- *GestioneFeedback* (fornita dalla componente *ControlloreFeedback.php*):  
Espone le funzioni che permettono di eseguire le operazioni di business che riguardano i feedback, come ad esempio, il rilascio di un feedback organizzatore-partecipante o partecipante-organizzatore.
- *OperazioniSessione* (fornita dalla componente *ControlloreSessione.php*):  
Espone le funzionalità che consentono alle componenti (viste) che l'utilizzano, di poter richiedere la gestione dei dati di sessione.

- OperazioniUtente (fornita dalla componente ControlloreUtente.php):  
Espone le funzionalità che consentono alla componente VistaUtente.php di poter richiedere l'esecuzione di operazioni inerenti gli utenti.
- OperazioniAccesso (fornita dalla componente ControlloreUtente.php):  
Espone le funzionalità che consentono alla componente VistaAccesso.php di poter richiedere l'esecuzione di operazioni inerenti l'accesso degli utenti.
- OperazioniItinerario (fornita dalla componente ControlloreItinerario.php):  
Espone le funzionalità che consentono alla componente VistaItinerario.php di poter richiedere l'esecuzione di operazioni inerenti gli itinerari.
- OperazioniAmministrazione (fornita dalla componente ControlloreAmministrazione.php):  
Espone le funzionalità che consentono alla componente VistaAmministrazione.php di poter richiedere l'esecuzione di operazioni inerenti le richieste d'amministrazione (disiscrizione e aggiornamento).
- OperazioniNotifica (fornita dalla componente ControlloreNotifica.php):  
Espone le funzionalità che consentono alla componente VistaNotifica.php di poter richiedere l'esecuzione di operazioni inerenti le notifiche.
- OperazioniPartecipazione (fornita dalla componente ControllorePartecipazione.php):  
Espone le funzionalità che consentono alla componente VistaPartecipazione.php di poter richiedere l'esecuzione di operazioni inerenti le richieste di partecipazione.
- OperazioniFeedback (fornita dalla componente ControlloreFeedback.php):  
Espone le funzionalità che consentono alla componente VistaFeedback.php di poter richiedere l'esecuzione di operazioni inerenti i feedback.
- GestioneSessione (fornita dalla componente Sessione.php):  
Espone le funzionalità che consentono alle componenti che l'utilizzano di richiedere la gestione dei dati di sessione.
- GestioneDatiSessione (fornita dalla componente GestoreSession.php):  
Espone le funzionalità che consentono alle componenti che l'utilizzano di poter gestire i dati di sessione a basso livello (lettura/creazione/rimozione).
- GestioneDatiPersistenti (fornita dalla componente GestoreDatabase.php):  
Espone le funzionalità che consentono alle componenti che l'utilizzano di poter gestire i dati di un DB a basso livello.

- CreazioneSchermate (fornita dalla componente Template):  
Espone le funzionalità che consentono la creazione di schermate per le varie viste.
- SpedizioneMail (fornita dalla componente PHPMailer):  
Espone le funzionalità che consentono la spedizione di email.

### 1.3 Diagramma di Deploy



## 2. User Stories

### 2.1 Sprint Backlog

Tabella di riepilogo che indica, per ognuno degli Sprint successivo allo Sprint n.0, la lista degli item del Product Backlog che verranno implementati nell'ambito dello sprint corrente unitamente ad una descrizione esplicativa.

Per semplificare l'esposizione e salvaguardare la tracciabilità tra semilavorati si è proceduto alle seguenti assunzioni:

- Una User Story è l'insieme di uno o più item del Product Backlog implementato all'interno di uno sprint.
- Lo Sprint Backlog relativo allo sprint corrente contiene pertanto l'insieme degli item del Product Backlog ricompresi nella User Story in corso di implementazione.
- Gli Item funzionali ricompresi nelle User Story dovranno essere tracciabili uno a uno, auspicabilmente seppur non necessariamente, con i casi d'uso.
- Ad ogni caso d'uso dovrà essere associato uno scenario di base più gli eventuali scenari alternativi.
- Ad ogni caso d'uso dovrà essere associato un diagramma di sequenza.

Con le assunzioni sopra si renderà quindi tracciabile la catena: Product Backlog Items -> User Story -> Casi d'Uso -> Scenari -> Diagrammi di Sequenza.

<i>Numero Sprint</i>	<i>Codice Item</i>	<i>Descrizione</i>
1	<u>IF GN11</u>	Registrazione utente
1	<u>IF GN12</u>	Effettua accesso
1	<u>IF GN13</u>	Recupero dell'accesso
1	<u>IF GN14</u>	Disconnessione utente
1	<u>IF GN15</u>	Attivazione dell'utente
1	<u>IF GN21</u>	Visualizzazione profilo
1	<u>IF GN22</u>	Modifica profilo
1	<u>IF GN23</u>	Modifica anagrafiche
1	<u>IF GN24</u>	Invio richiesta disiscrizione
1	<u>IF GN25</u>	Disiscrizione fruitore



1	<u>IF GN26</u>	Visualizza lista richieste amministrazione
?	<u>IF GN31</u>	Visualizzazione notifica
?	<u>IF GN32</u>	Rimozione notifica
?	<u>IF GN33</u>	Cambio preferenze di notifica
?	<u>IF 1111</u>	Creazione nuovo itinerario
?	<u>IF 1112</u>	Modifica itinerario
?	<u>IF 1113</u>	Rimozione itinerario
?	<u>IF 1121</u>	Visualizzazione richieste di partecipazione
?	<u>IF 1122</u>	Accordo richiesta di partecipazione
?	<u>IF 1123</u>	Annullamento richiesta di partecipazione
?	<u>IF 1124</u>	Declino richiesta di partecipazione
1	<u>IF 1131</u>	Invio richiesta di aggiornamento
1	<u>IF 1132</u>	Trasformazione in QuasiCicerone
1	<u>IF 1133</u>	Transizione a Cicerone confermata
?	<u>IF 2111</u>	Ricerca itinerari
?	<u>IF 2112</u>	Visualizzazione itinerario
?	<u>IF 2113</u>	Visualizzazione lista itinerari fruitore
?	<u>IF 2211</u>	Invio richiesta partecipazione
?	<u>IF 2212</u>	Invio richiesta annullamento
?	<u>IF 2221</u>	Visualizzazione feedback fruitore

?	<u>IF 2222</u>	Rilascio feedback partecipante-organizzatore
?	<u>IF 2223</u>	Rilascio feedback organizzatore-partecipante
?	<u>IF 2224</u>	Visualizza feedback

## 2.2 Product Specification

### 2.2.1 Diagramma dei casi d'uso



### 2.2.2 Attori

ID	<b>A_1</b>
Nome	Utente
Eredita da	
Descrizione	È un utente generico del sistema, esso può eseguire i compiti comuni a tutti i tipo di utenti, registrati e non.

ID	<b>A_2</b>
Nome	Ospite
Eredita da	Utente
Descrizione	È l'utente ospite, ovvero, quello che non è registrato o non ha effettuato l'accesso al sistema.

ID	<b>A_3</b>
Nome	Globetrotter
Eredita da	Utente
Descrizione	È l'utente fruitore che può richiedere la partecipazione agli itinerari.

ID	<b>A_4</b>
Nome	Cicerone
Eredita da	Utente
Descrizione	È l'utente fruitore che può

	organizzare gli itinerari e accordare le richieste di partecipazione di altri fruitori.
--	---

ID	<b>A_5</b>
Nome	Amministratore
Eredita da	Utente
Descrizione	È l'utente che gestisce le richieste d'amministrazione inviate dai fruitori.

ID	<b>A_6</b>
Nome	QuasiCicerone
Eredita da	Utente
Descrizione	È un ex-Globetrotter che deve confermare la transizione a Cicerone.

### 2.2.3 Scenari

<b>ID <u>CU 1</u></b> <b>Caso d'uso <u>Registrazione Utente</u></b>	
Descrizione	Il sistema provvede alla creazione di un nuovo utente registrato e del relativo profilo associato.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite NON possiede delle credenziali per accedere
Sequenza eventi	<u>1</u> L'ospite inserisce i dati richiesti (email, nome utente e password). <u>2</u> L'ospite chiede al sistema di procedere con la registrazione del nuovo utente.

	3 Il sistema provvede alla registrazione del nuovo utente.
Postcondizioni	L'ospite troverà nella casella di posta dell'indirizzo email inserito, un'email che conterrà un codice d'attivazione (questo servirà per attivare l'utente registrato).
Sequenze alternative	<u>ALT 1.1</u>
<b>ID <u>ALT 1.1</u></b>	
<b>Scenario alternativo <u>RegistraUtente:DatiInvalidi</u></b>	
Descrizione	Il sistema avvisa l'ospite dell'invalidità dei dati.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite deve aver inserito dei dati non validi (ad esempio, email malformata) OPPURE L'ospite deve aver inserito un nome utente già utilizzato OPPURE L'ospite deve aver inserito un'email già utilizzata.
Passo d'esecuzione	Dopo il passo 2 di <u>CU 1</u>
Sequenza eventi	1 Il sistema avvisa l'ospite che i dati inseriti non sono validi con un messaggio.
Postcondizioni	Il sistema NON effettuerà la creazione del nuovo utente registrato.

<b>ID <u>CU 2</u></b>	
<b>Caso d'uso <u>EffettuaAccesso</u></b>	
Descrizione	L'ospite effettua l'accesso al sistema.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'utente registrato con cui l'ospite si

	autenticherà deve essere nello stato <i>attivato o recuperando</i> .
Sequenza eventi	<p><u>1</u> L'ospite inserisce i dati richiesti (nome utente e password).</p> <p><u>2</u> L'ospite chiede al sistema di procedere con l'accesso.</p> <p><u>3</u> Il sistema verifica l'esistenza del nome utente e la corrispondenza della password.</p> <p><u>4</u> Il sistema accorda l'accesso.</p> <p><u>5</u> Se l'utente registrato è nello stato <i>recuperando</i>:</p> <p><u>5.1</u> Il sistema lo imposta ad <i>attivato</i>.</p> <p><u>5.2</u> Il sistema rimuove il codice d'attivazione associato.</p>
Postcondizioni	<p>L'ospite effettuerà l'accesso al sistema.</p> <p>L'ospite potrà compiere tutte le operazioni consentite in base al tipo di utente registrato a cui risulta appartenere.</p>
Sequenze alternative	<u>ALT 2.1</u>

ID <u>ALT 2.1</u>	
Scenario alternativo <u>EffettuaAccesso:AccessoNegato</u>	
Descrizione	Il sistema nega l'accesso all'ospite.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite ha inserito delle credenziali non corrette (il nome utente non è stato trovato oppure la password non corrisponde).
Passo d'esecuzione	Dopo il passo <u>3</u> di <u>CU 2</u>
Sequenza eventi	<u>1</u> Il sistema avvisa l'ospite del problema con un messaggio.
Postcondizioni	L'ospite NON effettua l'accesso al sistema.

<b>ID <u>CU 3</u></b> <b>Caso d'uso <u>RecuperaAccesso</u></b>	
Descrizione	L'ospite chiede al sistema di effettuare il recupero dell'accesso.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite non ricorda le credenziali per l'accesso.
Sequenza eventi	<u>1</u> L'ospite inserisce il suo indirizzo email. <u>2</u> Se l'indirizzo email inserito appartiene ad uno degli utenti registrati: <u>2.1</u> Il sistema imposta lo stato dell'utente registrato a <i>recuperando</i> . <u>2.2</u> Il sistema invia un'email all'indirizzo specificato con un codice d'attivazione utile per riattivare l'utente.
Postcondizioni	L'ospite riceve un email con il codice d'attivazione utile per effettuare la reimpostazione della password.
Sequenze alternative	

<b>ID <u>CU 4</u></b> <b>Caso d'uso <u>DisconnettiUtente</u></b>	
Descrizione	L'utente registrato effettua la disconnessione dal sistema.
Attori primari	Globetrotter, QuasiCicerone, Cicerone, Amministratore
Attori secondari	
Precondizioni	L'utente registrato deve aver effettuato l'accesso al sistema.
Sequenza eventi	<u>1</u> L'utente registrato chiede al sistema di effettuare la disconnessione.



	<u>2</u> Il sistema disconnette l'utente registrato.
Postcondizioni	L'utente registrato torna ad essere un utente ospite.
Sequenze alternative	

ID <u>CU 5</u> Caso d'uso <u>AttivaUtente</u>	
Descrizione	L'ospite chiede al sistema di effettuare l'attivazione dell'utente registrato con cui intende autenticarsi.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite possiede il codice d'attivazione che identifica il suo utente registrato  E lo stato dell'utente registrato risulta essere <i>inserito</i> oppure <i>recuperando</i> .
Sequenza eventi	<u>1</u> L'ospite fornisce al sistema il codice d'attivazione. <u>2</u> Il sistema provvede all'attivazione dell'utente registrato.
Postcondizioni	Lo stato dell'utente registrato verrà impostato ad <i>attivato</i> .
Sequenze alternative	<u>ALT 5.1</u> , <u>ALT 5.2</u> , <u>ALT 5.3</u>

ID <u>ALT 5.1</u> Caso d'uso <u>AttivaUtente:CodiceUnivocoNonValido</u>	
Descrizione	Il sistema nega l'attivazione poiché il codice non corrisponde ad alcun utente registrato.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'ospite invia un codice d'attivazione che

	non è associato ad alcun utente registrato che si trova negli stati <i>inserito</i> o <i>recuperando</i> .
Passo d'esecuzione	Dopo il passo <u>1</u> di <u>CU 5</u>
Sequenza eventi	<u>1</u> Il sistema ignora il problema <u>2</u> L'ospite può continuare ad interagire col sistema da ospite.
Postcondizioni	L'utente non viene informato del problema, può continuare ad essere ospite.

ID <u>ALT 5.2</u>	
Caso d'uso <u>AttivaUtente:Attivazione</u>	
Descrizione	L'utente registrato deve essere attivato per poter effettuare l'accesso.
Attori primari	Ospite
Attori secondari	
Precondizioni	L'utente registrato è nello stato <i>inserito</i> .
Passo d'esecuzione	Dopo il passo <u>1</u> di <u>CU 5</u>
Sequenza eventi	<u>1</u> Il sistema rimuove il codice d'attivazione. <u>2</u> Il sistema imposta lo stato dell'utente registrato ad <i>attivato</i> . <u>3</u> Il sistema mostra un messaggio riguardante l'accaduto.
Postcondizioni	Risulta possibile accedere al sistema con l'utente registrato.

ID <u>ALT 5.3</u>	
Caso d'uso <u>AttivaUtente:Recupero</u>	
Descrizione	L'utente registrato ha bisogno di essere recuperato. Il sistema consentirà di reimpostare la password.

Attori primari	Ospite
Attori secondari	
Precondizioni	L'utente registrato è nello stato <i>recuperando</i> .
Passo d'esecuzione	Dopo il passo <u>1</u> di <u>CU 5</u>
Sequenza eventi	<u>1</u> Il sistema permette il reinserimento della password. <u>2</u> L'ospite inserisce la password. <u>3</u> Fintantoché la verifica della password fallisce: <u>3.1</u> Il sistema invita l'ospite a riprovare l'inserimento. <u>4</u> Il sistema memorizza la nuova password. <u>5</u> Il sistema rimuove il codice d'attivazione. <u>6</u> Il sistema imposta lo stato dell'utente registrato ad <i>attivato</i> . <u>7</u> Il sistema mostra un messaggio inerente l'accaduto.
Postcondizioni	L'utente registrato può essere riutilizzato per accedere.

ID <u>CU 6</u>	
Caso d'uso <u>VisualizzaProfilo</u>	
Descrizione	L'utente visualizza il profilo di un utente registrato (oppure il proprio).
Attori primari	Utente
Attori secondari	
Precondizioni	
Sequenza eventi	<p><u>1</u> L'utente chiede al sistema di visualizzare il profilo di un particolare utente registrato.</p> <p><u>2</u> Il sistema mostra il profilo associato all'utente registrato scelto dall'utente.</p> <p><u>3</u> Se l'utente è un Cicerone che ha chiesto la visualizzazione del proprio profilo:</p> <p><u>3.1</u> Verrà mostrata anche l'anagrafica ad esso associata.</p>
Postcondizioni	All'utente verrà mostrato il profilo di un utente registrato (oppure il proprio).
Sequenze alternative	

ID <u>CU 7</u>	
Caso d'uso <u>ModificaProfilo</u>	
Descrizione	Un utente registrato modifica le informazioni associate al proprio profilo.
Attori primari	Globetrotter, QuasiCicerone, Cicerone, Amministratore
Attori secondari	
Precondizioni	Bisogna aver effettuato l'accesso.
Sequenza eventi	<p><u>1</u> L'utente registrato chiede al sistema di poter modificare il proprio profilo.</p> <p><u>2</u> Il sistema permette la modifica del profilo.</p> <p><u>3</u> L'utente effettua le modifiche.</p> <p><u>4</u> L'utente chiede al sistema di salvare i</p>

	dati. <u>5</u> Il sistema effettua il salvataggio dei dati.
Postcondizioni	Le informazioni di profilo saranno aggiornate in base alle modifiche apportate.
Sequenze alternative	<u>ALT 7.1</u>

ID <u>ALT 7.1</u> <b>Scenario alternativo</b> <u>ModificaProfilo:ModificaPasswordFallita</u>	
Descrizione	Un utente registrato modifica, tra le informazioni associate al suo profilo, la password, tuttavia la corrispondenza tra password fallisce.
Attori primari	Globetrotter, QuasiCierone, Cicerone, Amministratore
Attori secondari	
Precondizioni	L'utente registrato ha modificato la password del proprio profilo, tuttavia, la vecchia password non corrisponde oppure la verifica della nuova password fallisce.
Passo d'esecuzione	Dopo il passo <u>4</u> di <u>CU 7</u>
Sequenza eventi	<u>1</u> Il sistema avvisa l'utente di quel che è successo con un messaggio.
Postcondizioni	L'utente registrato può continuare a modificare il proprio profilo.

ID <u>CU 8</u> <b>Caso d'uso</b> <u>ModificaAnagrafica</u>	
Descrizione	Un Cicerone modifica le informazioni anagrafiche associate al proprio profilo.
Attori primari	Cicerone
Attori secondari	

Precondizioni	Bisogna aver effettuato l'accesso.
Sequenza eventi	<u>1</u> Il Cicerone chiede al sistema di poter modificare le informazioni anagrafiche. <u>2</u> Il sistema permette la modifica dell'anagrafica. <u>3</u> Il Cicerone effettua le modifiche. <u>4</u> Il Cicerone chiede al sistema di salvare i dati. <u>5</u> Il sistema effettua il salvataggio dei dati.
Postcondizioni	Le informazioni sull'anagrafica saranno aggiornate in base alle modifiche apportate.
Sequenze alternative	

<b>ID <u>CU 9</u></b> <b>Caso d'uso <u>InviaRichiestaDisiscrizione</u></b>	
Descrizione	Un fruitore invia una richiesta di disiscrizione, affinché gli amministratori possano disiscriverlo dal sistema.
Attori primari	Globetrotter, QuasiCicerone, Cicerone
Attori secondari	
Precondizioni	Bisogna aver effettuato l'accesso.
Sequenza eventi	<u>1</u> Il fruitore chiede al sistema la creazione di una richiesta di disiscrizione. <u>2</u> Il sistema permette la creazione di questo tipo di richiesta. <u>3</u> Il fruitore inserisce i dati necessari. <u>4</u> Il fruitore chiede al sistema l'invio della richiesta. <u>5</u> Il sistema effettua l'invio della richiesta di disiscrizione.
Postcondizioni	Viene creata una nuova richiesta di disiscrizione.
Sequenze alternative	

<b>ID <u>CU 10</u></b> <b>Caso d'uso <u>DisiscriviFruitore</u></b>	
Descrizione	Un amministratore provvede alla disiscrizione richiesta da un fruitore.
Attori primari	Amministratore
Attori secondari	
Precondizioni	Dev'essere già stata inviata ALMENO una richiesta di disiscrizione.
Sequenza eventi	<u>1</u> L'amministratore chiede di visualizzare una particolare richiesta di disiscrizione. <u>2</u> Il sistema mostra i dettagli associati alla richiesta di disiscrizione. <u>3</u> Se l'amministratore accorda la richiesta di disiscrizione: <u>3.1</u> Il sistema provvederà alla disiscrizione del fruitore che l'ha richiesta. <u>3.2</u> Il sistema invierà una notifica via email di ciò che è accaduto.
Postcondizioni	Se è stato scelto di disiscrivere il fruitore, il fruitore verrà disiscritto dal sistema, in caso contrario non ci saranno conseguenze.
Sequenze alternative	

<b>ID <u>CU 11</u></b> <b>Caso d'uso <u>VisualizzaRichiesteAmministrazione</u></b>	
Descrizione	Un amministratore visualizza le richieste d'amministrazione disponibili.
Attori primari	Amministratore
Attori secondari	
Precondizioni	L'amministratore deve aver effettuato l'accesso al sistema.
Sequenza eventi	<u>1</u> L'amministratore chiede al sistema di visualizzare le richieste d'amministrazione disponibili.

	<p><u>2</u> Il sistema mostra le richieste d'amministrazione.</p> <p><u>3</u> L'amministratore in base alle sue preferenze, chiede al sistema di mostrare la lista delle richieste di disiscrizione, o la lista delle richieste di aggiornamento.</p> <p><u>4</u> Il sistema mostra la lista di richieste d'amministrazione in base alla scelta dell'amministratore.</p>
Postcondizioni	L'amministratore visualizza le richieste di disiscrizione o di aggiornamento ricevute dai fruitori, in base alla sua scelta.
Sequenze alternative	

<b>ID <u>CU 12</u></b> <b>Caso d'uso <u>InviaRichiestaAggiornamento</u></b>	
Descrizione	Un Globetrotter invia una richiesta di aggiornamento, affinché gli amministratori possano farlo diventare un Cicerone.
Attori primari	Globetrotter
Attori secondari	
Precondizioni	Il Globetrotter deve aver effettuato l'accesso.
Sequenza eventi	<p><u>1</u> Il Globetrotter chiede al sistema la creazione di una richiesta di aggiornamento.</p> <p><u>2</u> Il sistema permette la creazione di questo tipo di richiesta.</p> <p><u>3</u> Il Globetrotter inserisce i dati necessari.</p> <p><u>4</u> Il Globetrotter chiede al sistema l'invio della richiesta.</p> <p><u>5</u> Il sistema effettua l'invio della richiesta di aggiornamento.</p>
Postcondizioni	Viene creata una nuova richiesta di



	aggiornamento.
Sequenze alternative	

<b>ID <u>CU 13</u></b>	
<b>Caso d'uso <u>TrasformaInQuasiCicerone</u></b>	
Descrizione	Un amministratore provvede alla trasformazione di un Globetrotter in un QuasiCicerone, in base alla richiesta ricevuta.
Attori primari	Amministratore
Attori secondari	
Precondizioni	Dev'essere già stata inviata ALMENO una richiesta di aggiornamento.
Sequenza eventi	<p><u>1</u> L'amministratore chiede al sistema di visualizzare una particolare richiesta di aggiornamento.</p> <p><u>2</u> Il sistema mostra i dettagli associati alla richiesta di aggiornamento.</p> <p><u>3</u> Se l'amministratore accorda la richiesta di aggiornamento:</p> <p><u>3.1</u> Il sistema provvederà ad impostare come accordata tale richiesta.</p> <p><u>3.2</u> Il sistema cambierà il tipo d'utente del mittente della richiesta da Globetrotter a QuasiCicerone.</p> <p><u>3.3</u> Il sistema notificherà al mittente della richiesta ciò che è accaduto.</p>
Postcondizioni	L'amministratore potrà continuare a gestire le richieste d'amministrazione.
Sequenze alternative	

<b>ID <u>CU 14</u></b>	
<b>Caso d'uso <u>ConfermaTransizioneCicerone</u></b>	
Descrizione	Dopo aver ricevuto la conferma dell'avvenuto accordo della richiesta di

	aggiornamento, un QuasiCicerone conferma la transizione a Cicerone.
Attori primari	QuasiCicerone
Attori secondari	
Precondizioni	La richiesta di aggiornamento mandata quando l'utente era ancora Globetrotter deve essere stata accordata da un amministratore
Sequenza eventi	<u>1</u> Il QuasiCicerone chiede al sistema di confermare la transizione a Cicerone. <u>2</u> Il sistema provvede a confermare tale transizione.
Postcondizioni	Il QuasiCicerone è diventato Cicerone. Viene cambiato il tipo dell'utente da QuasiCicerone a Cicerone.
Sequenze alternative	

## 2.2.4 Corrispondenza tra Item funzionali e Casi d'uso

<i>Item funzionale</i>	<i>Caso d'uso</i>
<u>IF GN11</u>	<u>CU 1, ALT 1.1</u>
<u>IF GN12</u>	<u>CU 2, ALT 2.1</u>
<u>IF GN13</u>	<u>CU 3</u>
<u>IF GN14</u>	<u>CU 4</u>
<u>IF GN15</u>	<u>CU 5, ALT 5.1, ALT 5.2, ALT 5.3</u>
<u>IF GN21</u>	<u>CU 6</u>
<u>IF GN22</u>	<u>CU 7, ALT 7.1</u>
<u>IF GN23</u>	<u>CU 8</u>
<u>IF GN24</u>	<u>CU 9</u>
<u>IF GN25</u>	<u>CU 10</u>
<u>IF GN26</u>	<u>CU 11</u>
<u>IF GN31</u>	
<u>IF GN32</u>	
<u>IF GN33</u>	
<u>IF 1111</u>	
<u>IF 1112</u>	
<u>IF 1113</u>	
<u>IF 1121</u>	
<u>IF 1122</u>	
<u>IF 1123</u>	
<u>IF 1124</u>	
<u>IF 1131</u>	<u>CU 12</u>

<u>IF 1132</u>	<u>CU 13</u>
<u>IF 1133</u>	<u>CU 14</u>
<u>IF 2111</u>	
<u>IF 2112</u>	
<u>IF 2113</u>	
<u>IF 2211</u>	
<u>IF 2212</u>	
<u>IF 2221</u>	
<u>IF 2222</u>	
<u>IF 2223</u>	
<u>IF 2224</u>	

## 2.2.5 Interfacce

I seguenti sono i mockup di interfacce più significativi creati per questo sistema.

Visualizzazione profilo Globetrotter/QuasiCicerone (come proprietario dello stesso)

# Cicerone

NOTIFICAZIONI

MENU

Visualizzazione profilo

Globetrotter

100 x 100

nomeutente

pincopallino@stoccafisso.com

Descrizione

DOMO ARIGATO MR ROBOTO  
DOMO ARIGATO MR ROBOTO  
DOMO ARIGATO MR ROBOTO  
DOMO ARIGATO MR ROBOTO  
DOMO ARIGATO MR ROBOTO

Vedi feedback

Vedi Itinerari a cui ho partecipato

Invia richiesta d'aggiornamento

Invia richiesta di disiscrizione

Conferma transizione a Cicerone

modifica

Annulla

Visualizzazione profilo Cicerone (come proprietario dello stesso)

# Cicerone

NOTIFICAZIONI

MENU

**Visualizzazione profilo** Cicerone



nomeutente  
pincopallino@stoccafisso.com

Nome:	stoccafisso	Descrizione
Cognome:	stoccafisso	DOMO ARIGATO MR ROBOTO
Data di nascita:	08 Marzo 2016	DOMO ARIGATO MR ROBOTO
Luogo di nascita:	Londra	DOMO ARIGATO MR ROBOTO
Residenza	Londra	DOMO ARIGATO MR ROBOTO
Codice fiscale	AFSD1234	DOMO ARIGATO MR ROBOTO
Telefono cellulare	0804444444	

Vedi feedback

Vedi Itinerari

modifica informazioni anagrafiche

Invia richiesta di disiscrizione

modifica profilo

Annulla

Modifica del profilo

# Cicerone

NOTIFICAZIONI

MENU

**Modifica profilo di** nomeutente

vecchia pss

nuova pss

conf. nuova pss

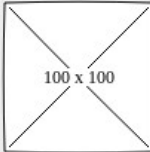
e-mail

text

text

text

text



Descrizione

Secrevit fontes liquidum locoque pronaque?  
Illas semine campoque declivia oppida

Applica modifiche

Annulla

## Modifica dell'anagrafica di un Cicerone

**Cicerone**

NOTIFICAZIONI

MENU

**Modifica informazioni anagrafiche**

Luogo di residenza

text

Telefono

text

Applica modifiche

Annulla

## Creazione richiesta disiscrizione

**Cicerone**

NOTIFICAZIONI

MENU

**Creazione richiesta di disiscrizione**

Motivi:

Motivazioni

Annulla

Invia

## Creazione richiesta aggiornamento

# Cicerone

NOTIFICA  
TIONS

MENU

### Creazione richiesta di divenire Cicerone

Nome:

Cognome:

Data di nascita

Luogo di nascita

Luogo di residenza

Telefono

Codice fiscale

Invia

Annulla



## 2.2.6 Qualità

Perspective: Overall Status ▾

Sort by: Name ▾

🔍 Search by project name or key

1 projects 🏠

★ Cicerone

Passed

0 A 🐛 Bugs

0 A 🛡 Vulnerabilities

25 A 🧑‍💻 Code Smells

0.0% 📉 Coverage

1.0% 🔄 Duplications

Last analysis: December 30, 2019, 2:29 AM

5.6k S 📄 PHP, CSS, ...

1 of 1 shown

## About This Project

New code: since previous version started 2 months ago

0 A

25

 Code Smells

0.0%

Coverage on  
1.8k New Lines to Cover

1.2%  
Duplications on  
9.7k New Lines

» No tags ▾

PHP 4.3k

HTML 423

JavaScript | 212

### Project Activity



December 30, 2019

1.0

December 4, 2019

Project Analyzed

December 3, 2019

Quality Gate: Green (was Red)

[Show More](#)

## Quality Gate

Sonar way (no code coverage)

## Quality Profiles

(CSS) Sonar way

(JavaScript) Sonar way

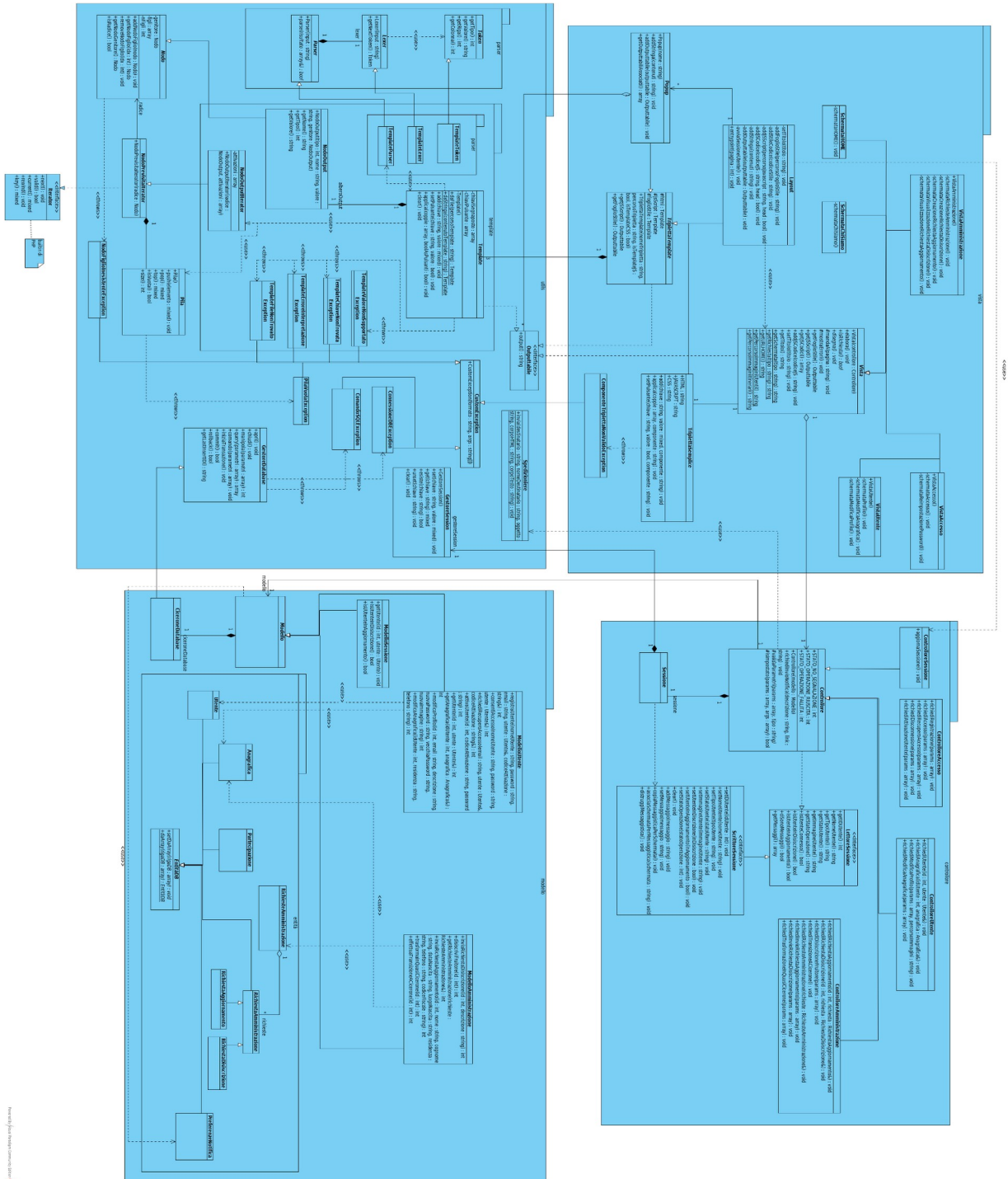
(PHP) Sonar way

(HTML) Sonar way

### 2.2.7 Altro

# 3. Product Design

## 3.1 Diagramma delle Classi



## 3.2 Specifiche delle Classi

### 3.2.1 I package

- vista:  
Contiene tutte le classi/interfacce che consentono al sistema Cicerone di mostrare le schermate per gli utenti. Si occupano anche di fare da “mediatori” tra utente e sistema, consentendo all’utente di richiedere l’esecuzione di servizi.
- controllore:  
Contiene tutte le classi/interfacce che consentono al sistema Cicerone di raccogliere le richieste avanzate dagli utenti e trasformarle in operazioni da eseguire. Si occupano di fare da “mediatori” tra viste e modelli.
- modello:  
Contiene tutte le classi/interfacce che realizzano (se si tratta di classi) le funzionalità che il sistema Cicerone è in grado di compiere. Tra le funzionalità abbiamo anche quelle di lettura e scrittura dei dati persistenti.
- modello/entità:  
Contiene tutte le classi che rappresentano i dati persistenti del sistema Cicerone.
- utils:  
Contiene tutte le classi e i package d’ausilio sfruttati dalle classi contenute negli altri tre package.
- utils/parser:  
Contiene tutte le classi (astratte) necessarie alla costruzione di nuovi linguaggi (molto semplici).
- utils/template:  
Contiene tutte le classi necessarie per realizzare appieno le funzionalità del motore utilizzato dal sistema Cicerone per la generazione delle schermate associate alle viste.
- utils/template/parser:  
Contiene tutte le classi necessarie per l’interpretazione del linguaggio Template utilizzato per la creazione delle schermate.
- builtin di PHP:  
Sebbene non si tratti di un vero e proprio package, si fa qui riferimento ad alcune classi/interfacce che fanno parte del linguaggio PHP.

Il “package” è considerato l'esterno, ovvero, ciò che non è inserito all'interno dei package del sistema Cicerone, sarà considerato *builtin di PHP*.

### 3.2.2 Le classi/interfacce

- L'interfaccia *Outputtabile* (package *utils*) definisce quelle che sono le operazioni disponibili ad eventuali classi realizzatrici. L'unica operazione che definisce, consente di stabilire il prodotto output testuale, di una classe realizzatrice. Tale risultato testuale, può esser poi sfruttato in vari modi.
- La classe *Spedizioniere* (package *utils*) consente di spedire email.
- La classe *GestoreSession* (package *utils*) consente di gestire a basso livello, in lettura e scrittura, i dati di sessione.
- La classe *CustomException* (package *utils*) rappresenta un'eccezione personalizzata.  
Tutte le eccezioni del sistema Cicerone ereditano da questa classe.
- La classe *GestoreDatabase* (package *utils*) è una classe di tipo *connection*, che consente di gestire le connessioni col DB, a basso livello.
- Le classi *ComandoSQLException* e *ConnessioneDBException* (package *utils*) sono eccezioni sollevate dalla classe *GestoreDatabase*, rispettivamente, quando c'è un problema con i comandi e quando c'è un errore di connessione col DB.
- La classe *Template* (package *utils/template*) è una realizzazione di *Outputtabile*, di conseguenza anch'essa produce un output testuale. Consente il caricamento di file o stringhe Template. Effettua dunque l'interpretazione, ed infine, crea un albero di output, ovvero, un albero che rappresenta la struttura interna del file/stringa Template, contenente testo semplice e chiavi. Il linguaggio Template supporta la definizione di due tipi di chiave:
  - Chiave segnaposto:  
Sono chiavi che, possono essere sostituite con del contenuto arbitrario.  
Fungono appunto da segnaposto.
  - Chiave pulsante:  
Sono chiavi che, contrariamente alle prime, raggruppano contenuti che possono essere testo semplice, ulteriori chiavi segnaposto e chiavi pulsante.  
Queste chiavi possono essere “attivate” o “disattivate” ed in base a

ciò, verranno mostrati o NON verranno mostrati i loro contenuti nell'output finale.

In fase di elaborazione dell'output, questo viene costruito effettuando una previsita dell'albero di output.

Se durante la visita:

- si incontra del testo semplice: viene banalmente inserito nell'output
  - si incontra una chiave segnaposto: si inserisce nell'output il contenuto assegnatole
  - si incontra una chiave pulsante: si inserisce nell'output il contenuto associato solo se la chiave risulta attivata.
- Le classi *TemplateNonTrovatoException*, *TemplateChiaveNonTrovataException*, *TemplateErroreInterpretazioneException* e *TemplateValoreNonSupportatoException* (package *utils/template*) sono eccezioni sollevate dalla classe *Template*, rispettivamente, quando il file Template da caricare non è stato trovato, quando si fa riferimento ad una chiave inesistente nel file/stringa Template, quando il file/stringa Template contiene errori, ad esempio di sintassi e quando si utilizzano come valori delle chiavi, tipi che non sono istanze di Outputtabile, stringhe e booleani.
  - La classe *NodoOutput* (package *utils/template*) viene utilizzata dalla classe *Template* per immagazzinare l'albero di output. Estende la classe astratta *Nodo* (package *utils*).
  - La classe *NodoOutputIterator* (package *utils/template*) viene utilizzata dalla classe *Template* durante l'elaborazione dell'output per effettuare la revisita dell'albero di output. Estende la classe *NodoPrevisitaIterator* (package *utils*).
  - Le classi *TemplateToken*, *TemplateLexer* e *TemplateParser* (package *utils/template/parser*) servono all'interpretazione del linguaggio Template effettuata dalla classe *Template*. Estendono rispettivamente le classi *Token*, *Lexer* e *Parser* (package *utils/parser*).
  - La classe astratta *Nodo* (package *utils*) a dispetto del nome, può rappresentare sia un nodo sia la struttura dati Albero n-ario.
  - La classe *NodoFiglioInesistenteException* (package *utils*) è un'eccezione sollevata dalla classe astratta *Nodo*, quando appunto, viene richiesto un nodo figlio inesistente.
  - La classe astratta *NodoPrevisitaIterator* (package *utils*) realizza l'interfaccia *Iterator* (*builtin di PHP*).

Consente di effettuare una pre visita di un albero n-ario realizzato mediante istanze di Nodo.

- La classe Pila (package utils) rappresenta la struttura dati Pila.
- Le classi astratte Token, Lexer e Parser (package utils/parser) sono tutte classi d'ausilio per la creazione di semplicissimi linguaggi. Per semplici non intendo dire che è semplice crearli, ma che sono semplici, cioè "non complessi".
- La classe CiceroneDatabase (package modello), eredita da GestoreDatabase (package utils). Consente di gestire le connessioni con lo specifico DB associato al sistema Cicerone.
- La classe astratta Modello (package modello), funge da base per tutte le classi che appartengono a questa categoria (come ModelloUtente). Possiede un'istanza di CiceroneDatabase, che garantisce dunque che ogni classe figlia potrà aprire connessioni col DB.
- La classe ModelloSessione (package modello) realizza tutte le funzionalità necessarie per il reperimento delle informazioni utili ad aggiornare i valori della sessione corrente.
- La classe ModelloUtente (package modello) realizza tutte le funzionalità necessarie per l'accesso degli utenti e per la gestione dei profili.
- La classe ModelloAmministrazione (package modello) realizza tutte le funzionalità necessarie per la gestione delle richieste d'amministrazione.
- La classe astratta EntitàDB (package modello/entità) rappresenta una qualunque entità presente nel DB. Ogni entità che rappresenta dei dati persistenti del sistema, deve estendere questa classe.
- La classe Utente (package modello/entità) rappresenta l'insieme dei dati persistenti inerenti gli utenti, che vengono memorizzati nel DB. Eredita da EntitàDB.
- La classe Anagrafica (package modello/entità) rappresenta l'insieme dei dati persistenti inerenti le anagrafiche associate ai Ciceroni. Eredita da EntitàDB.
- La classe RichiesteAmministrazione (package modello/entità) rappresenta una collezione di istanze di RichiestaAmministrazione.
- La classe astratta RichiestaAmministrazione (package modello/entità) rappresenta una richiesta d'amministrazione generica. Eredita da EntitàDB.



- La classe RichiestaDisiscrizione (package modello/entità) rappresenta l'insieme dei dati persistenti inerenti le richieste di disiscrizione. Eredita da RichiestaAmministrazione.
- La classe RichiestaAggiornamento (package modello/entità) rappresenta l'insieme dei dati persistenti inerenti le richieste di aggiornamento. Eredita da RichiestaAmministrazione.
- L'interfaccia LetturaSessione (package controllore) definisce quelle che sono le operazioni disponibili ad eventuali classi realizzatrici. Tali operazioni consentono la sola lettura dei dati di sessione (per impedirne eventuali modifiche). Definisce anche quali sono i dati memorizzati in una sessione.
- L'interfaccia ScritturaSessione (package controllore) definisce quelle che sono le operazioni disponibili ad eventuali classi realizzatrici. Tali operazioni consentono ANCHE la scrittura dei dati di sessione. Dico ANCHE e non SOLO perché ScritturaSessione è un'interfaccia figlia di LetturaSessione.
- La classe Sessione (package controllore) consente di gestire ad alto livello i dati di sessione. In particolare, consente la scrittura e lettura dei soli dati previsti in LetturaSessione. Realizza l'interfaccia ScritturaSessione e possiede un'istanza di GestoreSessione (package utils).
- La classe astratta Controllore (package controllore), funge da base per tutte le classi che appartengono a questa categoria (come ControlloreAccesso). Possiede un'istanza di Sessione, che garantisce dunque che ogni classe figlia potrà gestire i dati di sessione. Definisce anche alcuni codici d'errore generici che possono essere utilizzati per la creazione di messaggi d'errore/informativi che vengono inviati alle classi di tipo "vista".
- La classe ControlloreSessione (package controllore) è un controllore che possiede un'istanza di ModelloSessione (package modello). Consente solo la gestione dei dati di sessione e l'aggiornamento degli stessi in caso di cambiamenti particolari come la transizione a Cicerone o la trasformazione a QuasiCicerone. È utilizzato perlopiù dalla classe SchermataChisiamo e Layout (package vista).
- La classe ControlloreAccesso (package controllore) è un controllore che possiede un'istanza di ModelloUtente (package modello). Viene utilizzato dalla classe VistaAccesso (package vista), per eseguire

richieste, che poi vengono tramutate in operazioni della classe ModelloUtente.

- La classe ControlloreUtente (package controllore) è un controllore che possiede un'istanza di ModelloUtente (package modello). Viene utilizzato dalla classe VistaUtente (package vista), per eseguire richieste, che poi vengono tramutate in operazioni della classe ModelloUtente.
- La classe ControlloreAmministrazione (package controllore) è un controllore che possiede un'istanza di ModelloAmministrazione (package modello). Viene utilizzato dalla classe VistaAmministrazione (package vista), per eseguire richieste, che poi vengono tramutate in operazioni della classe ModelloAmministrazione.
- La classe astratta TriplettaTemplate (package vista) rappresenta una generica tripla di file, che viene utilizzata per definire un generico elemento appartenente alle schermate del sistema Cicerone. Ogni tripla è composta da tre file con lo stesso nome, ma con diverse estensioni, ovvero, html (file Template HTML), js (file Template Javascript) e css (file Template foglio di stile). Ogni classe figlia rappresenta dunque una particolare tripla che definisce un elemento di una o più schermate, che assolve determinati compiti.  
Lo scopo di questa classe astratta è quello di semplificare il caricamento dei tre diversi file, fornendo il percorso e il nome della tripletta. Essa possiede ben tre istanze di Template (package utils/template) e realizza Outputtabile (package utils).
- La classe astratta Vista (package vista) funge da base per tutte le classi che appartengono a questa categoria (come VistaAccesso). Possiede un'istanza di Controllore (package controllore). Le classi figlie possono fare riferimento a controllori più specifici per consentire la richiesta di particolari servizi. Il metodo astratto *disegna*, consente di definire quel che deve essere fatto per creare la schermata (o le schermate) prima della messa in output.  
Realizza Outputtabile (package utils).
- La classe Layout (package vista) rappresenta la disposizione generale delle pagine del sistema Cicerone. È la classe che dà ordine al sistema di generare le pagine che verranno servite agli utenti e di elaborare le eventuali richieste. Possiede il metodo statico *entrypoint*, che istanzia se stessa e una particolare vista in base al parametro fornitogli, in maniera tale da automatizzare la creazione delle singole pagine.

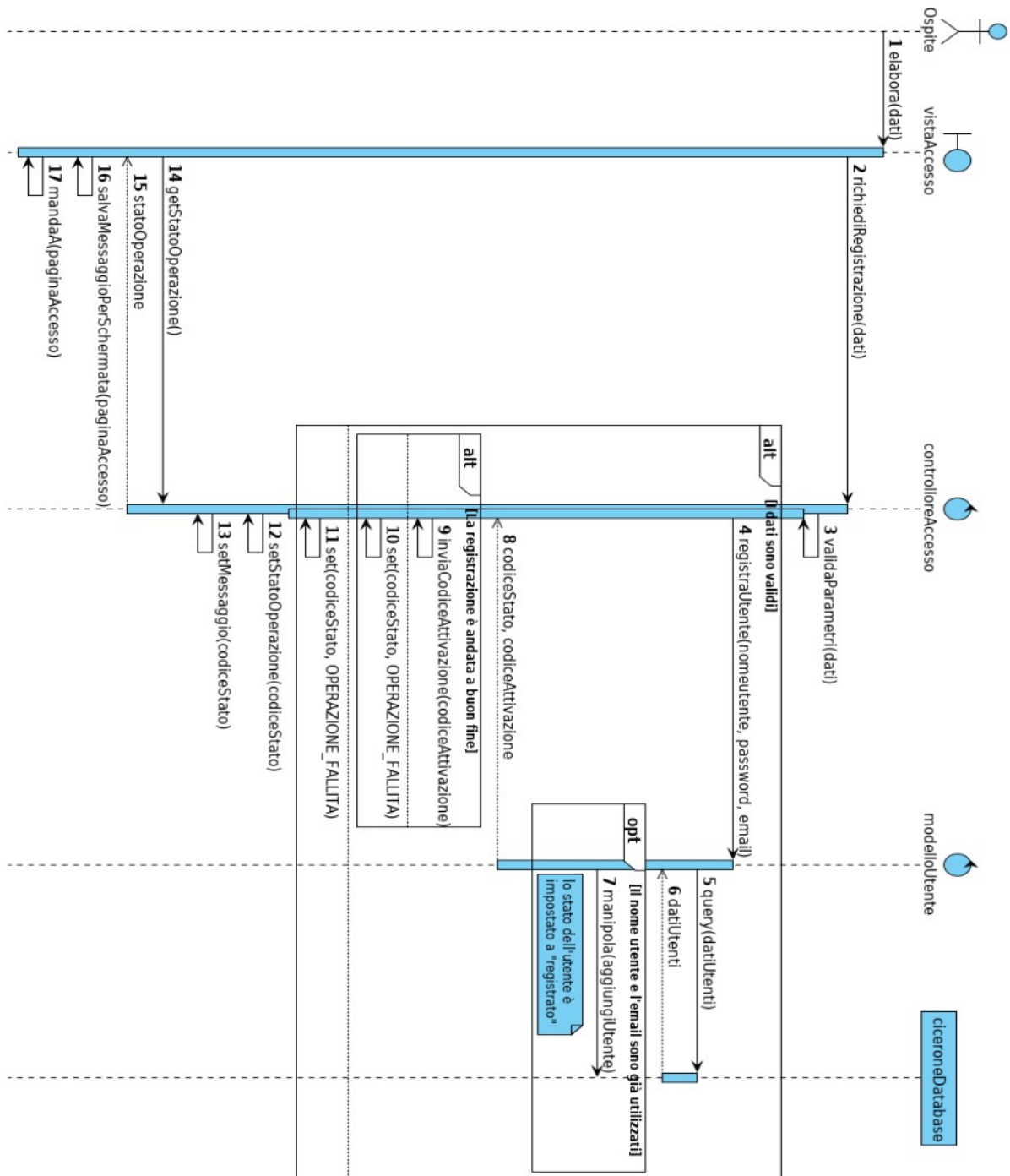
Permette l'aggiunta di istanze di Outputtabile (package utils), di impostare la barra laterale (o sidebar), aggiungere codici Javascript, CSS ed altri contenuti arbitrari. Poichè questa classe è figlia di TriplettaTemplate, che a sua volta realizza Outputtabile, alla chiamata del metodo output, avremo una stringa che rappresenta il contenuto della pagina attualmente richiesta dall'utente. La suddetta classe ha riferimenti, possiede ed utilizza tutte le classi figlie di Vista.

- La classe Popup (package vista) rappresenta una finestrella "popup" che visualizza messaggi o consente ulteriori interazioni. Questa classe è figlia di TriplettaTemplate.
- La classe TriplettaSemplice (package vista) rappresenta una tripletta Template "semplice", in contrapposizione alle altre classi figlie di TriplettaTemplate che esibiscono un comportamento più articolato (vedasi Popup e Layout). Viene utilizzata per gestire triplette Template che non richiedono comportamenti azioni particolari, oltre alla sostituzione/attivazione di chiavi segnaposto/pulsante.
- La classe ComponenteTriplettaNonValidoException (package vista) è un'eccezione sollevata dalla classe TriplettaSemplice quando si cerca di impostare una chiave su un componente inesistente (notare l'uso del maschile).
- La classe SchermataChiSiamo (package vista) rappresenta la schermata utilizzata dal sistema Cicerone per mostrare le informazioni su chi ha prodotto/gestisce il sistema stesso. È classe figlia di Vista.
- La classe VistaAccesso (package vista) rappresenta le schermate utilizzate dal sistema per consentire agli utenti di effettuare l'accesso, registrarsi, recuperare l'accesso e così via. È classe figlia di Vista.
- La classe VistaUtente (package vista) rappresenta le schermate utilizzate dal sistema per consentire agli utenti di vedere profili, gestire il proprio e modificare le informazioni anagrafiche (se trattasi di Ciceroni). È classe figlia di Vista.
- La classe VistaAmministrazione (package vista) rappresenta le schermate utilizzate dal sistema per consentire la visualizzazione e gestione delle richieste d'amministrazione. È classe figlia di Vista.

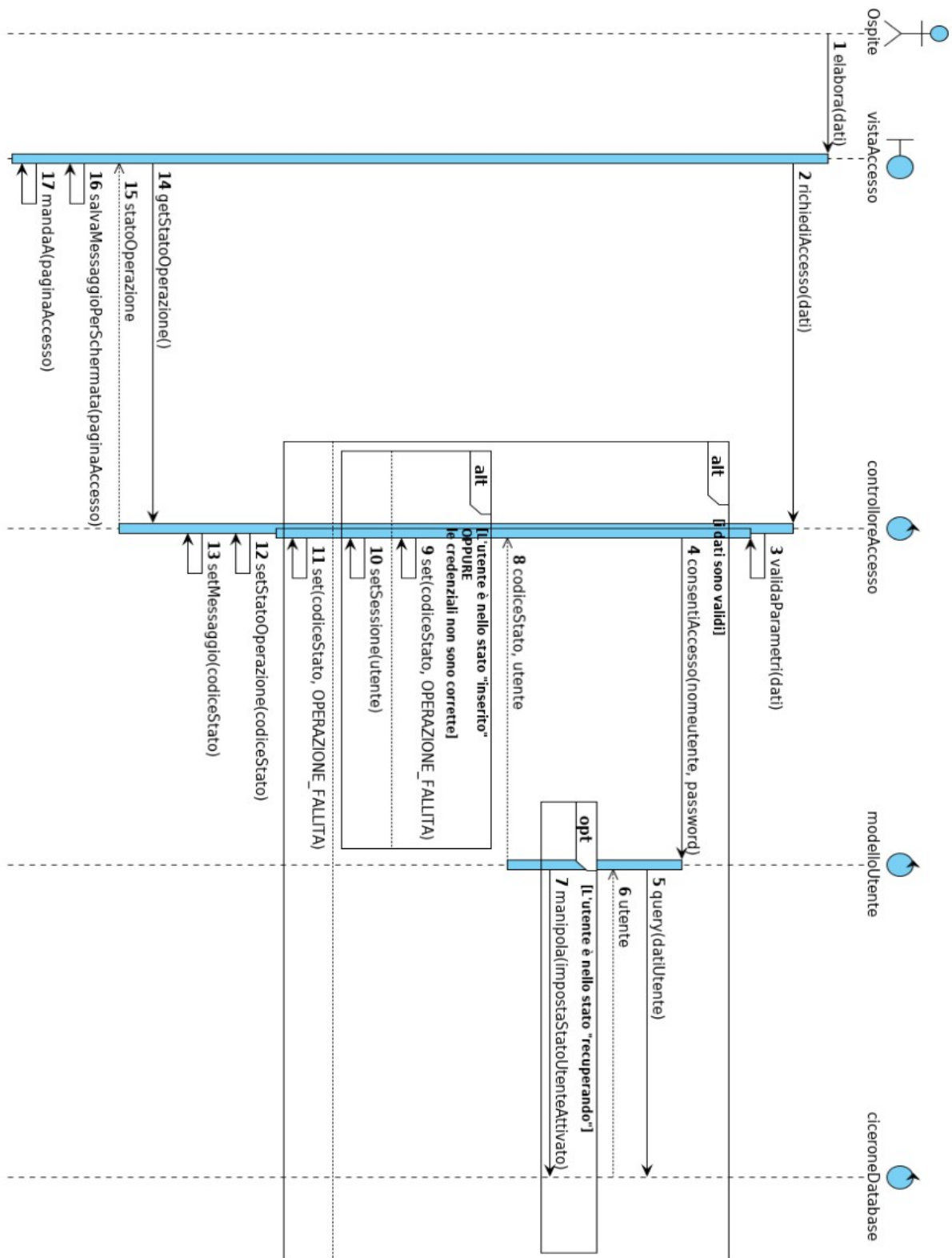
### 3.3 Diagrammi di sequenza

#### 3.3.1 Diagrammi

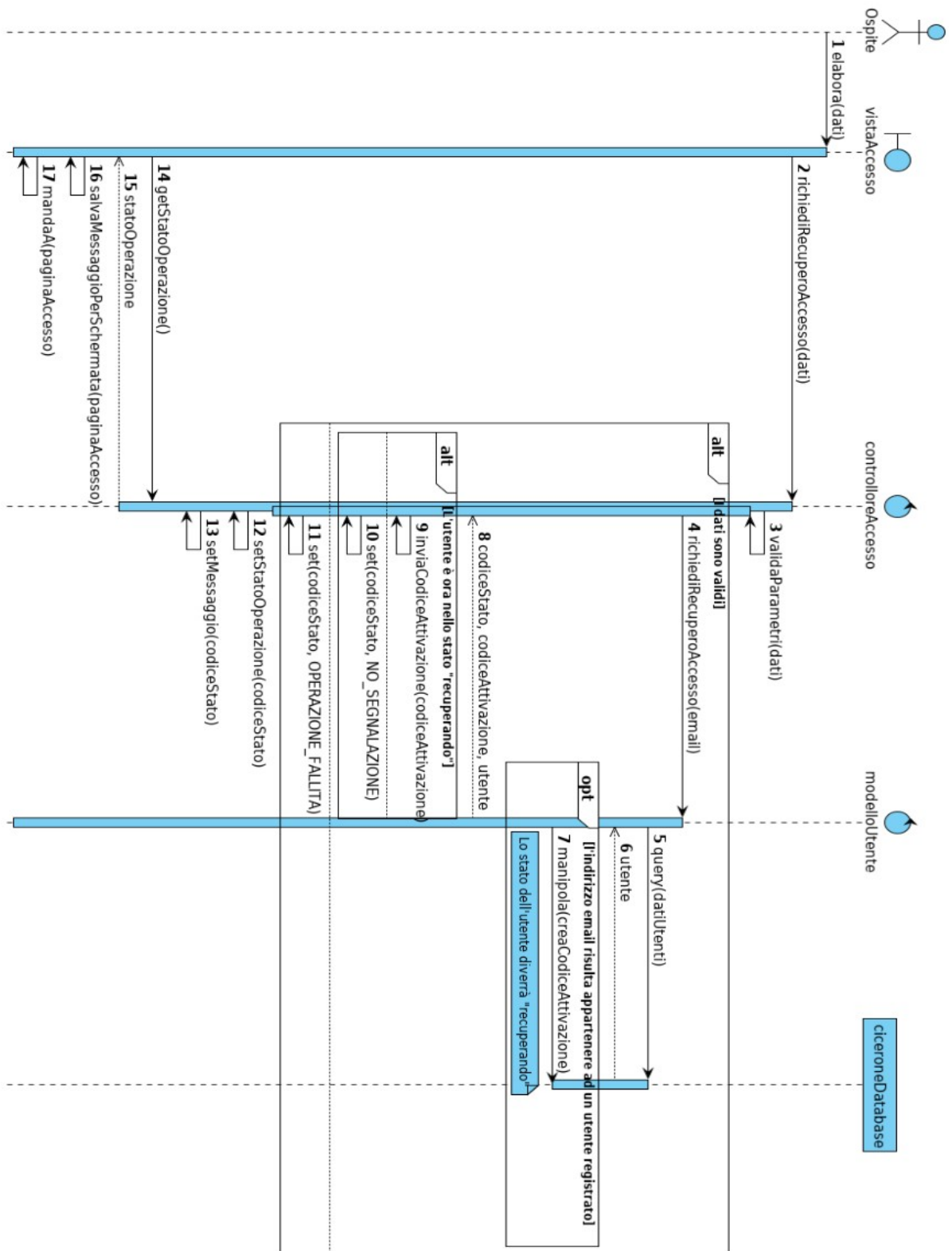
#### SD\_1 – RegistrazioneUtente



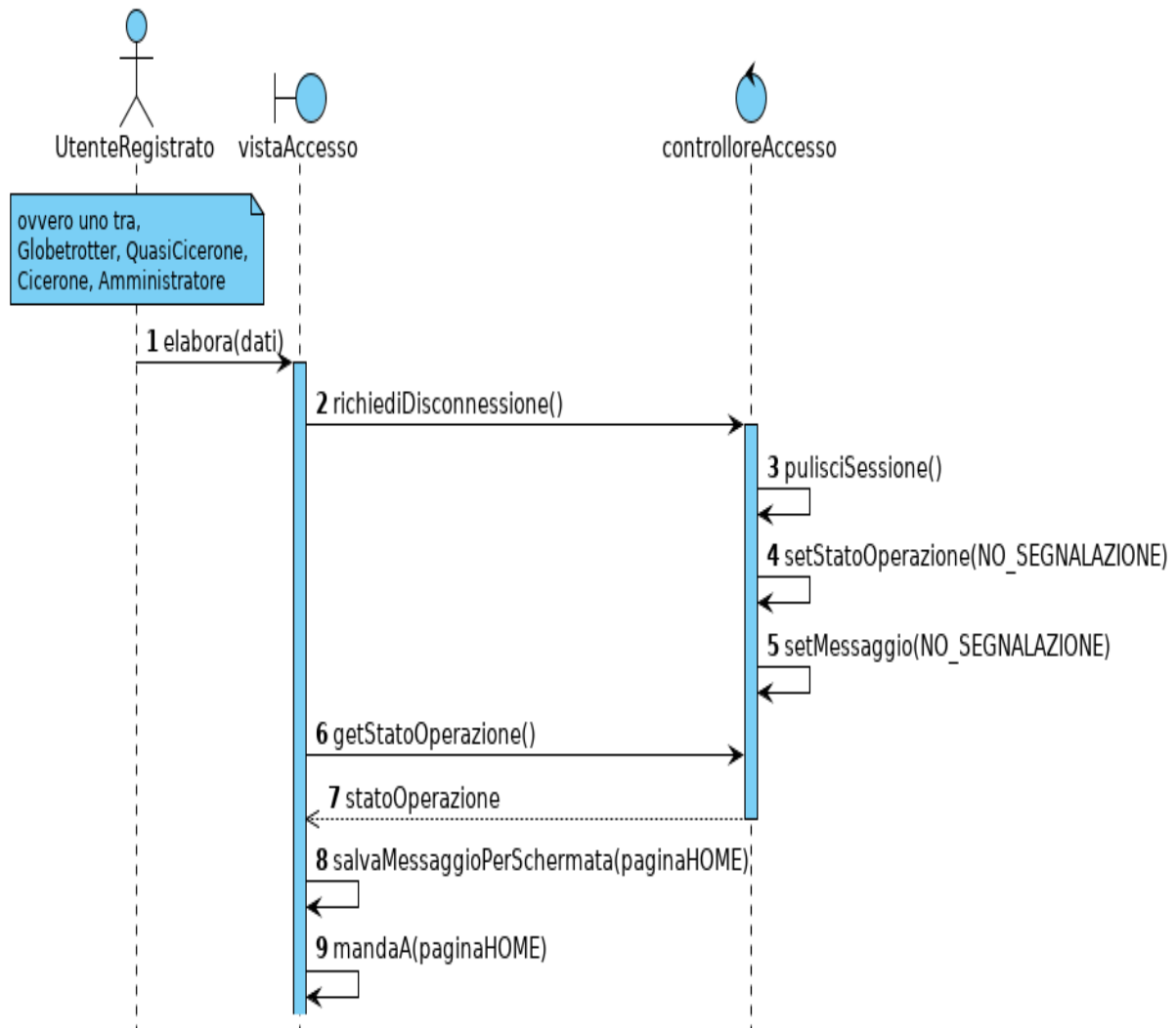
## SD\_2 - EffettuaAccesso



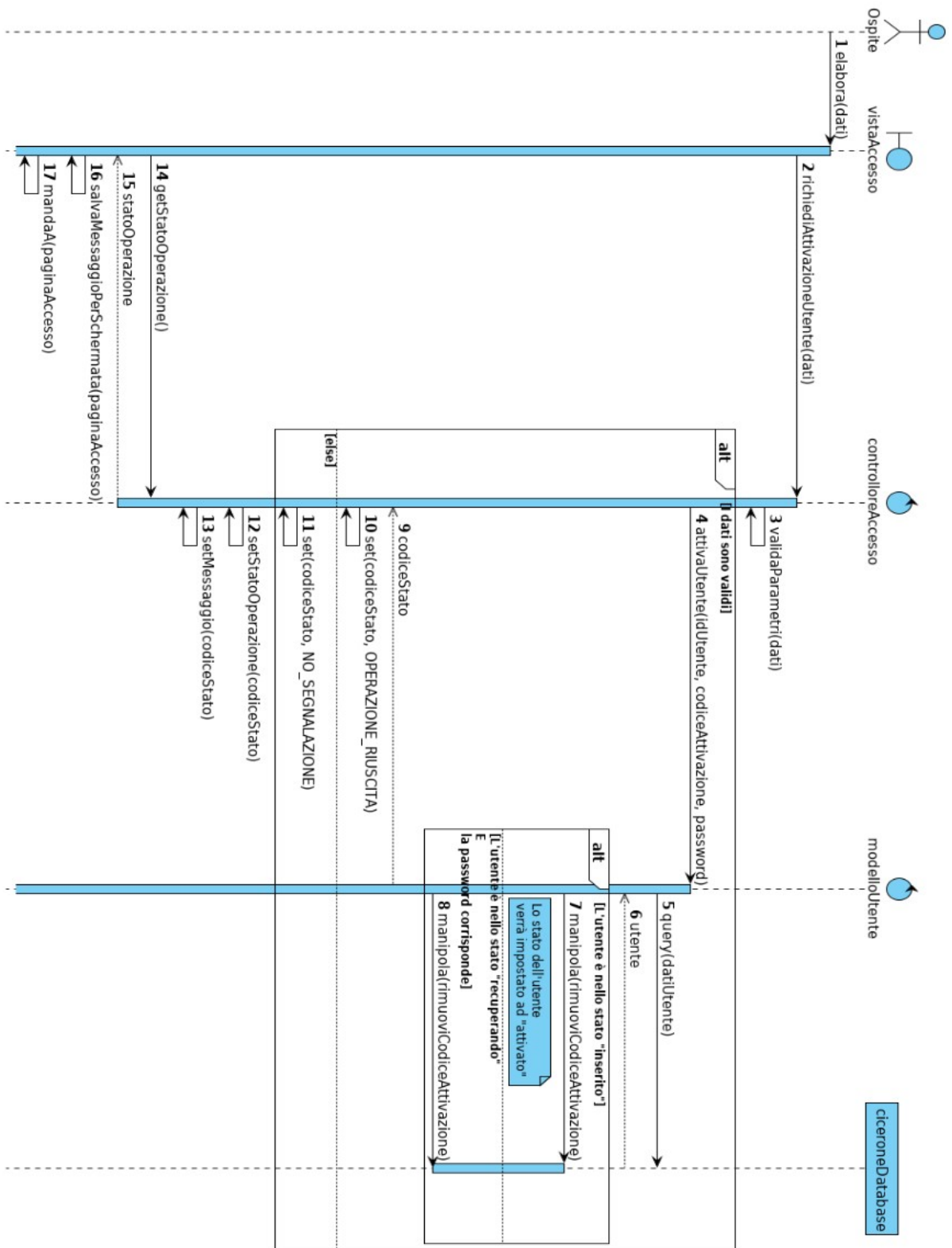
## SD\_3 - RecuperaAccesso



## SD\_4 – DisconnettiUtente

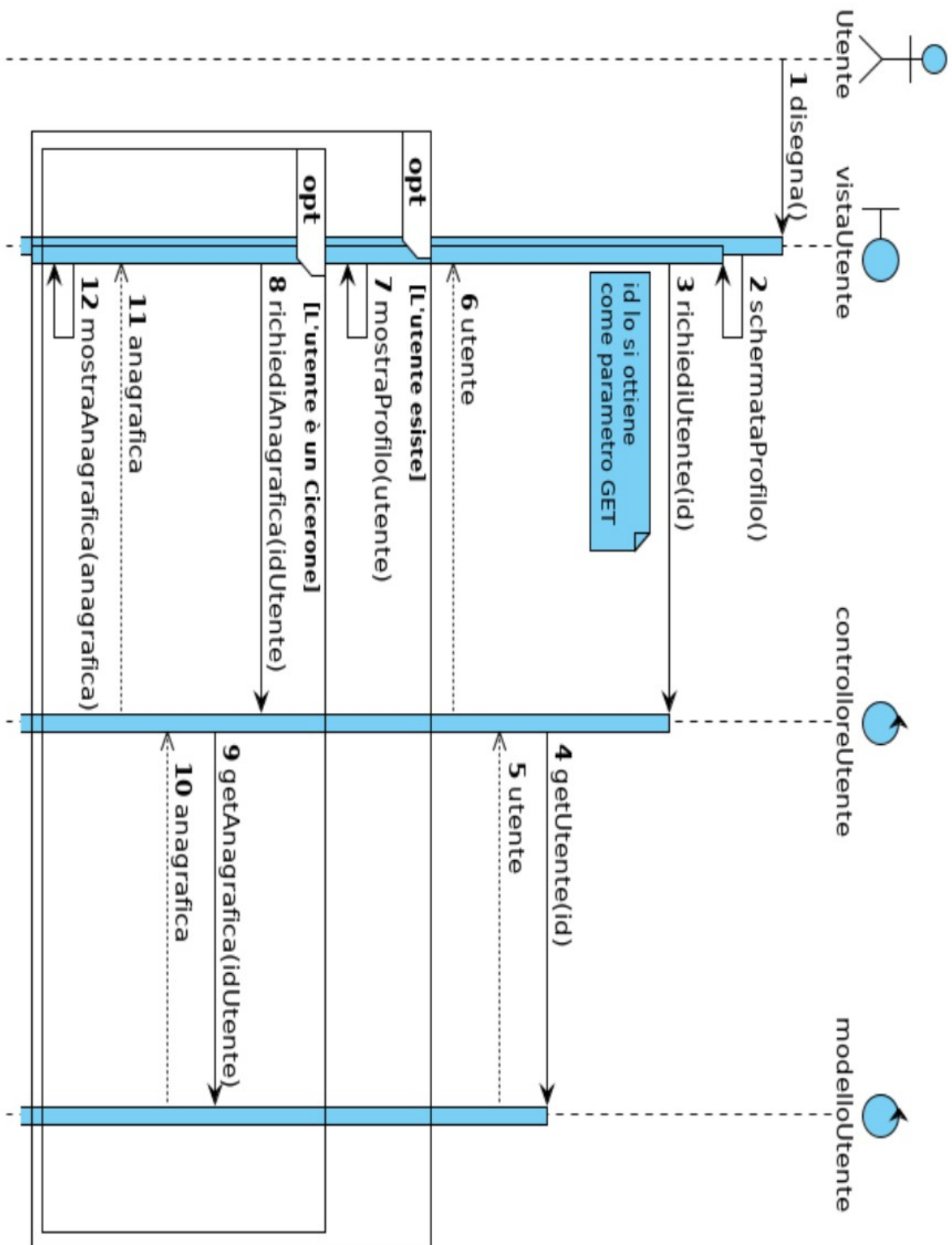


## SD\_5 - AttivaUtente

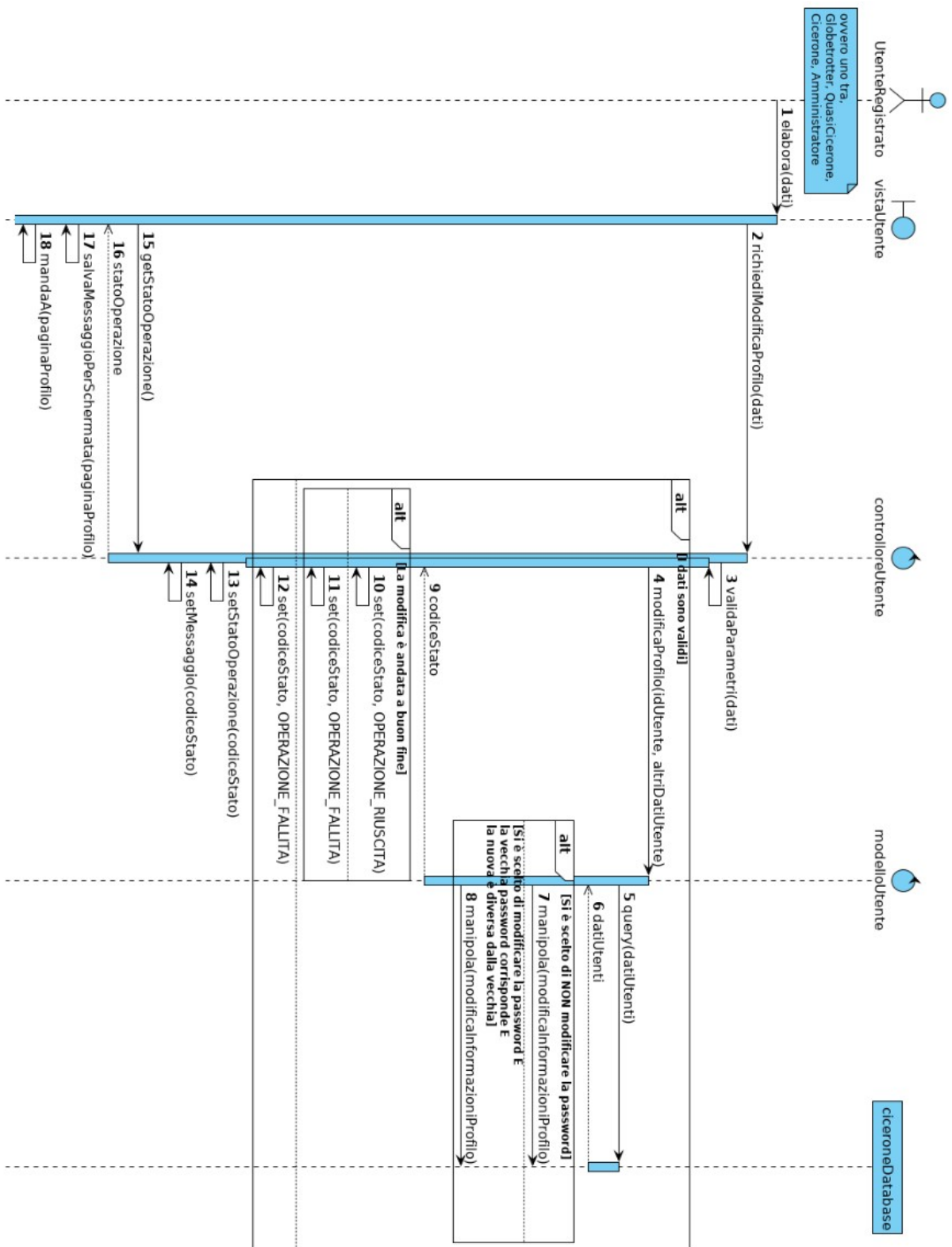




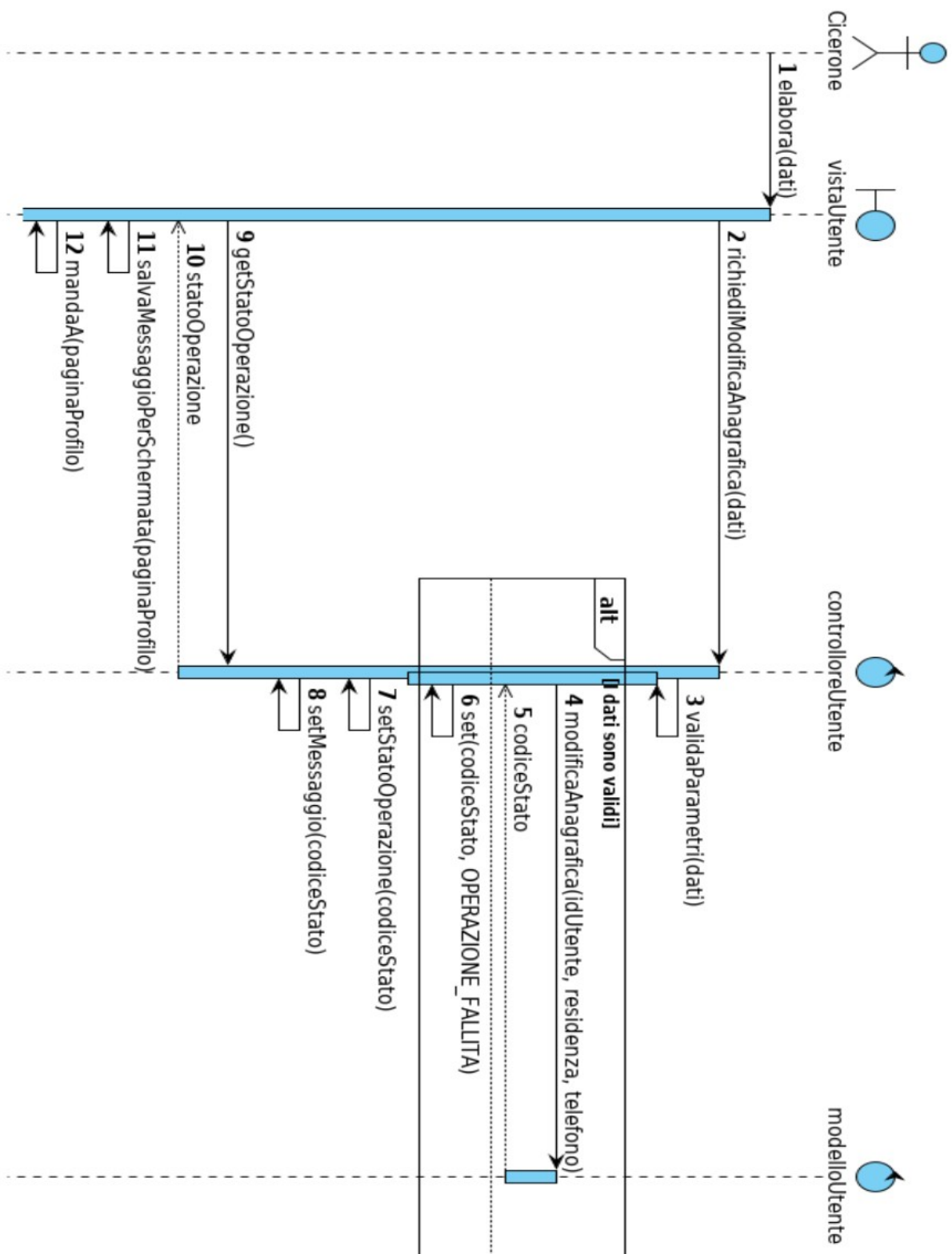
## SD\_6 - VisualizzaProfilo



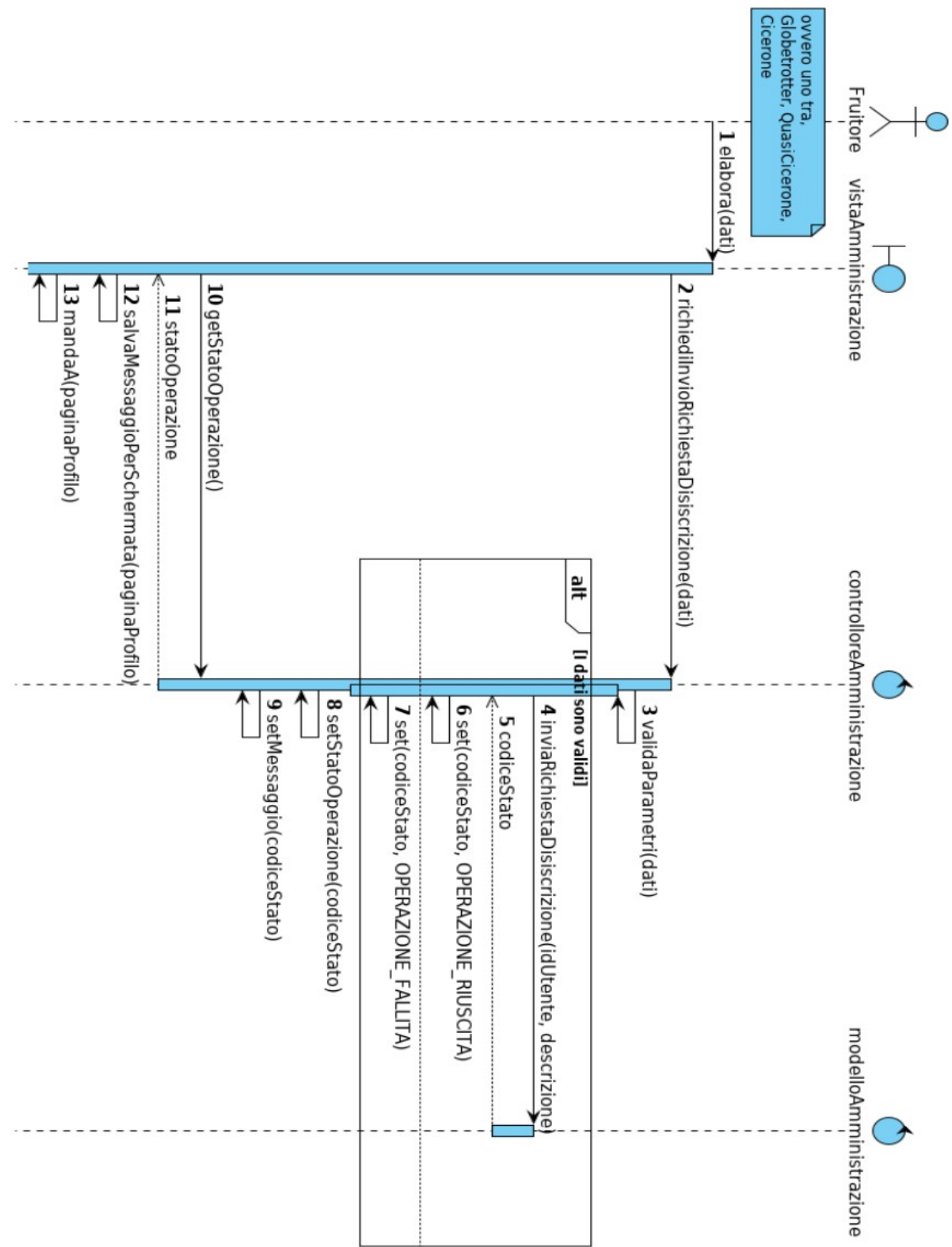
## SD\_7 - ModificaProfilo



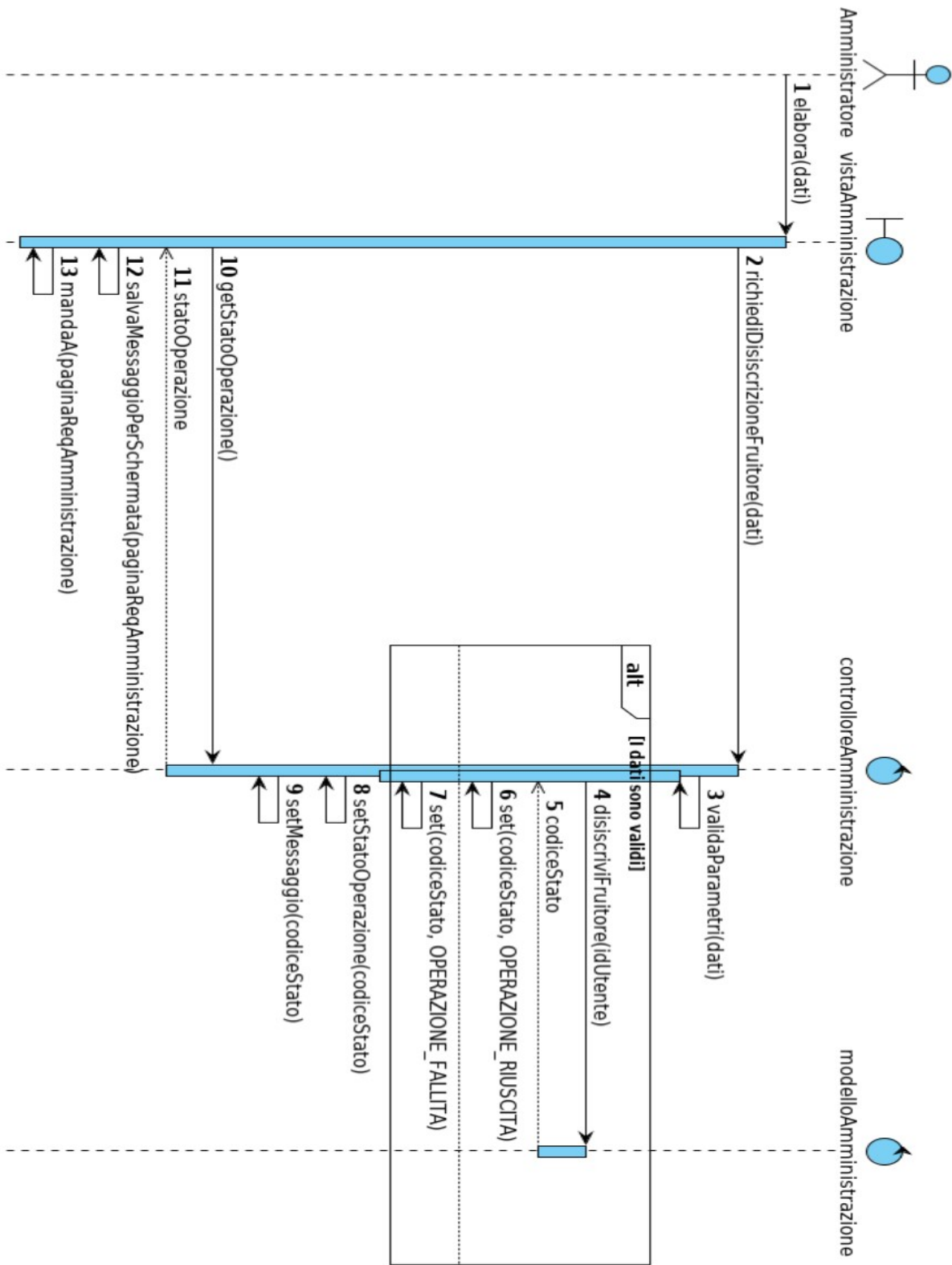
## SD\_8 - ModificaAnagrafica



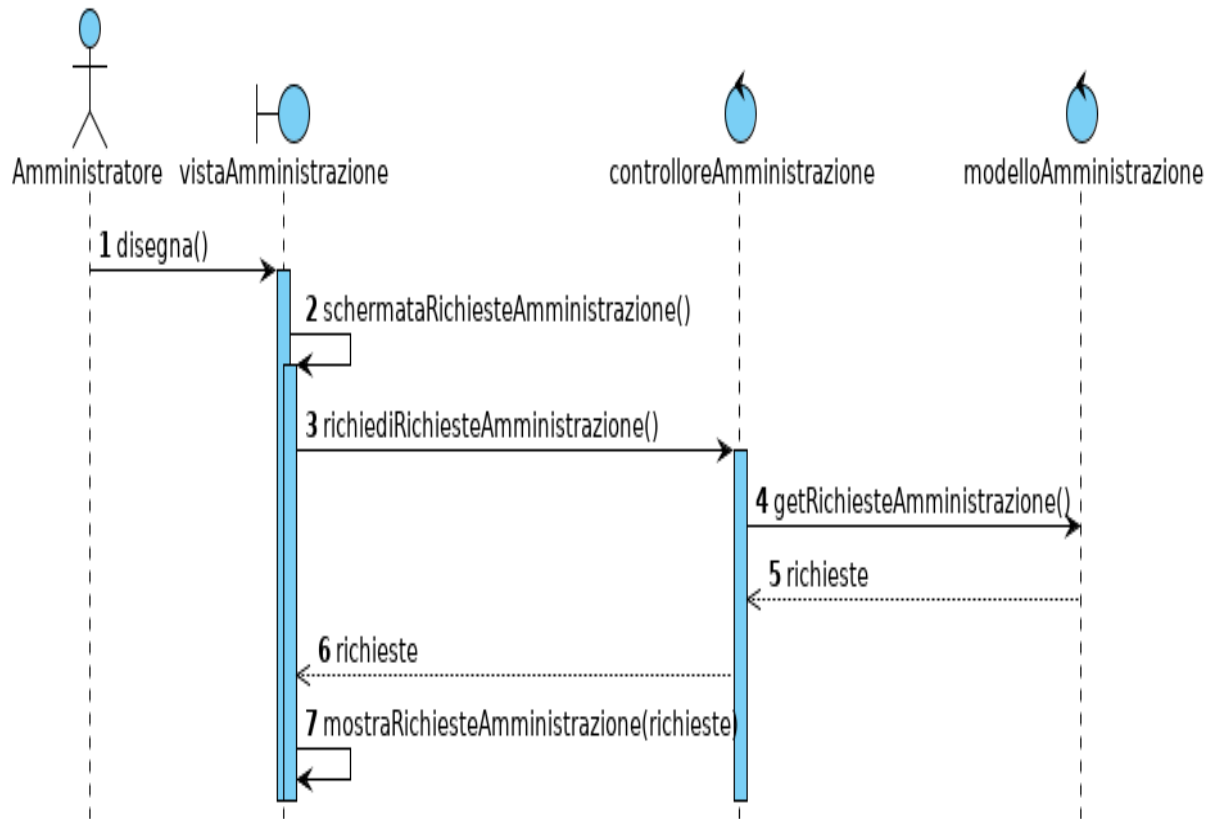
SD\_9 - InviaRichiestaDisiscrizione



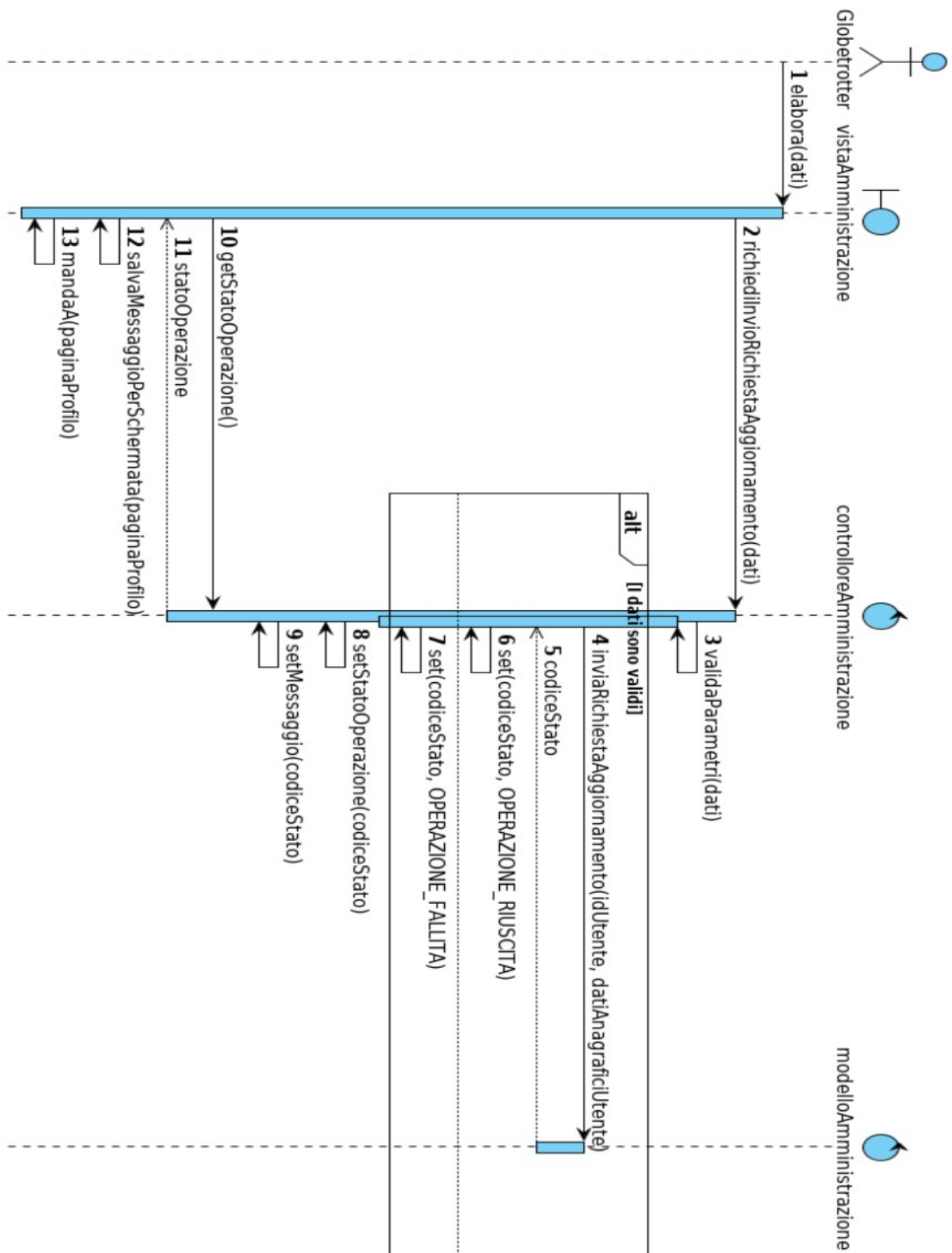
SD\_10 - DisiscriviFruitore



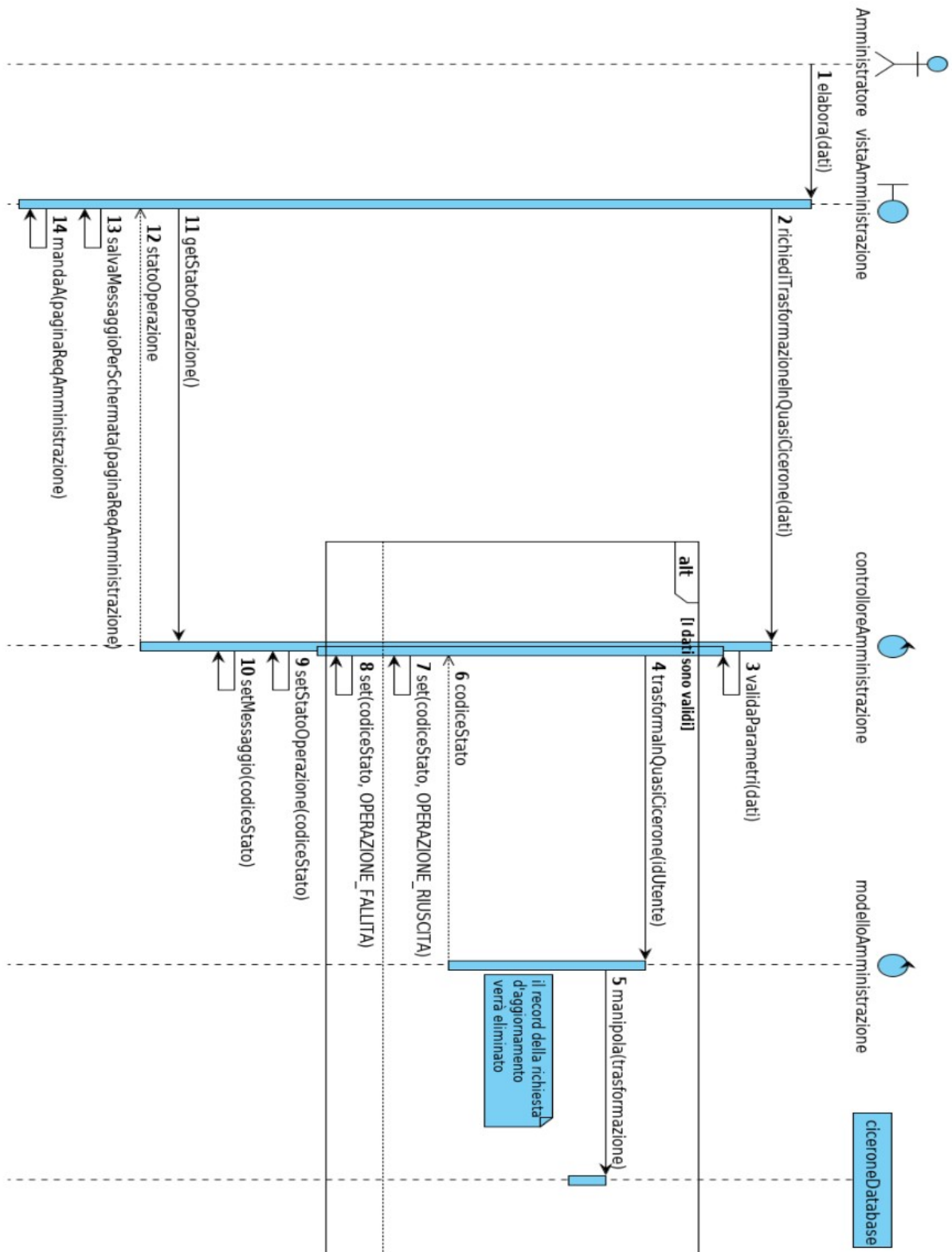
## SD\_11 - VisualizzaRichiesteAmministrazione



## SD\_12 - InviaRichiestaAggiornamento

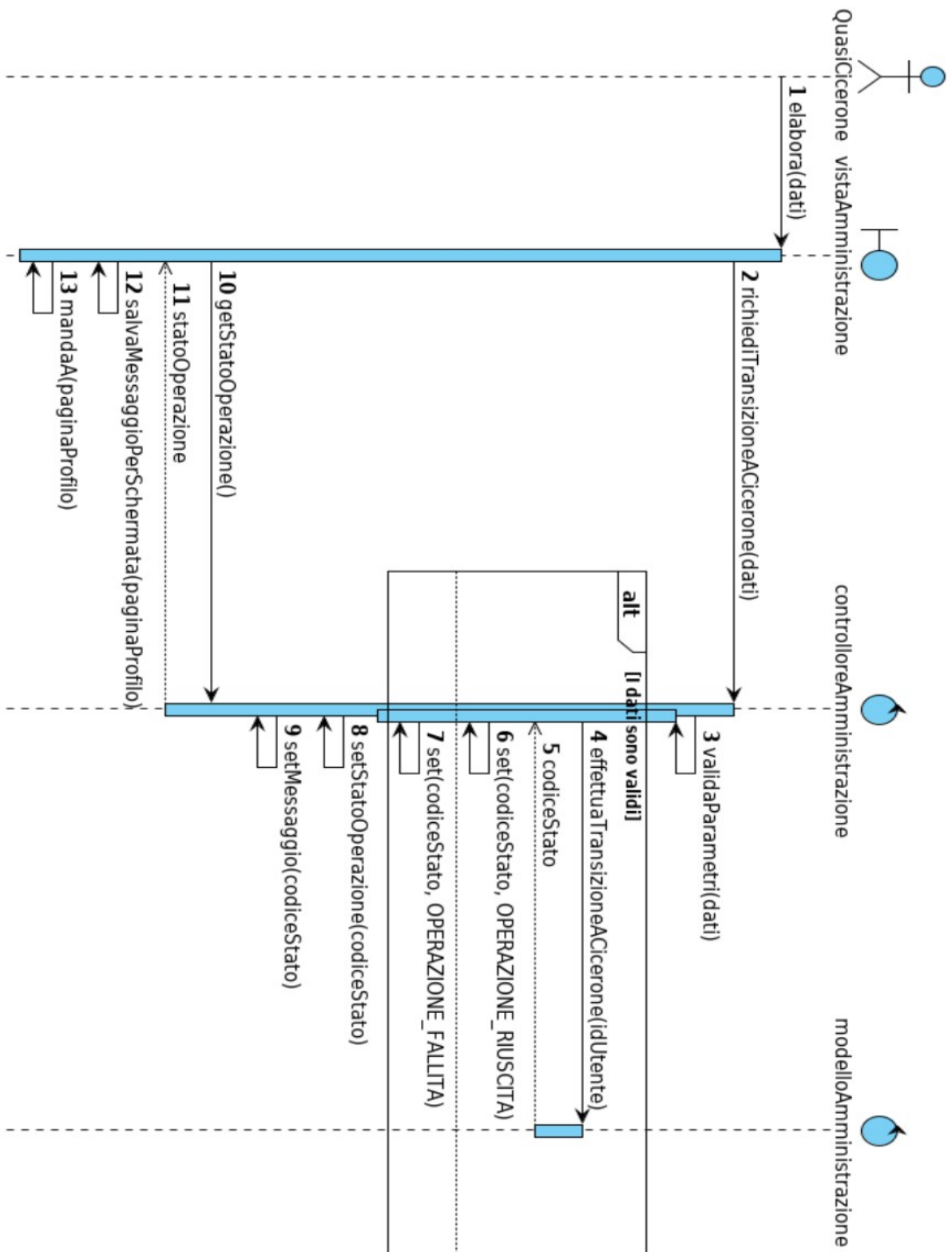


## SD\_13 - TrasformaInQuasiCicerone





## SD\_14 - ConfermaTransizioneCicerone



### 3.3.2 Corrispondenza tra Casi d'uso e Diagrammi di sequenza

<i>Casi d'uso</i>	<i>Diagramma di sequenza</i>
<u>CU 1, ALT 1.1</u>	<u>SD 1</u>
<u>CU 2, ALT 2.1</u>	<u>SD 2</u>
<u>CU 3</u>	<u>SD 3</u>
<u>CU 4</u>	<u>SD 4</u>
<u>CU 5, ALT 5.1, ALT 5.2, ALT 5.3</u>	<u>SD 5</u>
<u>CU 6</u>	<u>SD 6</u>
<u>CU 7, ALT 7.1</u>	<u>SD 7</u>
<u>CU 8</u>	<u>SD 8</u>
<u>CU 9</u>	<u>SD 9</u>
<u>CU 10</u>	<u>SD 10</u>
<u>CU 11</u>	<u>SD 11</u>
<u>CU 12</u>	<u>SD 12</u>
<u>CU 13</u>	<u>SD 13</u>
<u>CU 14</u>	<u>SD 14</u>


## 4. Data Design

### 4.1 Specifica dei Dati e dei Vincoli Informativi

Sono state individuate le entità concettuali Utente, Anagrafica, Richiesta d'aggiornamento, Richiesta di disiscrizione, Itinerario, Partecipazione, Feedback, Notifica e Preferenze di Notifica.

- Ad un Utente vengono associati un indirizzo email, un nome utente, una password, un'immagine (di profilo), una descrizione, un tipo, uno stato ed un codice d'attivazione. Due o più utenti registrati non possono avere lo stesso nome utente, ergo, ad ogni Utente, viene associato un nome utente univoco NON MODIFICABILE. Anche l'email deve essere diversa per ogni Utente, non possono esserci due utenti con lo stesso indirizzo. Il tipo d'utente può essere: *globetrotter*, *quasicicerone*, *cicerone*, *amministratore*. Lo stato dell'utente viene utilizzato per indicare il suo stato d'attivazione (dato che una volta registrato, non si può accedere direttamente). Tale stato può essere: *inserito*, *attivato*, *recuperando*.
- L'Anagrafica entra in gioco quando abbiamo a che fare con un utente Cicerone, o con un Globetrotter che ha inviato una richiesta d'aggiornamento. Ad una Anagrafica vengono associati un nome, un cognome, una data di nascita, un luogo di nascita, un numero di telefono, ed un codice fiscale. Ad ogni Cicerone e Globetrotter che ha inviato la richiesta d'aggiornamento corrisponde un'unica anagrafica.
- Ad una Richiesta di disiscrizione viene associata una descrizione. È in relazione uno a uno con un fruitore.
- Ad una Richiesta di aggiornamento è associata un'anagrafica, con la quale è in relazione uno ad uno.
- Ad un Itinerario vengono associati una data e ora (appuntamento), una descrizione, una lingua parlata, un'immagine (usata per rappresentare visivamente il luogo o l'attività), un luogo (la meta dell'appuntamento), una popolarità, una valuta, un compenso ed uno stato. Il compenso viene memorizzato come intero, per semplificare l'elaborazione, tale valore dovrà essere maggiore o uguale a 0. Il significato effettivo del campo compenso, sarà dipendente dalla sua

valuta.

La popolarità è un intero compreso tra 1 e 5 (inclusi).  
Lo stato può avere i valori: *aperto, itinere, concluso, chiuso*.

- Ad una Partecipazione vengono associati un itinerario, un fruitore ed uno stato.  
Ad un'unica associazione corrisponde un'unica coppia itinerario-partecipante, in altri termini, non possono esserci più associazioni in cui si ha lo stesso itinerario e lo stesso partecipante.  
Lo stato può avere i valori: *accordanda, accordata, annullanda*.
- Ad un Feedback vengono associati un itinerario, un partecipante, una descrizione, un voto ed un tipo.  
Il voto è un valore intero compreso tra 0 e 5.  
Il tipo può avere i valori: *partecipante-organizzatore, organizzatore-partecipante*.

La terna itinerario, fruitore e tipo è unica, nel senso che non si possono avere diversi feedback in cui itinerario, fruitore e tipo abbiano gli stessi valori.

Per stabilire "il verso" del feedback, si utilizza il campo *tipo*.  
Detto semplicemente, se il feedback è di tipo *partecipante-organizzatore*, allora tale feedback è stato rilasciato dal partecipante specificato nei confronti dell'organizzatore, se al contrario è di tipo *organizzatore-partecipante*, esso è stato rilasciato dall'organizzatore, nei confronti del partecipante specificato.

Esempio: Supponendo, di avere il partecipante "pinco", l'organizzatore "pallino" e l'itinerario "viaggio in montagna", è consentito avere i seguenti feedback:

- partecipante: "pallino", itinerario: "viaggio in montagna", descrizione: "pinco è un buon organizzatore", tipo: *partecipante-organizzatore*
- partecipante: "pallino", itinerario: "viaggio in montagna", descrizione: "pallino è un buon partecipante", tipo: *organizzatore-partecipante*

Come è possibile constatare, i due feedback differiscono tra loro, tralasciando la descrizione, soltanto per il tipo.  
Non è consentito avere feedback come i seguenti, ovvero feedback in cui partecipante, itinerario e tipo siano uguali:

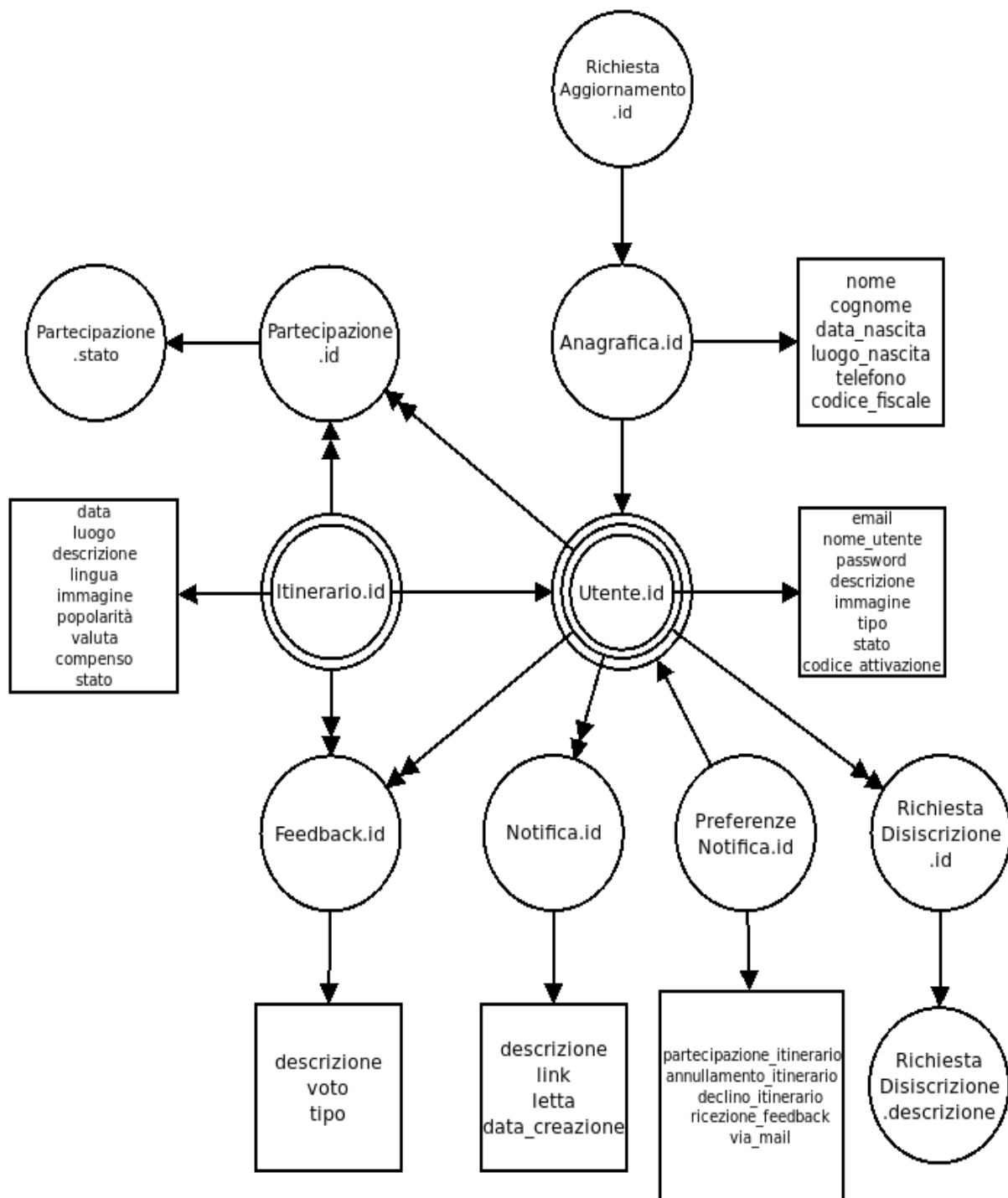
- partecipante: "pallino", itinerario: "viaggio in montagna", descrizione: "pinco è un buon organizzatore", tipo: *partecipante-organizzatore*

- partecipante: "pallino", itinerario: "viaggio in montagna", descrizione: "pallino è un buon partecipante", tipo: *partecipante-organizzatore*

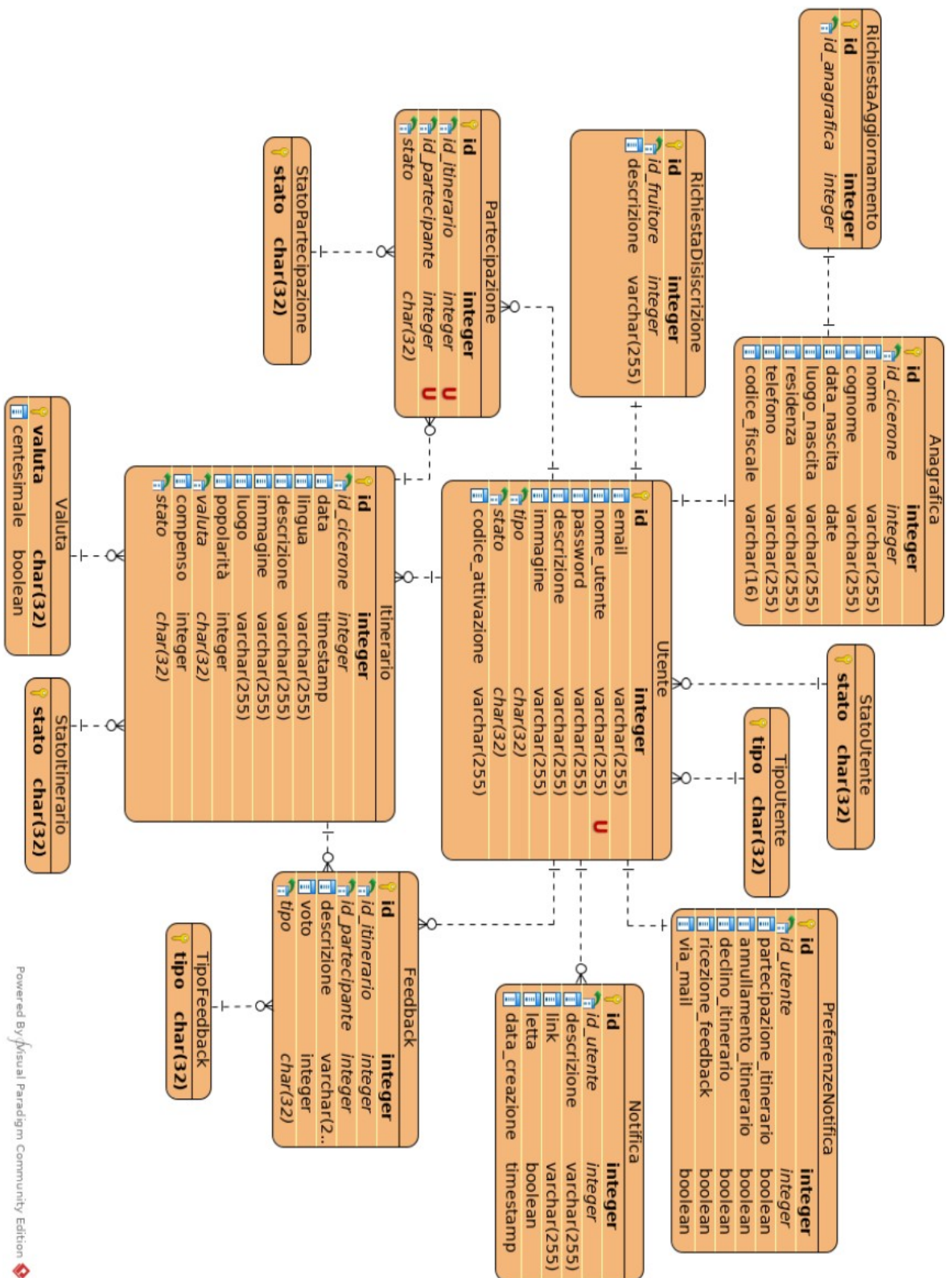
Ad una Notifica vengono associati un utente registrato, una descrizione, un link, uno stato di lettura e una data di creazione.

Ad una Preferenze di Notifica vengono associati un utente registrato e i campi di accordata partecipazione all'itinerario, annullamento partecipazione all'itinerario, declino partecipazione all'itinerario, ricezione di feedback, ricezione notifiche via email. Ad ogni utente registrato corrisponde un'unica istanza delle preferenze di notifica. Agli amministratori è concesso utilizzare solo il campo ricezione notifiche via mail.

#### 4.1.1 Diagramma delle Dipendenze dei Dati



## 4.1.2 Modello del Database





### 4.1.3 Lista delle dipendenze

- Utente.tipo dipende da TipoUtente.tipo (dipendenza *N a 1*)
- Anagrafica.id\_cicerone dipende da Utente.id (dipendenza *1 a 1*)
- PreferenzeNotifica.id\_utente dipende da Utente.id (dipendenza *1 a 1*)
- Notifica.id\_utente dipende da Utente.id (dipendenza *N a 1*)
- RichiestaDisiscrizione.id\_fruitore dipende da Utente.id (dipendenza *1 a 1*)
- RichiestaAggiornamento.id\_anagrafica dipende da Anagrafica.id (dipendenza *1 a 1*)
- Itinerario ha le seguenti dipendenze:
  - id\_cicerone dipende da Utente.id (dipendenza *N a 1*)
  - valuta dipende da Valuta.valuta (dipendenza *N a 1*)
  - stato dipende da StatoItinerario.stato (dipendenza *N a 1*)
- Partecipazione ha le seguenti dipendenze:
  - id\_itinerario dipende da Itinerario.id (dipendenza *N a 1*)
  - id\_partecipante dipende da Utente.id (dipendenza *N a 1*)
  - stato dipende da StatoPartecipazione.stato (dipendenza *N a 1*)
- Feedback ha le seguenti dipendenze:
  - id\_itinerario dipende da Itinerario.id (dipendenza *N a 1*)
  - id\_partecipante dipende da Utente.id (dipendenza *N a 1*)
  - tipo dipende da TipoFeedback.tipo (dipendenza *N a 1*)

### 4.1.4 Dettaglio dei Dati

Siano X ed Y campi della tabella T in esame e Z un campo della tabella F diversa dalla tabella T.

Seguono qui, i vincoli esprimibili nella sezione “vincoli” delle tabelle:

Nulla	Il campo X può essere nullo
Vuoto	Il campo (di tipo testuale) X può assumere come valore la stringa vuota
NONullo	Il campo X NON può essere nullo
Chiave unica	Il campo X deve essere unico (non

	possono esserci due o più righe in cui il campo X ha lo stesso valore). Se X ed Y (ed eventualmente ulteriori campi di T) sono identificati come unici, allora non possono esserci righe in cui si ha la stessa combinazione di valori di X, Y ed eventuali altri.
Chiave primaria	Indica che il campo X è chiave primaria della tabella in esame
Chiave esterna a F.Z	Indica che il campo X è chiave esterna e fa riferimento al campo Z della tabella F.

## Tabella TipoUtente

Campo	Tipo	Vincoli	Descrizione
tipo	char(32)	Chiave primaria	Indica il tipo di utente registrato

Alla creazione del DB, verranno inseriti i seguenti valori: *globetrotter*, *cicerone*, *quasicicerone*, *amministratore*.

## Tabella StatoUtente

Campo	Tipo	Vincoli	Descrizione
stato	char(32)	Chiave primaria	Indica lo stato dell'utente registrato

Alla creazione del DB, verranno inseriti i seguenti valori: *inserito*, *attivato*, *recuperando*.

## Tabella StatoItinerario

Campo	Tipo	Vincoli	Descrizione
stato	char(32)	Chiave primaria	Indica lo stato in cui si trova un certo itinerario

Alla creazione del DB, verranno inseriti i seguenti valori: *aperto, itinere, concluso, chiuso*.

## Tabella StatoPartecipazione

Campo	Tipo	Vincoli	Descrizione
stato	char(32)	Chiave primaria	Indica lo stato di una particolare richiesta di partecipazione

Alla creazione del DB, verranno inseriti i seguenti valori: *accordanda, accordata, annullanda*.

## Tabella TipoFeedback

Campo	Tipo	Vincoli	Descrizione
tipo	char(32)	Chiave primaria	Indica il tipo di feedback rilasciato

Alla creazione del DB, verranno inseriti i seguenti valori: *partecipante-organizzatore, organizzatore-partecipante*.

## Tabella Valuta

Campo	Tipo	Vincoli	Descrizione
-------	------	---------	-------------

valuta	char(32)	Chiave primaria	Indica la particolare valuta
centesimale	boolean	NONullo	Sta ad indicare se la valuta prevede il conteggio dei centesimi

## Tabella Utente

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico dell'utente registrato
email	varchar(255)	NONullo	È l'email associata all'utente registrato
nome_utente	char(32)	Chiave unica NONullo	È il nome utente associato all'utente registrato
password	char(32)	NONullo	È la password associata all'utente registrato
immagine	varchar(255)	Nullo	È il nome del file immagine associato all'immagine utilizzata dell'utente registrato
descrizione	varchar(255)	NONullo	È una breve descrizione

tipo		Chiave esterna a TipoUtente.tipo	Indica il tipo di utente registrato
stato		Chiave esterna a StatoUtente.tipo	Indica lo stato dell'utente registrato
codice_attivazione	varchar(255)	NONullo	È il codice utilizzato per l'attivazione dell'utente

Il campo *codice\_attivazione*, verrà utilizzato sia per attivare l'utente registrato affinché si possa iniziare ad utilizzarlo (il campo stato è impostato a *inserito*), sia nel caso sia necessario il recupero dell'accesso (il campo stato è impostato a *recuperando*).

## Tabella Anagrafica

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico dell'anagrafica associata ad un Cicerone
id_cicerone		Chiave unica Chiave esterna a Utente.id	È l'identificativo numerico dell'utente Cicerone associato a questa anagrafica
nome	varchar(255)	NONullo	È il nome proprio del Cicerone
cognome	varchar(255)	NONullo	È il cognome del Cicerone
data_nascita	date	NONullo	È la data di nascita del

			Cicerone
luogo_nascita	varchar(255)	NONullo	È il luogo di nascita del Cicerone
residenza	char(10)	NONullo	È il luogo di residenza del Cicerone
telefono	char(32)	NONullo	È il numero di telefono del Cicerone
codice_fiscale	char(16)	NONullo	È il codice fiscale del Cicerone

Questa tabella usa una politica di cancellazione CASCADE. In questa maniera, se un Cicerone/Globetrotter/QuasiCicerone viene rimosso, la corrispondente anagrafica verrà rimossa in automatico.

## Tabella *RichiestaDisiscrizione*

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico della richiesta d'amministrazione
id_fruitore		Chiave unica Chiave esterna a Utente.id	È l'utente che ha inviato la richiesta di disiscrizione
descrizione	varchar(255)	NONullo	È una breve descrizione

Questa tabella usa una politica di cancellazione CASCADE. In questa maniera, se un fruitore viene rimosso, la richiesta di disiscrizione inviata da lui verrà rimossa in automatico.

## Tabella RichiestaAggiornamento

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico della richiesta d'aggiornamento
id_anagrafica		Chiave unica Chiave esterna a Utente.id	È l'utente che ha inviato la richiesta di disiscrizione

Questa tabella usa una politica di cancellazione CASCADE. In questa maniera, se un'anagrafica viene rimossa, la richiesta d'aggiornamento inviata dal Globetrotter a cui tale anagrafica è associata, verrà rimossa in automatico.

Tabella *Itinerario*

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico dell'itinerario
id_cicerone		Chiave esterna a Utente.id	È il Cicerone che lo ha organizzato
data	timestamp	NONullo	È la data e ora in cui l'itinerario avrà luogo
descrizione	varchar(255)	NONullo	È una breve descrizione
lingua	varchar(255)	NONullo	È la lingua parlata dal Cicerone organizzatore
luogo	varchar(255)	NONullo	È la meta da raggiungere per partecipare all'itinerario
immagine	varchar(255)	NONullo	È un immagine utilizzata per spiegare visivamente il luogo o l'attività che riguarda l'itinerario
popolarità	integer	NONullo	È la popolarità dell'itinerario
valuta	char(32)	Chiave esterna a Valuta.valuta	È la valuta utilizzata per il compenso
compenso	integer	NONullo	È il compenso richiesto dal Cicerone



stato	char(32)	Chiave esterna a StatoItinerario.stato	È lo stato in cui si può trovare l'itinerario.
-------	----------	--	--

Il campo *popolarità* può avere un valore che va da 1 a 5 (inclusi).

Il campo *compenso* deve essere maggiore o uguale a 0. Se vale 0, la partecipazione all'itinerario è gratuita. L'effettiva interpretazione del valore dipenderà dal campo *valuta*.

Questa tabella usa una politica di cancellazione CASCADE. In questa maniera, se un Cicerone viene rimosso, gli itinerari creati da lui verranno rimossi in automatico.

## Tabella Partecipazione

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico dell'associazione
id_itinerario	integer	Chiave unica Chiave esterna a Itinerario.id	È l'identificativo numerico dell'itinerario
id_partecipante	integer	Chiave unica Chiave esterna a Utente.id	È l'identificativo numerico del fruitore partecipante
stato	char(32)	Chiave unica Chiave esterna a StatoPartecipazione.stato	Indica lo stato della richiesta di partecipazione.

Questa tabella usa una politica di cancellazione CASCADE sia su *id\_itinerario*, che su *id\_partecipante*. Se un itinerario viene rimosso, tutte le associazioni itinerario-partecipante di cui fa parte quell'itinerario verranno rimosse. Se un partecipante viene rimosso, tutte le associazioni itinerario-partecipante di cui fa parte quel partecipante verranno rimosse.

## Tabella *Feedback*

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico del particolare feedback
id_itinerario	integer	Chiave unica Chiave esterna a Itinerario.id	È l'identificativo numerico dell'itinerario
id_partecipante	integer	Chiave unica Chiave esterna a Utente.id	È l'identificativo numerico del fruitore partecipante. Rappresenta il partecipante all'itinerario, rispetto al quale, questo feedback viene rilasciato
descrizione	varchar(255)	NONullo	È una breve descrizione
voto	integer	NONullo	È il voto assegnato all'organizzatore o al partecipante.
tipo		Chiave unica Chiave esterna a TipoFeedback.tipo	È il tipo di feedback

Questa tabella usa una politica di cancellazione CASCADE sia su *id\_itinerario*, che su *id\_partecipante*.  
 Se un itinerario viene rimosso, tutti i feedback associati a quell'itinerario verranno rimossi.  
 Se un partecipante viene rimosso, tutti i feedback associati quel partecipante verranno rimossi.

Notiamo che se un feedback è di tipo *partecipante-organizzatore*, significa che è stato il partecipante specificato nel feedback a rilasciarlo nei confronti del Cicerone organizzatore, al contrario, se è di tipo *organizzatore-partecipante*, significa che è stato l'organizzatore dell'itinerario a rilasciarlo nei confronti del partecipante specificato.

Esempio: Supponendo di avere un *id\_partecipante* pari a 87, le seguenti righe sono valide:

- *id*: 0, *id\_partecipante*: 87, *descrizione*: "<qualcuno> è un buon organizzatore", *voto*: 4, *tipo*: *partecipante-organizzatore*
- *id*: 1, *id\_partecipante*: 87, *descrizione*: "<qualcun'altro> è un buon partecipante", *voto*: 4, *tipo*: *organizzatore-partecipante*

Queste invece NON SONO VALIDE:

- *id*: 0, *id\_partecipante*: 87, *descrizione*: "<qualcuno> è un buon organizzatore", *voto*: 4, *tipo*: *partecipante-organizzatore*
- *id*: 1, *id\_partecipante*: 87, *descrizione*: "<qualcun'altro> è un buon partecipante", *voto*: 4, *tipo*: *partecipante-organizzatore*

## Tabella Notifica

Campo	Tipo	Vincoli	Descrizione
<i>id</i>	integer	Chiave primaria	È l'identificativo numerico della richiesta d'amministrazione
<i>id_utente</i>	integer	Chiave esterna a <i>Utente.id</i>	È l'utente registrato a cui la notifica è associata
<i>descrizione</i>	varchar(255)	NONullo	È una breve descrizione
<i>link</i>	varchar(255)	NONullo	È l'URL contenuto nella notifica
<i>letta</i>	boolean	NONullo	Indica se la notifica è stata letta

data_creazione	timestamp	NONullo	Indica la data di creazione (ed invio) della notifica
----------------	-----------	---------	---

Questa tabella usa una politica di cancellazione CASCADE. Se un utente viene rimosso, tutte le notifiche associate a quell'utente verranno rimosse.

## Tabella *PreferenzeNotifica*

Campo	Tipo	Vincoli	Descrizione
id	integer	Chiave primaria	È l'identificativo numerico della particolare preferenza
id_utente	integer	Chiave unica Chiave esterna a Utente.id	È l'utente registrato a cui la preferenza è associata
partecipazione_itinerario	boolean	NONullo	Indica se si vogliono ricevere notifiche in caso di richiesta di partecipazione accordata.
annullamento_itinerario	boolean	NONullo	Indica se si vogliono ricevere notifiche in caso di richiesta di partecipazione annullata.
declino_itinerario	boolean	NONullo	Indica se si vogliono ricevere notifiche in caso di richiesta di partecipazione

			declinata.
ricezione_feedback	boolean	NONullo	Indica se si vogliono ricevere notifiche in caso di richiesta di ricezione di feedback.
via_mail	boolean	NONullo	Indica se si vogliono ricevere notifiche anche via email.

Questa tabella usa una politica di cancellazione CASCADE. Se un utente viene rimosso, vengono rimosse anche le corrispondenti preferenze di notifica in automatico.

## 4.2 File System

- accesso.php
- amministrazione.php
- chiamo.php
- controllore/
  - Controllore.php
  - ControlloreAccesso.php
  - ControlloreAmministrazione.php
  - ControlloreSessione.php
  - ControlloreTest.php
  - ControlloreUtente.php
  - LettoreSessione.php
  - ScrittoreSessione.php
  - Sessione.php
- dbms\_setup/
  - creazione\_db.sql
  - datitest\_db.sql
- debug/
  - debug.php
  - DebugSettings.php
  - pagesAndPresets.js
  - rmdir/
- immagini/
  - itinerari/
  - utenti/
    - default.png
- index.php
- inizializza.sh
- modello/
  - entità
    - Anagrafica.php
    - EntitàDB.php
    - RichiestaAggiornamento.php
    - RichiestaDisiscrizione.php
    - RichiestaAmministrazione.php
    - RichiesteAmministrazione.php
    - Utente.php
  - CiceroneDatabase.php
  - Modello.php
  - ModelloAmministrazione.php
  - ModelloSessione.php

- ModelloTest.php
  - ModelloUtente.php
- profilo.php
- sonar-project.properties
- test.php
- utils/
  - CustomException.php
  - GestoreDatabase.php
  - GestoreSession.php
  - Nodo.php
  - NodoPrevisitaIterator.php
  - Outputtabile.php
  - parser/
    - Lexer.php
    - Parser.php
    - Token.php
  - PHPMailer/
  - Pila.php
  - Spedizioniere.php
  - template/
    - NodoOutput.php
    - NodoOutputIterator.php
    - parser/
      - TemplateLexer.php
      - TemplateParser.php
      - TemplateToken.php
    - Template.php
- vista/
  - immagini/
    - chiusura.png
    - cicerone.jpg
    - sidebar.png
  - Layout.php
  - Popup.php
  - SchermataChiSiamo.php
  - template/
    - chi\_siamo.html
    - debug/
      - debug\_utils.css
      - debug\_utils.html
      - debug\_utils.js
      - test.html
      - test.js
    - form/

- accesso/
  - accesso.css
  - accesso.html
  - accesso.js
- amministrazione/
  - amministrazione.css
  - amministrazione.html
  - amministrazione.js
  - richiesta.css
  - richiesta.html
- utente/
  - profilo.css
  - profilo.html
- layout/
  - layout.css
  - layout.html
  - layout.js
  - sidebar.css
  - sidebar.html
  - sidebar.js
- popup/
  - popup.css
  - popup.html
  - popup.js
- TriplettaSemplice.php
- TriplettaTemplate.php
- Vista.php
- VistaAccesso.php
- VistaAmministrazione.php
- VistaTest.php
- VistaUtente.php

#### 4.2.1 Descrizione Struttura del File System

- *accesso.php*, *amministrazione.php*, *profilo.php*, *chiamo.php*, *index.php*, *test.php* sono file che generano la schermata che compete loro. Al loro interno viene usato il metodo statico *entrypoint* della classe Layout con un particolare parametro (diverso per ogni file), che consente la generazione di una determinata pagina e dunque la visualizzazione da parte di un utente.
- *sonar-project.properties* è il file di configurazione per SonarQube.
- *inizializza.sh* è uno script Bourne shell che dapprima imposta i permessi corretti sui file e cartelle del sistema Cicerone, ed infine, sfrutta i file presenti in *dbms\_setup* per creare ed inizializzare il database, creando le tabelle ed inserendo dei dati di test.



- La cartella *immagini* conterrà le immagini di profilo degli utenti (sottocartella *immagini/utenti*) e le immagini associate agli itinerari (sottocartella *immagini/itinerari*).
- La cartella *controllore* contiene tutti i file che implementano ciò che appartiene al package *controllore*. Ogni file al suo interno implementa/definisce la rispettiva classe/interfaccia con lo stesso nome (ad esempio, *ControlloreVista.php*, implementa la classe *ControlloreVista*).
- La cartella *debug* contiene i file utilizzati esclusivamente per il debugging, in particolare, *DebugSettings.php* e *debug.php*. Il primo file implementa la classe *DebugSettings* il secondo invece viene utilizzato mediante chiamata AJAX per abilitare/disabilitare la modalità debug per le chiamate header("location: url").
- La cartella *modello* molti file e una cartella.
  - La cartella *entità* contiene al suo interno i file che implementano le classi adeguate a rappresentare i dati persistenti del sistema Cicerone (*Utente.php*, *Anagrafica.php*, e così via...).
  - I file implementano ciò che appartiene al package *modello*. Ogni file al suo interno implementa la rispettiva classe con lo stesso nome (ad esempio, *CiceroneDatabase.php*, implementa la classe *CiceroneDatabase*).
- La cartella *utils* contiene molti file e tre cartelle.
  - La prima cartella è chiamata *PHPMailer* e contiene al suo interno i file che implementano la libreria che porta lo stesso nome. Tale libreria consente di spedire email affidandosi a server SMTP esterni.
  - La seconda cartella è chiamata *parser*, contiene tutti i file che implementano ciò che appartiene al package *utils/parser*. Ogni file al suo interno implementa la rispettiva classe con lo stesso nome.
  - La terza cartella è chiamata *template*, contiene tutti i file che implementano il package *utils/template* ed una cartella denominata *parser*. Ogni file al suo interno implementa la rispettiva classe con lo stesso nome. Eccezione a questa regola è il file *Template.php* che oltre a contenere l'implementazione della classe *Template*, contiene l'implementazione delle classi *TemplateFileNotFoundException* e *TemplateChiaveNonTrovataException* e *TemplateErroreInterpretazioneException*.
  - La cartella *utils/template/parser* inclusa nella terza cartella, contiene tutti i file che implementano il package *utils/template/parser*. Ogni file al suo interno implementa la rispettiva classe con lo stesso nome.
  - I file implementano ciò che appartiene al package *utils*. Ogni file implementa/definisce al suo interno la rispettiva classe/interfaccia che porta lo stesso nome (ad esempio, *Outputtabile.php*, definisce l'interfaccia *Outputtabile*). Eccezioni a questa regola sono i file *Pila.php*, *GestoreDatabase.php* e *Nodo.php* che oltre a contenere l'implementazione, rispettivamente, delle

classi `Pila`, `GestoreDatabase` e `Nodo`, contengono anche, sempre rispettivamente, l'implementazione delle classi `PilaVuotaException`, `ConnessioneDBException`, `ComandoSQLException` e `NodoFiglioInesistenteException`.

- La cartella *vista* contiene molti file e molte cartelle.
  - La cartella *immagini* contiene le immagini utilizzate dal sistema (in particolare la foto della scultura di Marco Tullio Cicerone, usata nel titolo).
  - La cartella *template* contiene i vari file Template utilizzati dal sistema per produrre le sue schermate. Tali file sono organizzati in diverse sottocartelle per avere un po' d'ordine.
  - Ogni file al suo interno implementa la rispettiva classe con lo stesso nome. Eccezione a questa regola è il file *TriplettaSemplice.php*, che oltre a contenere l'implementazione della classe `TriplettaSemplice`, contiene anche, l'implementazione della classe `ComponenteTriplettaNonValidoException`.

#### 4.2.2 Altro.....

## 5. Glossario

### 5.1 Acronimi

Acronimo	Significato
MVC	Model, View, Controller
DB	Database
URL	Uniform Resource Locator
SMTP	Simple Mail Transfer Protocol
AJAX	Asynchronous JavaScript And XML
DFS	Depth First Search

### 5.2 Definizioni

Vocabolo	Sinonimi	Significato
Sistema	Sito, piattaforma, servizio online	È il sistema "Cicerone".
Utente		Con questo termine si indica un qualunque tipo di utente del sistema. Può essere un utente ospite, o un utente registrato.
Ospite	Navigante, utente anonimo	Con questo termine si indicano quegli utenti che non essendo registrati (o non avendo eseguito l'accesso) , non possono usufruire di tutte le funzionalità del

		sistema.
Amministratore	Gestore	È un utente che legge le richieste d'amministrazione inviate dai fruitori e prende provvedimenti in base ad esse.
Utente registrato		È un utente del sistema che appartiene alle categorie "Globetrotter", "Cicerone" o "amministratore".
Profilo		È un insieme di informazioni associate ad un utente registrato. Tale associazione viene creata all'atto della registrazione dell'utente al sistema.
Fruitore	Utente registrato [non amministratore]	È un utente registrato NON amministratore, quindi, può essere solo "Globetrotter" o "Cicerone".
Globetrotter	Viaggiatore, turista	È un utente del sistema che partecipa ad un itinerario.
Cicerone	Organizzatore, guida	È un utente del sistema che organizza itinerari ed accompagna i fruitori partecipanti.
QuasiCicerone		Era un utente Globetrotter. Dopo che gli è stata accordata la richiesta d'aggiornamento, è divenuto un

		<p>QuasiCicerone.</p> <p>Per completare la transizione a Cicerone deve dare conferma lui stesso.</p> <p>Ad eccezione di questo particolare, ha le stesse funzioni di un Globetrotter.</p>
Itinerario	Escursione, attività, gita, passeggio	È l'attività a cui i fruitori partecipanti e il Cicerone che l'ha organizzata prendono parte.
Popolare	Mainstream, tradizionale, molto frequentato	Detto di un itinerario, indica che esso è molto frequentato da turisti. Più l'itinerario NON È popolare, più È stimolante culturalmente.
Feedback	Opinione, esperienza, recensione	È una recensione che i fruitori rilasciano ad itinerario concluso.
Feedback partecipante-organizzatore	Feedback ricevuto da un organizzatore (inviato da un partecipante)	È il feedback rilasciato da un fruitore partecipante nei confronti di un Cicerone organizzatore, rispetto ad un itinerario concluso.
Feedback organizzatore-partecipante	Feedback ricevuto da un partecipante (inviato da un organizzatore)	È il feedback rilasciato da un Cicerone organizzatore nei confronti di un fruitore partecipante, rispetto ad un itinerario concluso.
Luogo	Meta, destinazione, posto, locazione	Il posto in cui avverrà l'itinerario organizzato

		dal Cicerone. È la meta dei fruitori.
Partecipante	Globetrotter, Cicerone [non organizzatore], fruitore	Detto di un fruitore rispetto ad un itinerario, sta ad indicare che tale fruitore partecipa (o ha richiesto di partecipare) all'itinerario.
Organizzatore		Detto di un Cicerone rispetto ad un itinerario, sta ad indicare che tale Cicerone è colui che lo ha organizzato.
Richiesta di partecipazione		È la richiesta che un fruitore invia ad un Cicerone organizzatore di un itinerario
Richiesta d'annullamento		È la richiesta che un fruitore manda ad un Cicerone organizzatore per chiedere di annullare una precedente richiesta di partecipazione che il Cicerone ha già accordato.
Richiesta d'amministrazione		È una richiesta che un fruitore manda ad un amministratore.
Richiesta di aggiornamento		È una richiesta d'amministrazione. È la richiesta che un Globetrotter invia ad un amministratore per richiedere di diventare Cicerone.
Richiesta di disiscrizione		È una richiesta d'amministrazione. È la richiesta che un

		fruitore invia ad un amministratore per richiedere l'annullamento dell'iscrizione.
Prendere parte [ad un itinerario]	Partecipare	Si usa per indicare che un fruitore partecipa ad un itinerario. Inserito per evitare la ripetizione del verbo "partecipare" (concetto già usato in "Richiesta di partecipazione").
Accordare [una richiesta di partecipazione]	Accettare	Detto di una richiesta di partecipazione, è l'atto di un Cicerone organizzatore di accettarla, inserendo così il partecipante nella lista dei partecipanti.
Declinare [una richiesta di partecipazione]	Rifiutare	Detto di una richiesta di partecipazione, è l'atto di un Cicerone organizzatore di rifiutarla, NON inserendo dunque il partecipante nella lista dei partecipanti.
Annullare [una richiesta di partecipazione]	Cancellare, rimuovere	Detto di una richiesta di partecipazione accordata, è l'atto di un Cicerone organizzatore di annullarla, rimuovendo dunque il partecipante dalla lista dei partecipanti.
Recuperare [l'accesso]		Con questo verbo viene indicata l'azione che il sistema mette in atto,

		quando l'ospite chiede al sistema di poter accedere, sebbene egli abbia perso o non ricorda più le credenziali.
Attivare [un utente registrato]		È l'atto di cambiare il suo stato da <i>registrato</i> o <i>recupero</i> ad <i>attivato</i> .
linguaggio Template		È il linguaggio utilizzato dal sistema Cicerone per poter interpretare un file Template e quindi procedere alla generazione di un risultato.
file/stringa Template		È un file o una stringa, che contiene marcature del linguaggio Template.
tripletta [Template]		È una tripla di file Template che il sistema Cicerone utilizza per costruire le schermate. I tre file hanno estensione html, js e css.
Grammatica LLk		Nella teoria dei linguaggi formali, la Grammatica, detto in maniera semplice, rappresenta le frasi corrette di un linguaggio, in particolare una Grammatica LLk (Left-to-right Leftmost derivation) è un sottoinsieme delle grammatiche libere da contesto (context-free Grammars).



		<p>Quel k sta ad indicare il "lookahead", ovvero il numero di token (cioè i lessemi del linguaggio) da considerare durante l'operazione di analisi (parsing).</p> <p>In generale, per la creazione di un interprete di un linguaggio, si utilizza sia una macchina capace di riconoscere i vocaboli di un linguaggio (analizzatore lessicale o lexer), sia una macchina capace di stabilire se sono state scritte le frasi corrette del linguaggio (analizzatore sintattico o parser).</p>
previsita	visita con ordine anticipato	<p>Particolare tipo di visita DFS di un Albero (struttura dati) che effettua PRIMA la visita di un nodo padre e POI la visita dei rispettivi nodi figli (se esistono).</p> <p>È detta "previsita" in contrapposizione alla postvisita (PRIMA i nodi figli e POI il nodo padre) e alla invisita (PRIMA alcuni nodi figli, POI il padre, INFINE i rimanenti nodi figli). Vengono dette anche, rispettivamente visita con ordine differito e simmetrico.</p>

## 6. Appendice

### 6.1 .....

### 6.2 .....