

# SPRINT REPORT 3

## [Team Intuition]

### Cicerone

Versione 3.0

Data di rilascio:

16-07-2019

INGEGNERIA DEL SOFTWARE A.A. 2018-2019

#### **Realizzato da**

Caiati Giuseppe 643386 ITPS g.caiati4@studenti.uniba.it  
De Pasquale Davide 641348 ITPS d.depasquale5@studenti.uniba.it  
Fallacara Antonio 626713 ITPS a.fallacara8@studenti.uniba.it  
Napoli Giovanni 656473 ITPS g.napoli4@studenti.uniba.it

---

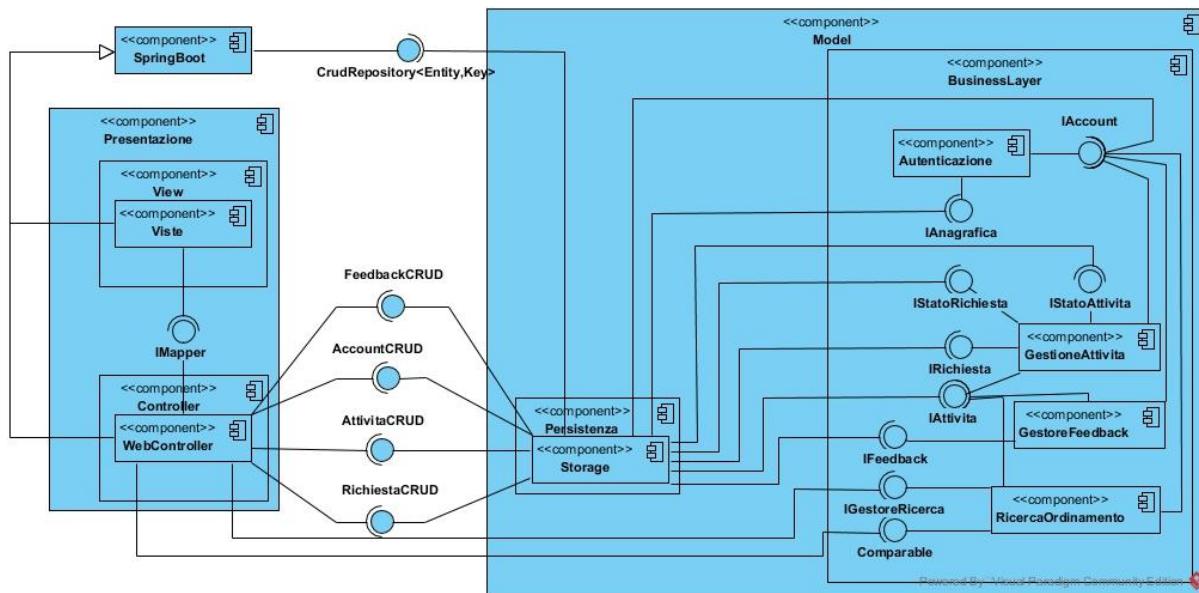
<b>1. SOFTWARE SYSTEM SKELETON .....</b>	<b>4</b>
1.1 DIAGRAMMA DELLE COMPONENTI .....	4
1.2 DESCRIZIONE DELLE COMPONENTI.....	4
1.3 DIAGRAMMA DI DEPLOY.....	6
<b>2. USER STORIES .....</b>	<b>7</b>
2.1 SPRINT BACKLOG .....	7
2.2 PRODUCT SPECIFICATION .....	12
2.2.1 <i>Casi d'uso</i> .....	12
2.2.2 <i>Descrizione testuale casi d'uso</i> .....	13
2.2.3 <i>Interfaccia</i> .....	40
Presentazione piattaforma ( <i>Home</i> ) ( <i>Sprint 1</i> ).....	40
Presentazione piattaforma ( <i>Home: Cos'è?</i> ) ( <i>Sprint 1</i> ) .....	41
Presentazione piattaforma ( <i>Home: Come funziona?</i> ) ( <i>Sprint 1</i> ).....	41
Presentazione piattaforma ( <i>Home: Area assistenza</i> ) ( <i>Sprint 1</i> ) .....	42
Registrazione Account ( <i>Sprint 1</i> ).....	42
Login Account ( <i>Sprint 1</i> ) .....	43
Menu di scelta funzionalità account post login ( <i>Sprint 1</i> ) .....	43
Modifica profilo - Pannello di controllo Cicerone e Globetrotter ( <i>Sprint 1</i> ) .....	44
<i>Crea attività</i> ( <i>Sprint 2</i> ) .....	44
<i>Modifica attività</i> ( <i>Sprint 2</i> ).....	45
<i>Modifica profilo</i> ( <i>Sprint 2</i> ).....	45
<i>Gestisci iscritti</i> ( <i>Sprint 2</i> ) .....	46
<i>Visualizza attività create</i> ( <i>Sprint 2</i> ) .....	46
<i>Visualizza catalogo attività</i> ( <i>Sprint 2</i> ).....	47
<i>Visualizza dettagli attività</i> ( <i>Sprint 2</i> ) .....	47
<i>Visualizza richieste create</i> ( <i>Sprint 2</i> ) .....	48
2.2.4 <i>Qualità</i> .....	49
<b>3. PRODUCT DESIGN .....</b>	<b>50</b>
3.1 DIAGRAMMA DELLE CLASSI .....	50
3.2 TABELLA DI TRACCIABILITÀ COMPONENTI-CLASSI .....	51
3.3 SPECIFICHE DEGLI ENUMERATIVI .....	53

---

3.4	SPECIFICHE DELLE CLASSI .....	53
3.5	DIAGRAMMI DI SEQUENZA .....	93
<b>4.</b>	<b>DATA DESIGN .....</b>	<b>132</b>
4.1	DATABASE .....	132
4.1.1	<i>Diagramma delle Dipendenze dei Dati .....</i>	<i>132</i>
4.1.2	<i>Modello del Database .....</i>	<i>133</i>
4.1.3	<i>Lista delle dipendenze e Dettaglio dei Dati .....</i>	<i>133</i>
4.2	FILE SYSTEM .....	137
<b>5.</b>	<b>GLOSSARIO .....</b>	<b>140</b>
5.1	ACRONIMI .....	140
5.2	DEFINIZIONI .....	140

# 1. SOFTWARE SYSTEM SKELETON

## 1.1 Diagramma delle Componenti



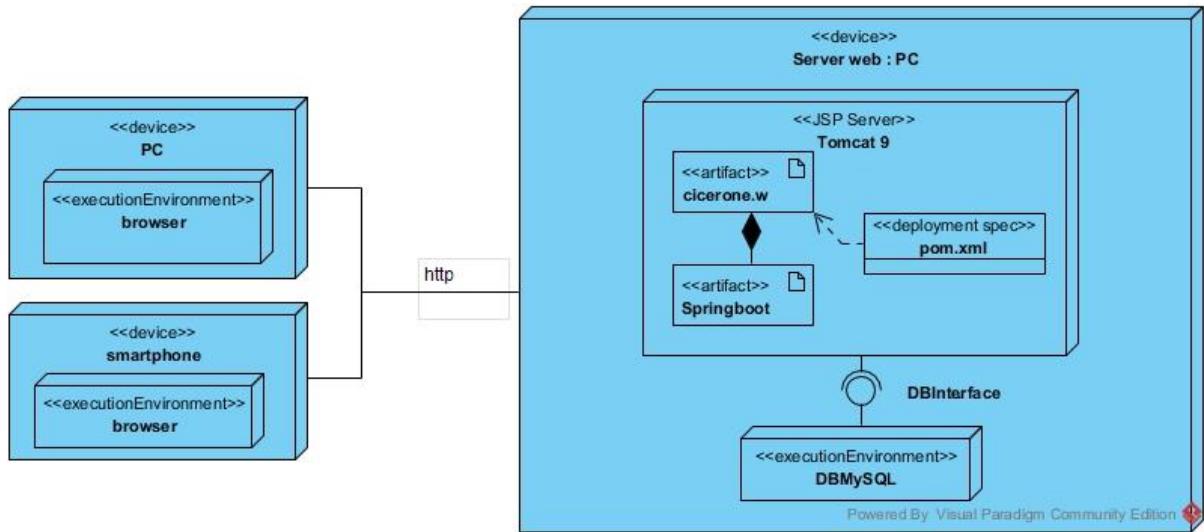
## 1.2 Descrizione delle componenti

Le componenti individuate che costituiscono il nostro sistema sono:

- **Spring Boot:** il sistema utilizza componenti che fanno parte di Spring framework in particolare Spring boot, framework che fornisce un modo semplice di inizializzare, configurare e ed eseguire semplici applicazioni web-based.
  - Offre: CRUDRepository;
- **Viste:** componente realizzato grazie ai componenti di Spring MVC. Esso racchiude le Server Page *jsp* che generano le viste relative ai modelli.
  - Utilizza: IMapper;
- **Storage:** componente creato per l'archiviazione dei dati persistenti; inoltre permette la mappatura delle entità che popolano il database con quelle del nostro dominio di business.
  - Offre: AccountCRUD, AttivitaCRUD, RichiestaCRUD;
  - Utilizza: CRUDRepository;

- 
- **WebController:** componente creato grazie ai componenti di Spring MVC. Il WebController è un insieme di classi che permettono la processazione dell'input del Client convertendolo in comandi per le viste e per i modelli.
    - Offre: IMapper;
    - Utilizza: AccountCRUD, AttivitaCRUD, RichiestaCRUD, IGestoreRicerca, Comparable;
  - **Autenticazione:** componente del sistema che realizza le funzionalità per la creazione, convalidazione di un account e la gestione dell'autenticazione sulla piattaforma;
    - Offre: IAccount, IAnagrafica;
  - **Gestione attività:** componente del sistema che permette di gestire le attività create dai ciceroni e le richieste di iscrizione dei globetrotter alle stesse;
    - Offre: IAttivita, IRichiesta, IStatoRichiesta, IStatoAttivita;
    - Utilizza: IAccount;
  - **GestoreFeedback:** componente del sistema che permette di gestire i feedback creati dai Globetrotter per le attività conclusive a cui hanno partecipato;
    - Offre: IFeedback;
    - Utilizza: IAccount, IAttivita;
  - **Ricerca e Ordinamento:** componente del sistema che permette di cercare, filtrare e ordinare le attività create dai ciceroni secondo alcuni criteri.
    - Offre: IGestoreRicerca, Comparable;
    - Utilizza: IAccount, IAttivita.

## 1.3 Diagramma di Deploy





## 2. USER STORIES

### 2.1 Sprint Backlog

Numero Sprint	Codice casi d'uso	Codice user story	Codice Item	Descrizione
1	2	1	Item 1	Realizzazione interfaccia UI che permette l'inserimento dei dati dell'utente.
1	2	1	Item 2	Creazione di un account con validazione dei campi immessi dall'utente e relativo controllo sulla pregressa esistenza dell'email.
1	2	1	Item 3	Memorizzazione mappatura dell'account all'interno del database.
1	1	2	Item 4	Realizzazione interfaccia UI che permetta di inserire i campi di email e password per accedere alla piattaforma.
1	1	2	Item 5	Verifica dell'autenticazione dell'account tramite il confronto degli hash tra il campo password immesso dall'utente e del campo password associato all'email inserita dall'utente (presenti nel database). L'utente è autenticato.
1	3	3	Item 6	Realizzazione interfaccia UI che permette la visualizzazione del profilo dell'account.
1	4	3	Item 7	Realizzazione interfaccia UI che permette la modifica dei campi

				del profilo dell'utente.
1	3	3	Item 8	Visualizzazione delle informazioni pubbliche di un account (memorizzato all'interno del database) tramite id associato.
1	4	3	Item 9	Modifica dei campi pubblici e privati tramite verifica dell'autenticazione dell'account utilizzando l'id.
1	4	3	Item 10	Salvataggio dei campi pubblici e privati dell'account nel database.
1	5	15	Item 11	Realizzazione dell'interfaccia UI della home della piattaforma.
2	9	4	Item 12	Realizzazione interfaccia UI in cui il Cicerone può inserire i dati relativi all'attività.
2	9	4	Item 13	Creazione di un'attività con validazione dei campi obbligatori (titolo, punto d'incontro, orario d'incontro, lingue parlate, retribuzione e descrizione).
2	9	4	Item 14	Memorizzazione mappatura dell'attività all'interno del database.
2	8	5	Item 15	Realizzazione interfaccia UI in cui il Cicerone può visualizzare i dati relativi alle attività da lui create.
2	10	6	Item 16	Realizzazione interfaccia UI per la modifica dei dati relativi all'attività con la validazione dei campi obbligatori (punto d'incontro e orario d'incontro).

2	17	7	Item 17	Realizzazione interfaccia UI per la gestione degli iscritti all'attività.
2	11	8	Item 18	Eliminazione della mappatura dell'attività all'interno del database.
2	6	11	Item 19	Modifica del ruolo del proprio account con passaggio delle funzionalità a globetrotter.
2	12,13,15	12,13,21	Item 20	Modifica della mappatura della richiesta all'interno del database.
2	13	13	Item 21	Realizzazione interfaccia UI per la gestione della declinazioni dei Globetrotter relative alle attività inserite dai Ciceroni.
2	7	14	Item 22	Invalidare la sessione relativa all'account.
2	16	19	Item 23	Realizzazione interfaccia UI per la visualizzazione delle informazioni delle attività svolte (memorizzate all'interno del database) tramite id associato.
2	14	16	Item 24	Realizzazione interfaccia UI per la visualizzazione delle attività create dal Cicerone.
2	18	20	Item 25	Realizzazione interfaccia UI per la visualizzazione dettagliata di una singola attività.
3	20	9	Item 26	Modifica della UI Cicerone Attività con aggiunta della ricerca
3	20	9	Item 27	Integrazione ricerca nel WebControllerCiceroneAttività
3	19	10	Item	Modifica UI CiceroneAttività con

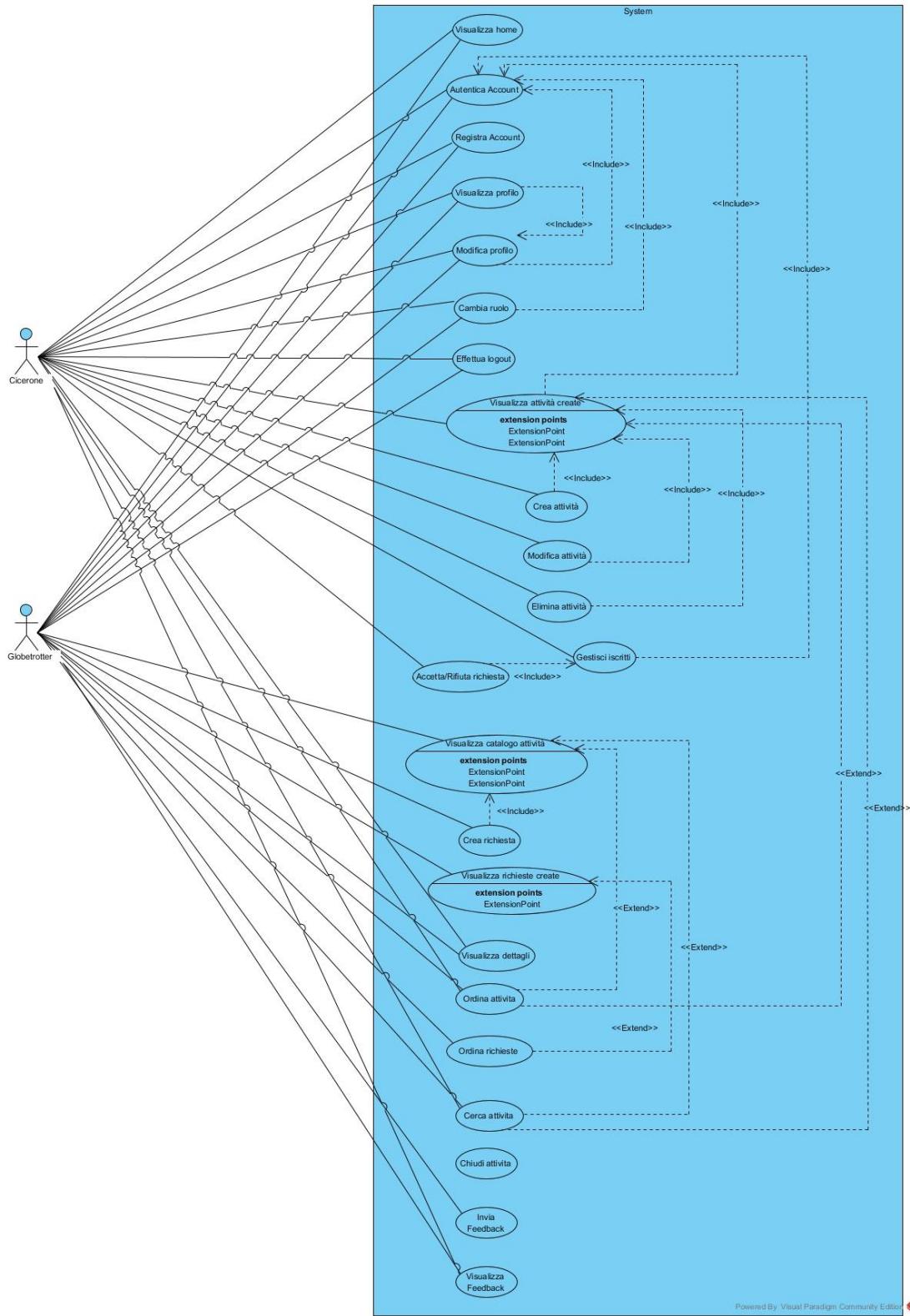
			28	I'aggiunta dell'ordinamento
3	19	10	Item 29	Integrazione ordinamento in WebControllerCiceroneAttivita
3	20	17	Item 30	Modifica della UI Globetrotter Attivita con aggiunta della ricerca
3	20	17	Item 31	Integrazione ricerca nel WebControllerGlobetrotterAttivita
3	19	18	Item 32	Modifica UI GlobetrotterAttivita con l'aggiunta dell'ordinamento
3	19	18	Item 33	Integrazione ordinamento in WebControllerGlobetrotterAttivita
3	21	22	Item 34	Modifica UI Gestione Richieste con la possibilità di ordinarle a piacimento
3	21	22	Item 35	Integrazione WebControllerGestioneRichieste dell'ordinamento delle richieste
3	23	23	Item 36	Modifica UI Gestione Richieste con l'inserimento e la visualizzazione del feedback relativo all'attività
3	23	23	Item 37	Integrazione nel WebControllerGestioneRichieste dell'inserimento del feedback
3	22	25	Item 38	Implementazione script schedulato per la chiusura automatica delle attività
3	24	24	Item 39	Modifica UI gestione richieste e visualizza dettagli per la visualizzazione del feedback relativo all'attività
3	24	24	Item 40	Integrazione nel WebControllerVisualizzaDettagli

---

				dell'inserimento del feedback
--	--	--	--	-------------------------------

## 2.2 Product Specification

### 2.2.1 Casi d'uso



## 2.2.2 Descrizione testuale casi d'uso

Nome	Autentica Account
Id	1
Breve descrizione	Il Globetrotter/Cicerone si autentica sulla piattaforma.
Attori primari	Globetrotter, Cicerone
Attori secondari	nessuno
Precondizioni	Il Globetrotter/Cicerone sia registrato sulla piattaforma
Sequenza principale degli eventi	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "Login".</li> <li>2 Il sistema chiede al Globetrotter/Cicerone di inserire le sue informazioni: email e password associata.</li> <li>3 Il sistema valida le informazioni del Globetrotter/Cicerone.</li> <li>4 Il sistema autentica l'account del Globetrotter/Cicerone.</li> </ol>
Post-condizioni	Il Globetrotter/Cicerone è autenticato sulla piattaforma
Sequenza alternativi degli eventi	EmailNonValida PasswordNonValida

Nome	Registra Account
Id	2

<b>Breve descrizione</b>	Il sistema crea un nuovo account per il Globetrotter/Cicerone
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	Nessuno.
<b>Precondizioni</b>	Nessuna.
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "Registrati".</li> <li>2 Il Globetrotter/Cicerone inserisce le sue informazioni personali tra cui email, password associata, conferma password, città di residenza, genere, data di nascita, nome e cognome.</li> <li>3 Il sistema valida le informazioni del Globetrotter/Cicerone.</li> <li>4 Il sistema registra l'account del Globetrotter/Cicerone.</li> </ol>
<b>Post-condizioni</b>	Il Globetrotter/Cicerone ha registrato un account sulla piattaforma
<b>Sequenza alternativi degli eventi</b>	EmailNonValida PasswordNonValida CittaNonValida NomeNonValido CognomeNonValido NumeroDiTelefonoNonValido

Nome	Visualizza profilo
<b>Id</b>	3
<b>Breve descrizione</b>	Il Globetrotter/Cicerone visualizza i dati relativi ad un account e al suo profilo.
<b>Attori primari</b>	Globetrotter, Cicerone

<b>Attori secondari</b>	Nessuno.
<b>Precondizioni</b>	Nessuna.
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "visualizza profilo".</li> <li>2 Il Globetrotter/Cicerone inserisce l'id dell'account.</li> <li>3 Il sistema ricerca l'account.</li> <li>4 Se il sistema trova l'account           <ol style="list-style-type: none"> <li>4.1 Il sistema mostra le informazioni pubbliche dell'account.</li> </ol> </li> <li>5 Altrimenti           <ol style="list-style-type: none"> <li>5.1 Il sistema comunica al Globetrotter/Cicerone che non esistono account con quell'id.</li> </ol> </li> </ol>
<b>Post-condizioni</b>	Nessuna.
<b>Sequenza alternativi degli eventi</b>	Nessuna.

Nome	Modifica profilo
<b>Id</b>	4
<b>Breve descrizione</b>	Il Globetrotter/Cicerone modifica le informazioni relative al suo account
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter/Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "modifica profilo".</li> <li>2 Include(Visualizza profilo).</li> <li>3 Include(Autentica account).</li> </ol> <p>4 Il sistema chiede al</p>

	Globetrotter/Cicerone di inserire le sue informazioni personali tra cui nuova password, conferma nuova password, città di residenza, numero di telefono e lingua.
	5 Il sistema valida le nuove informazioni del Globetrotter/Cicerone. 6 Il sistema modifica le informazioni dell'account del Globetrotter/Cicerone.
<b>Post-condizioni</b>	Il Globetrotter/Cicerone ha modificato le informazioni riguardanti il suo account.
<b>Sequenza alternativi degli eventi</b>	PasswordNonValida NumeroDiTelefonoNonValido NomeNonValido CognomeNonValido

Nome	Visualizza home
<b>Id</b>	5
<b>Breve descrizione</b>	Il Globetrotter/Cicerone visualizza la pagina di presentazione della piattaforma
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	nessuna
<b>Sequenza principale degli eventi</b>	1 Il caso d'uso inizia quando il Globetrotter/Cicerone si connette al sito. 2 Il sistema mostra la vista della home
<b>Post-condizioni</b>	Il Globetrotter/Cicerone visualizza la pagina di presentazione della

---

	piattaforma
Sequenza alternativi degli eventi	nessuna

Nome	Autentica Account:EmailNonValida
<b>Id</b>	1.1
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un email non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un email non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito un email non valida.

Nome	Autentica Account:PasswordNonValida
<b>Id</b>	1.2
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito una password non valida.
<b>Post-condizioni</b>	nessuna

---

<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida.
--	---

<b>Nome</b>	<b>Registra Account:EmailNonValida</b>
<b>Id</b>	2.1
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un email non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un email non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito un email non valida.

<b>Nome</b>	<b>Registra Account:PasswordNonValida</b>
<b>Id</b>	2.2
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito

	una password non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida.

<b>Nome</b>	Registra Account:CittaNonValida
<b>Id</b>	2.3
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito una città di residenza non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito una città di residenza non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito una città di residenza non valida.

<b>Nome</b>	Registra Account: NomeNonValido
<b>Id</b>	2.4
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un nome non valido
<b>Attori primari</b>	Globetrotter, Cicerone

<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un nome non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito un nome non valido.

<b>Nome</b>	<b>Registra Account:</b> <b>CognomeNonValido</b>
<b>Id</b>	2.5
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un cognome non valido
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un cognome non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito un cognome non valido.

<b>Nome</b>	<b>Registra Account:</b> <b>NumeroDiTelefonoNonValido</b>
<b>Id</b>	2.6

<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un numero di telefono non valido
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un numero di telefono non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito un numero di telefono non valido.

<b>Nome</b>	<b>Modifica profilo:PasswordNonValida</b>
<b>Id</b>	4.1
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito una password non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 5 della sequenza principale degli eventi. 2 Il sistema informa il Globetrotter/Cicerone che ha inserito una password non valida.

<b>Nome</b>	<b>Modifica profilo:NumeroDiTelefonoNonValido</b>
<b>Id</b>	4.2
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un numero di telefono non valido
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un numero di telefono non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	3 La sequenza alternativa degli eventi inizia dopo il passo 5 della sequenza principale degli eventi. 4 Il sistema informa il Globetrotter/Cicerone che ha inserito un numero di telefono non valido.

<b>Nome</b>	<b>Modifica Profilo: NomeNonValido</b>
<b>Id</b>	4.3
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un nome non valido
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Globetrotter/Cicerone ha inserito un nome non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 5 della sequenza principale degli eventi. 2 Il sistema informa il

---

Globetrotter/Cicerone che ha inserito un nome non valido.

Nome	Modifica Profilo:CognomeNonValido
<b>Id</b>	4.4
<b>Breve descrizione</b>	Il sistema informa il Globetrotter/Cicerone che ha inserito un cognome non valido
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	2 Il Globetrotter/Cicerone ha inserito un cognome non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	3 La sequenza alternativa degli eventi inizia dopo il passo 5 della sequenza principale degli eventi. 4 Il sistema informa il Globetrotter/Cicerone che ha inserito un cognome non valido.

Nome	Cambia Ruolo
<b>Id</b>	6
<b>Breve descrizione</b>	Il Globetrotter/Cicerone cambia il ruolo associato al suo account accedendo alle rispettive funzionalità
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter/Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "passa a"

	2 Include(Autenticazione account)
	3 Il sistema modifica il ruolo associato all'account
	4 Il sistema indirizza il Globetrotter/Cicerone nella vista home page del ruolo scelto
<b>Post-condizioni</b>	Il Globetrotter/Cicerone ha cambiato il suo ruolo associato all'account
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Effettua logout
<b>Id</b>	7
<b>Breve descrizione</b>	Il Globetrotter/Cicerone
<b>Attori primari</b>	Globetrotter, Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter/Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "logout" 2 Include(Autenticazione account)  3 Il sistema invalida la sessione associata all'account del Globetrotter/Cicerone  4 Il sistema indirizza il Globetrotter/Cicerone alla vista di autenticazione.
<b>Post-condizioni</b>	Il Globetrotter/Cicerone ha effettuato il logout dal proprio account sulla piattaforma

---

<b>Sequenza alternativi degli eventi</b>	nessuna
--	---------

Nome	Visualizza attività create
<b>Id</b>	8
<b>Breve descrizione</b>	Il Cicerone visualizza la lista delle attività create
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "gestione attività"</li> <li>2 Include(Autenticazione account)</li> <li>3 Il sistema cerca le attività create dal Cicerone e associate al suo account id   <i>punto di estensione: cerca attivita</i>  <i>punto di estensione: ordina attivita</i></li> <li>4 Per ogni attività trovata:                     <ol style="list-style-type: none"> <li>4.1 Il sistema mostra l'itinerario dell'attività e le informazioni di base (Titolo, descrizione)</li> <li>4.2 Il sistema mostra lo stato</li> <li>4.3 Se lo stato dell'attività non è "chiuso"                             <ol style="list-style-type: none"> <li>4.3.1 Il sistema mostra l'opzione di</li> </ol> </li> </ol> </li> </ol>

---

	modifica 4.3.2 Il sistema mostra l'opzione di cancellazione 4.3.3 Il sistema mostra l'opzione di gestione dei partecipanti
<b>Post-condizioni</b>	Il Cicerone ha visualizzato lo storico delle sue attività create.
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Crea attività
<b>Id</b>	9
<b>Breve descrizione</b>	Il Cicerone crea una nuova attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Cicerone seleziona la voce "crea attività"</li> <li>2 Include(Visualizza attività create)</li>   <li>3 Il sistema chiede al Cicerone di inserire le informazioni dell'attività tra cui il titolo, le lingue parlate, la retribuzione, il luogo d'incontro, la data e l'ora</li> <li>4 Il sistema valida le informazioni dell'attività.</li> <li>5 Il sistema crea la nuova attività</li> </ol>
<b>Post-condizioni</b>	Il Cicerone ha creato la nuova attività
<b>Sequenza alternativi degli eventi</b>	TitoloNonValido RetribuzioneNonValida LuogoNonValido

DataNonValida
---------------

Nome	Crea attività:TitoloNonValido
<b>Id</b>	9.1
<b>Breve descrizione</b>	Il sistema informa il Cicerone che ha inserito un titolo non valido
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	1 Il Cicerone ha inserito un titolo non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Cicerone che ha inserito un titolo non valido.

Nome	Crea attività:RetribuzioneNonValida
<b>Id</b>	9.2
<b>Breve descrizione</b>	Il sistema informa il Cicerone che ha inserito una retribuzione non valida
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	2 Il Cicerone ha inserito una retribuzione non valida.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi.

- 
- 2 Il sistema informa il Cicerone che ha inserito una retribuzione non valida.

Nome	Crea attività:LuogoNonValido
<b>Id</b>	9.3
<b>Breve descrizione</b>	Il sistema informa il Cicerone che ha inserito un luogo non valido
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone ha inserito un luogo non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi. 2 Il sistema informa il Cicerone che ha inserito un luogo non valido.

Nome	Crea attività:DataNonValida
<b>Id</b>	9.4
<b>Breve descrizione</b>	Il sistema informa il Cicerone che ha inserito una data non valida
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone ha inserito una data non valida.
<b>Post-condizioni</b>	nessuna

---

<b>Sequenza alternativi degli eventi</b>	<ol style="list-style-type: none"> <li>1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi.</li> <li>2 Il sistema informa il Cicerone che ha inserito una data non valida.</li> </ol>
--	--

Nome	Modifica attività
<b>Id</b>	10
<b>Breve descrizione</b>	Il Cicerone modifica un'attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Cicerone seleziona la voce "modifica attività"</li> <li>2 Include(Visualizza attività create)</li> <li>3 Il sistema chiede al Cicerone di inserire le informazioni da modificare dell'attività tra il luogo d'incontro e l'ora</li> <li>4 Il sistema valida le informazioni dell'attività.</li> <li>5 Il sistema modifica l'attività</li> </ol>
<b>Post-condizioni</b>	Il Cicerone ha modificato l'attività
<b>Sequenza alternativi degli eventi</b>	LuogoNonValido

Nome	Modifica attività: <u>LuogoNonValido</u>
<b>Id</b>	10.1
<b>Breve descrizione</b>	Il sistema informa il Cicerone che ha

	inserito un luogo non valido
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone ha inserito un luogo non valido.
<b>Post-condizioni</b>	nessuna
<b>Sequenza alternativi degli eventi</b>	<ol style="list-style-type: none"> <li>1 La sequenza alternativa degli eventi inizia dopo il passo 3 della sequenza principale degli eventi.</li> <li>2 Il sistema informa il Cicerone che ha inserito un luogo non valido.</li> </ol>

Nome	Elimina attività
<b>Id</b>	11
<b>Breve descrizione</b>	Il Cicerone modifica un'attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Cicerone seleziona la voce "elimina attività"</li> <li>2 Include(Visualizza attività create)</li> <li>3 Il sistema elimina l'attività selezionata dal Cicerone</li> </ol>
<b>Post-condizioni</b>	Il Cicerone ha eliminato l'attività
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Accetta richiesta
<b>Id</b>	12

<b>Breve descrizione</b>	Il Cicerone accetta una richiesta d'iscrizione alle sue attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Cicerone seleziona la voce "accetta"</li> <li>2 Include (Gestisci iscritti)</li> <li>3 Il sistema modifica lo stato della richiesta in "accettata"</li> <li>4 Il sistema aggiunge l'account richiedente alla lista dei partecipanti all'attività modificando l'attività</li> </ol>
<b>Post-condizioni</b>	Il Cicerone ha accettato la richiesta
<b>Sequenza alternativi degli eventi</b>	nessuna

Rifiuta richiesta	
<b>Id</b>	13
<b>Breve descrizione</b>	Il Cicerone rifiuta una richiesta d'iscrizione alle sue attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Cicerone seleziona la voce "accetta"</li> <li>2 Include (Gestisci iscritti)</li> <li>3 Il sistema modifica lo stato della richiesta in "rifiutata"</li> </ol>

<b>Post-condizioni</b>	Il Cicerone ha rifiutato la richiesta
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Visualizza catalogo attività
<b>Id</b>	14
<b>Breve descrizione</b>	Il Globetrotter visualizza la lista delle attività disponibili
<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter seleziona la voce "trova attività"</li> <li>2 Include(Autenticazione account)</li> <li>3 Il sistema cerca le attività disponibili ad iscrizione per data</li> </ol> <p style="text-align: center;"><i>punto di estensione: cerca attivita punto di estensione: ordina attivita</i></p> <ol style="list-style-type: none"> <li>4 Per ogni attività trovata             <ol style="list-style-type: none"> <li>4.1 Il sistema mostra l'itinerario dell'attività e le informazioni di base (Titolo, descrizione)</li> <li>4.2 Il sistema mostra l'opzione "iscriviti"</li> </ol> </li> </ol>

---

<b>Post-condizioni</b>	Il Globetrotter ha visualizzato le attività.
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Crea Richiesta
<b>Id</b>	15
<b>Breve descrizione</b>	Il Globetrotter crea una richiesta d'iscrizione ad un'attività
<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter seleziona la voce "iscriviti"</li> <li>2 Include(Visualizza catalogo attività)</li> <li>3 Il sistema crea una nuova richiesta d'iscrizione con stato "In sospeso", con l'account del Globetrotter come richiedente e con l'attività a cui si riferisce l'iscrizione</li> </ol>
<b>Post-condizioni</b>	Il Globetrotter ha creato una richiesta d'iscrizione ad un'attività
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Visualizza richieste create
<b>Id</b>	16
<b>Breve descrizione</b>	Il Globetrotter visualizza la lista delle richieste d'iscrizione create

<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Globetrotter è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<p>1 Il caso d'uso inizia quando il Globetrotter seleziona la voce "storico richieste"</p> <p>2 Include(Autenticazione account)</p> <p>3 Il sistema cerca le richieste associate all'account del Globetrotter</p> <p><i>punto di estensione: ordina richieste</i>  <i>punto di estensione: Invia feedback</i></p> <p>4 Per ogni richiesta trovata</p> <p>4.1 Il sistema mostra la data di creazione della richiesta</p> <p>4.2 Il sistema mostra le informazioni parziali dell'attività a cui si riferisce la richiesta</p> <p>4.3 Il sistema mostra lo stato della richiesta</p> <p>4.4 Se lo stato è uguale ad "accettata"</p> <p>4.5 Visualizza l'email e il numero di telefono del Cicerone creatore della attività</p>
<b>Post-condizioni</b>	Il Globetrotter ha visualizzato le attività.
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Gestisci iscritti
Id	17

<b>Breve descrizione</b>	Il Cicerone visualizza la lista dei partecipanti di un'attività
<b>Attori primari</b>	Cicerone
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il Cicerone è autenticato sulla piattaforma
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "gestione partecipanti"</li> <li>2 Include(Autenticazione account)</li> <li>3 Il sistema cerca gli account partecipanti ad un'attività utilizzando l'id dell'attività</li> <li>4 Per ogni account partecipante trovato                     <ol style="list-style-type: none"> <li>4.1 Il sistema mostra il nome, il cognome, l'email e il numero di telefono dell'account</li> <li>5 Il sistema carica le richieste associate all'attività utilizzando l'id dell'attività stessa</li> <li>6 Per ogni richiesta in sospeso                             <ol style="list-style-type: none"> <li>6.1 Il sistema mostra il nome, il cognome e l'email dell'account mittente</li> <li>6.2 Il sistema mostra l'opzione "accetta"</li> <li>6.3 Il sistema mostra l'opzione "rifiuta"</li> </ol> </li> </ol> </li> </ol>
<b>Post-condizioni</b>	Il Cicerone ha visualizzato la lista dei partecipanti per un'attività.
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Visualizza Dettagli
<b>Id</b>	18
<b>Breve descrizione</b>	Il Cicerone/Globetrotter visualizza i dettagli completi dell'attività
<b>Attori primari</b>	Cicerone/Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	nessuna
<b>Sequenza principale degli eventi</b>	<p>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "dettagli"</p> <p>2 Il sistema cerca l'attività utilizzando l'id dell'attività</p> <p>3 Il sistema visualizza tutti i campi dell'attività</p> <p><i>punto di estensione: visualizza feedback</i></p>
<b>Post-condizioni</b>	Il Cicerone/Globetrotter ha visualizzato la lista dei partecipanti per un'attività.
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Ordina attivita
<b>Id</b>	19
<b>Breve descrizione</b>	Il Cicerone/Globetrotter ordina le attività
<b>Attori primari</b>	Cicerone/Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il sistema ha cercato le attivita per id
<b>Sequenza principale degli eventi</b>	<p>1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "ordina" scegliendo l'attributo di ordinamento</p>

	2 Il sistema ordina le attivita in base all'attributo scelto 3 Il sistema visualizza le attivita ordinate
<b>Post-condizioni</b>	Il Cicerone/Globetrotter visualizza le attivita ordinate
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Cerca attivita
<b>Id</b>	20
<b>Breve descrizione</b>	Il Cicerone/Globetrotter cerca le attivita
<b>Attori primari</b>	Cicerone/Globetrotter
<b>Attori secondari</b>	nessuno
<b>Precondizioni</b>	Il sistema ha cercato le attivita per id
<b>Sequenza principale degli eventi</b>	1 Il caso d'uso inizia quando il Globetrotter/Cicerone seleziona la voce "filtra" scegliendo il filtro di interesse 2 Il sistema mostra le attivita filtrate in base al filtro scelto
<b>Post-condizioni</b>	Il Cicerone/Globetrotter visualizza le attivita filtrate
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Ordina richieste
<b>Id</b>	21
<b>Breve descrizione</b>	Il Globetrotter ordina le richieste che ha effettuato
<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	nessuno

<b>Precondizioni</b>	Il Globetrotter visualizza le richieste create
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il caso d'uso inizia quando il Globetrotter seleziona la voce "ordina richieste"</li> <li>2 Il sistema mostra le richieste ordinate</li> </ol>
<b>Post-condizioni</b>	Il Globetrotter visualizza le richieste ordinate
<b>Sequenza alternativi degli eventi</b>	nessuna

<b>Nome</b>	Chiudi attività
<b>Id</b>	22
<b>Breve descrizione</b>	Il sistema chiude le attività che sono scadute
<b>Attori primari</b>	Sistema
<b>Attori secondari</b>	Tempo
<b>Precondizioni</b>	Sono scattate le 00:00 della giornata odierna
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il sistema cerca le attività con data di incontro uguale alla data odierna</li> <li>2 Per ogni attività trovata                     <ol style="list-style-type: none"> <li>2.1 Sarà modificato lo stato dell'attività in chiuso</li> </ol> </li> </ol>
<b>Post-condizioni</b>	Le attività vengono concluse
<b>Sequenza alternativi degli eventi</b>	nessuna

<b>Nome</b>	Invia feedback
<b>Id</b>	23
<b>Breve descrizione</b>	Il Globetrotter può inserire una valutazione ad un'attività a cui ha

	partecipato
<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	Nessuno
<b>Precondizioni</b>	<ol style="list-style-type: none"> <li>1 Il Globetrotter deve essere autenticato</li> <li>2 Il Globetrotter è nella lista dei partecipanti all'attività,</li> <li>3 L'attività è nello stato "chiuso"</li> </ol>
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il Globetrotter inserisce i dati del feedback quali commento e valutazione</li> <li>2 Il sistema crea il feedback e lo salva</li> </ol>
<b>Post-condizioni</b>	<ol style="list-style-type: none"> <li>1 Il Globetrotter ha rilasciato un feedback per l'attività correlata</li> <li>2 Il sistema salva il feedback per quella attività</li> </ol>
<b>Sequenza alternativi degli eventi</b>	nessuna

Nome	Visualizza feedback
<b>Id</b>	24
<b>Breve descrizione</b>	Il Globetrotter/Cicerone i feedback rilasciati per un'attività
<b>Attori primari</b>	Globetrotter
<b>Attori secondari</b>	Nessuno
<b>Precondizioni</b>	<ol style="list-style-type: none"> <li>1 Il Globetrotter/Cicerone è nella schermata di visualizzazione dei dettagli di un'attività</li> </ol>
<b>Sequenza principale degli eventi</b>	<ol style="list-style-type: none"> <li>1 Il sistema cerca i feedback per l'attività</li> <li>2 Per ogni feedback trovato</li> </ol>

	2.1 visualizza il nome e il cognome del creatore del feedback 2.2 visualizza il commento 2.3 visualizza la valutazione numerica
<b>Post-condizioni</b>	1 Il Globetrotter/Cicerone ha visualizzato i feedback dell'attività
<b>Sequenza alternativi degli eventi</b>	nessuna

### 2.2.3 Interfaccia

#### Presentazione piattaforma (Home) (Sprint 1)



## Presentazione piattaforma (Home: Cos'è?) (Sprint 1)

The screenshot shows the Cicerone platform's home page. At the top, there is a teal header bar with the Cicerone logo, a navigation menu with links to 'Cos'è?', 'Come funziona?', 'Assistenza', and 'Accedi', and a search bar. Below the header, the main content area has a dark background. The title 'Cos'è Cicerone?' is displayed in white. A descriptive paragraph follows, explaining that Cicerone is a platform born from the real need to travel and learn about the world we live in, offering unique travel experiences and intercultural encounters through direct immersion in local culture and customs. An illustration below the text shows a stack of travel-related icons (airplane, suitcase, map) above a small figure of a person, with mathematical operators (+, =) indicating a combination or result.

## Presentazione piattaforma (Home: Come funziona?) (Sprint 1)

The screenshot shows the Cicerone platform's home page. At the top, there is a teal header bar with the Cicerone logo, a navigation menu with links to 'Cos'è?', 'Come funziona?', 'Assistenza', and 'Accedi', and a search bar. Below the header, the main content area has a dark background. The title 'Come funziona?' is displayed in white. A descriptive paragraph explains that everyone is a Globetrotter because it is innate in us to want to discover and know the world around us, and it asks if there is a better way to do this than by accompanying our figure with that of a Cicerone? These two figures represent the center of the platform. Below the text, there are two circular icons: 'Globetrotter' (a cartoon character with a globe and a backpack) and 'Cicerone' (a cartoon character with a lightbulb above their head). Each icon has a corresponding text block and a 'Mostra descrizione' button.

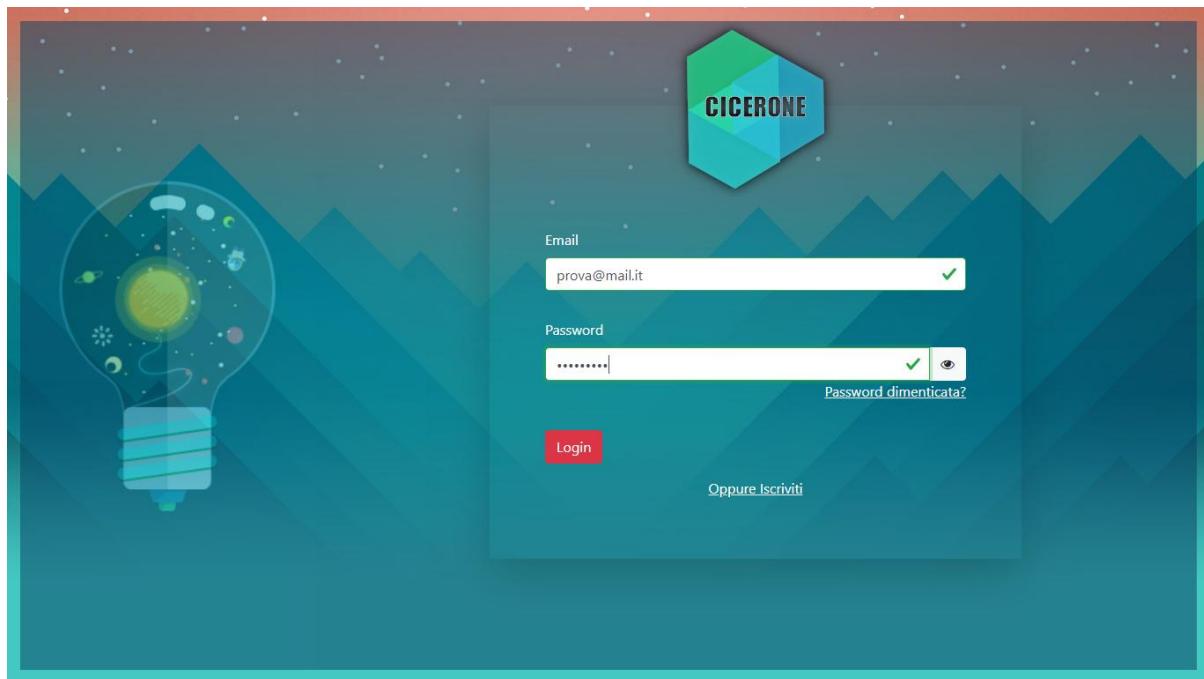
## Presentazione piattaforma (Home: Area assistenza) (Sprint 1)

The screenshot shows the Cicerone support home page. At the top, there is a navigation bar with links for "Cos'è?", "Come funziona?", "Assistenza", and "Accedi". Below the navigation is a stylized illustration of a team working together. A message below the illustration reads: "Il nostro team di sviluppo è sempre pronto a rispondere. Compila il modulo qui di seguito e sarai ricontattato al più presto." The main area contains a form with four fields: "Email Address\*", "Nome\*", "Cognome\*", and "Inserisci la tua domanda qui\*". Each field has a red error message below it: "Inserisci un email valida.", "Inserisci un nome valido.", "Inserisci un cognome valido.", and "Inserisci una descrizione valida.". A red "Invia richiesta" button is at the bottom.

## Registrazione Account (Sprint 1)

The screenshot shows the Cicerone account registration page. The background features a large, glowing lightbulb icon on the left and a green hexagonal logo with the word "CICERONE" in the center. The form fields include: "Nome" (Mario, checked), "Cognome" (Rossi, checked), "Email" (prova@mail.it, checked), "Numero di telefono" (3330000000, checked), "Città" (Pisa, Pisa, Italia, checked), "Password" (\*\*\*\*\*, checked), "Conferma password" (\*\*\*\*\*, checked), "Sesso:" (radio buttons for Uomo and Donna), "Data di nascita:" (23/12/2018, checked), and a checkbox for "Accetta i termini e il contratto della piattaforma" (checked). A red "Iscriviti" button is at the bottom.

## Login Account (Sprint 1)



## Menu di scelta funzionalità account post login (Sprint 1)



## Modifica profilo – Pannello di controllo Cicerone e Globetrotter (Sprint 1)

The screenshot shows the 'Modifica il tuo profilo:' (Edit your profile) section of the Cicerone and Globetrotter application. On the left, a sidebar menu includes 'Home', 'Gestisci Attività', 'Notifiche', 'Impostazioni', 'Modifica profilo' (selected), 'a Globetrotter', and 'Esci'. The main area displays a profile picture of a cartoon character, the last used role 'globetrotter', and basic profile information: Nome: Mario, Cognome: Rossi, Residenza: Pisa, Pisa, Italia. Below this is a section for 'Informazioni di base' (Basic information) with fields for Email (prova@mail.it), Nuova password (Password field with eye icon), Numero di telefono (3330000000), Genere (DONNA), and Data di nascita (23/12/2018).

## Crea attività (Sprint 2)

The screenshot shows the 'Creazione attività' (Create activity) section of the Cicerone and Globetrotter application. The sidebar menu is identical to the previous screenshot. The main area allows creating a new activity with fields for Titolo Attività (Itinerario), Lingua (Inserisci qui la lingua... with 'Aggiungi lingua' button, showing '#English #Italian'), and Retribuzione (Non Retribuito). Below these fields is a map editor interface with a road network, directional arrows, and zoom controls (+, -, circle).

## Modifica attività (Sprint 2)

Cicerone

Benvenuto  
Mario Rossi

Modifica attività

**Titolo Attività:**  
Itinerario

**Lingua:**  
Italiano

**Retribuzione:**  
NONRETRIBUITO  
0.0

Strada

Palese S556

Rivoli

1 miglio 1 km

## Modifica profilo (Sprint 2)

Cicerone

Benvenuto  
Mario

Ricerca

Modifica il tuo profilo:

Ultimo ruolo utilizzato: cicerone

Nome:  
Mario

Cognome:  
Rossi

Residenza:  
Pisa, Pisa, Italia

Informazioni di base

Email: prova@mail.it

Nuova password:

Numero di telefono: 3330000000

Genere: UOMO

Data di nascita: 23/12/2018

## Gestisci iscritti (Sprint 2)

The screenshot shows the Cicerone application interface. At the top, there is a green header bar with the Cicerone logo and user information: "Benvenuto Mario Rossi". On the right side of the header are icons for profile, messages, settings, and search. Below the header is a sidebar on the left containing navigation links: Home, Attività create (selected), Nuova Attività, Modifica profilo, a Globetrotter, and Esci. The main content area has a title "Gestione partecipanti per l'attività:" followed by a "Partecipanti:" section. Under "Richieste d'iscrizione:", it shows a participant named "Partecipante: Gianfranco Rugani". At the bottom of this section are two buttons: "Accetta" (Accept) in green and "Rifiuta" (Reject) in red.

## Visualizza attività create (Sprint 2)

The screenshot shows the Cicerone application interface. At the top, there is a green header bar with the Cicerone logo and user information: "Benvenuto Mario Rossi". On the right side of the header are icons for profile, messages, settings, and search. Below the header is a sidebar on the left containing navigation links: Home, Attività create (selected), Nuova Attività, Modifica profilo, a Globetrotter, and Esci. The main content area displays a summary of one activity found: "1 Attività trovate". It includes fields for "Creatore: Mario Rossi", "Data apertura: 2019-07-12", "Luogo d'incontro: Pisano Maria Grazia, Via Benedetto Cairoli Carbonara 71, Bari, BA, Italia", "Data d'incontro: 2019-07-23", "Ora d'incontro: 15:23", and an "Itinerario" section with a placeholder "Descrizione opzionale.". At the bottom of the activity card are four buttons: "Cancella" (Delete) in red, "Partecipanti" (Participants) in blue, "Modifica" (Edit) in blue, and "Dettagli" (Details) in blue. A small green circular button with a plus sign is located in the bottom right corner of the main content area.



## Visualizza catalogo attività (Sprint 2)

The screenshot shows the Cicerone application interface. On the left is a dark sidebar with user information (Luigi Verdi) and navigation links: Home, Trova Attività, Richieste, Modifica profilo, a Cicerone, and Esci. The main area has a teal header with the Cicerone logo and a search bar. Below the header are buttons for List and Grid view, and filters. A message indicates 1 Attività trovate. The results show a single activity entry for Mario Rossi, dated 2019-07-12, with details: Luogo d'incontro: Pisano Maria Grazia, Via Benedetto Cairoli Carbonara 71, Bari, BA, Italia; Data d'incontro: 2019-07-23; Ora d'incontro: 15:23. There is an optional itinerary description field and two buttons at the bottom: Iscriviti and Dettagli.

## Visualizza dettagli attività (Sprint 2)

The screenshot shows the Cicerone application interface. The title is "Visualizza attività". It displays basic metadata: Data creazione: 2019-07-12 and Ultima modifica: 2019-07-12. Below this, detailed activity information is shown in a table-like format: Titolo Attività: Itinerario; Lingua: #English #Italian; Retribuzione: NONRETRIBUITO, 0.0. The right side of the screen contains a large, empty white box.



## Visualizza richieste create (Sprint 2)

The screenshot shows a user profile for 'Luigi Verdi' on the left sidebar. The main content area displays a request with ID 51f3fa0534b2402f85dc87542fa35da8, created on 2019-07-12. The request is for an itinerary created by 'Mario Rossi'. The status is listed as 'in sospeso'.

Richiesta id: 51f3fa0534b2402f85dc87542fa35da8      Data di creazione: 2019-07-12

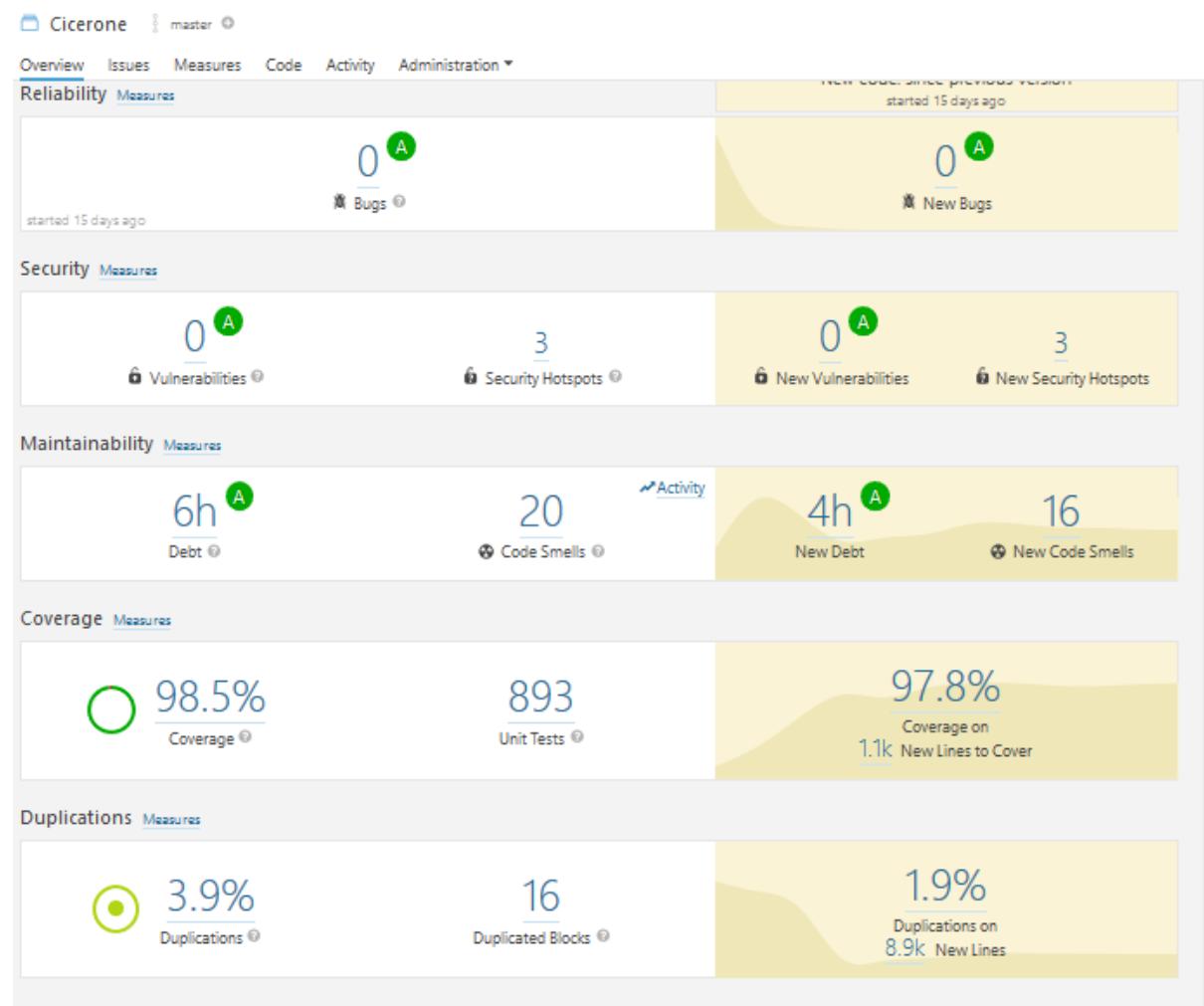
**Itinerario**

Hai chiesto di unirti alla attività:Itinerario  
creata da:Mario Rossi

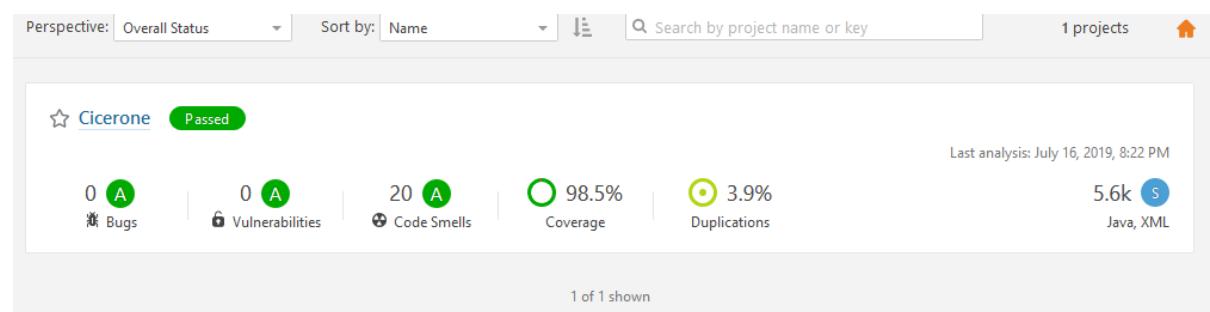
stato: in sospeso

## 2.2.4 Qualità

Qualità generale della terza release di Cicerone.

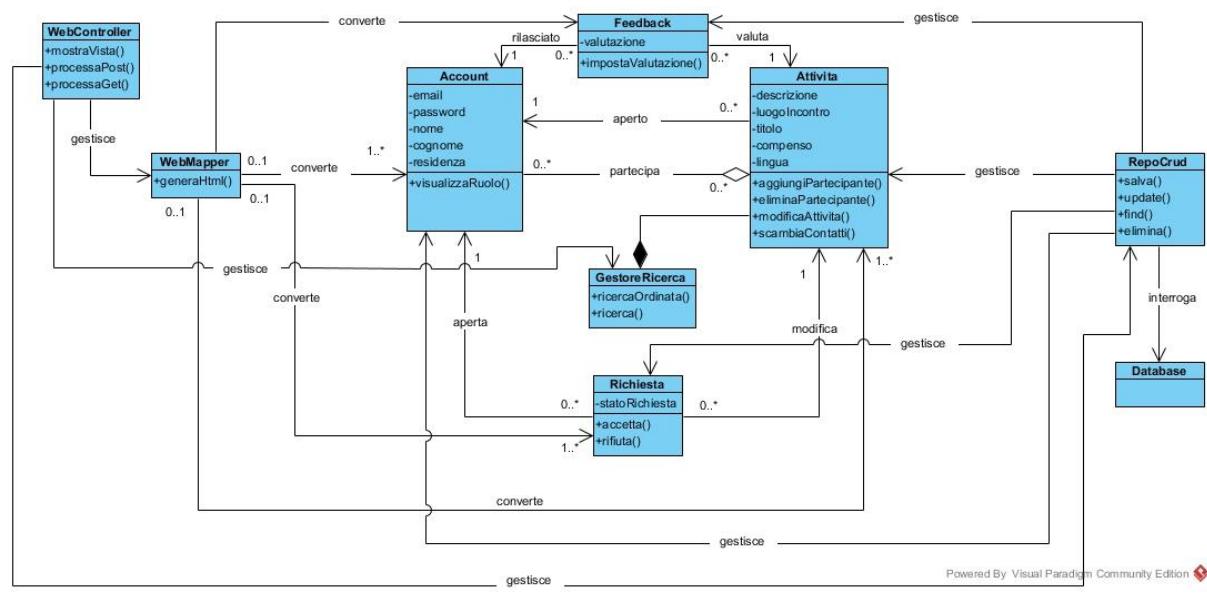
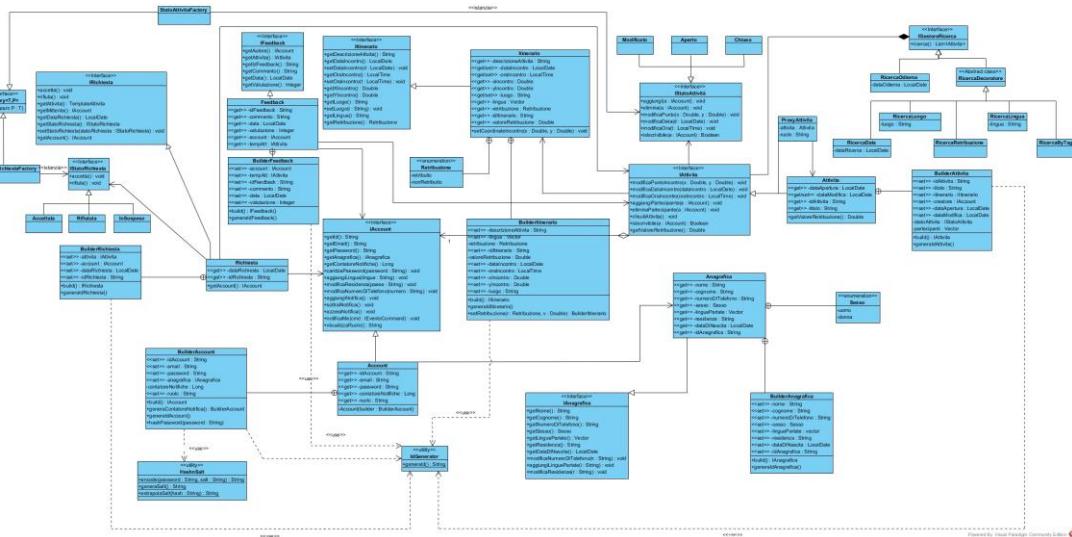


Qualità report della terza release di Cicerone.



### 3. PRODUCT DESIGN

#### 3.1 Diagramma delle Classi



## 3.2 Tabella di tracciabilità Componenti-Classi

	SpringBoot	Storage	WebController	Autenticazione
CrudRepository	X			
TaskChiusuraAutomatica	X			
AccountRepo		X		
AccountCRUD		X		
AccountCrudImpl		X		
AccountEntity		X		
AnagraficaRepo		X		
AnagraficaEntity		X		
LinguaAccountRepo		X		
LinguaAccountEntity		X		
PartecipantiRepo		X		
PartecipantiEntity		X		
AttivitaCRUD		X		
AttivitaRepo		X		
AttivitaEntity		X		
ItinerarioRepo		X		
ItinerarioEntity		X		
LinguaAttivitaRepo		X		
LinguaAttivitaEntity		X		
RichiestaCRUD		X		
RichiestaRepo		X		
RichiestaEntity		X		
FeedbackCRUD		X		
FeedbackRepo		X		
FeedbackEntity		X		
IMapper<T>				X
AccountModelMapper				X
AttivitaModelMapper				X
RichiestaModelMapper				X
WebsiteIndice				X
WebConstant				X
WebControllerSignup				X
WebControllerVisualizzaProfilo				X
WebControllerLogin				X
WebControllerHome				X
WebControllerModificaProfilo				X
WebControllerCiceroneAttivita				X
WebControllerEliminaAttivita				X

WebControllerModificaAttivita			X	
WebControllerGestisciPartecipanti			X	
WebControllerGlobetrotterAttivita			X	
WebControllerIscriviti			X	
WebControllerGestioneRichiesta			X	
WebControllerCiceroneRichiesta			X	
WebControllerCreaAttivita			X	
WebControllerVisualizzaDettagli			X	
WebControllerValidaRichiesta			X	
IAccount				X
Account				X
BuilderAccount				X
IAnagrafica				X
BuilderAnagrafica				X
Anagrafica				X
Sesso				X
HashnSalt				X
IdGenerator				X
	GestioneAttivita	RicercaOrdinamento	Viste	GestoreFeedBack
ErrorMessage	X			
Factory<T,P>	X			
StatoAttivitaFactory	X			
StatoRichiestaFactory	X			
IItinerario	X			
Itinerario	X			
BuilderItinerario	X			
Retribuzione	X			
IStatoRichiesta	X			
Accettata	X			
Rifiutata	X			
InSospeso	X			
Richiesta	X			
IRichiesta	X			
BuilderRichiesta	X			
IAttivita	X			
IStatoAttivita	X			
BuilderAttivita	X			
ProxyAttivita	X			
Modificato	X			
Aperto	X			
Chiuso	X			
Attivita	X			

IGestoreRicerca		X		
RicercaOdierna		X		
RicercaDecoratore		X		
RicercaLuogo		X		
RicercaData		X		
RicercaRetribuzione		X		
RicercaLingua		X		
VisualizzaProfilo.jsp				X
Login.jsp				X
Choose.jsp				X
Signup.jsp				X
ModificaProfilo.jsp				X
CiceroneAttivita.jsp				X
ModificaAttivita.jsp				X
GestisciPartecipanti.jsp				X
GlobetrotterAttivita.jsp				X
GestioneRichieste.jsp				X
CreaAttivita.jsp				X
Feedback				X
BuilderFeedback				X
IFeedback				X

### 3.3 Specifiche degli enumerativi

- **Sesso:** Enumerativo che rappresenta il genere di una persona e che può assumere valore di uomo o valore di donna.
- **Retribuzione:** enumerativo che rappresenta la quantità di denaro richiesta eventualmente dal Cicerone e che può assumere valore di retribuito e di non retribuito.

### 3.4 Specifiche delle classi

- **Richiesta:** classe che rappresenta e gestisce una richiesta di iscrizione da parte di un account ad un'attività.
  - Attributo/i:

- 
- `private String idRichiesta`: attributo che permette l'identificazione univoca della richiesta. L'utilizzo del tipo `String` è dovuto all'unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;
  - `private IStatoRichiesta statoRichiesta`: attributo che rappresenta la relazione di stato con `IStatoRichiesta` definita tramite l'utilizzo dello *State Pattern Design*;
  - `private IAttivita attivita`: attributo che rappresenta la relazione con l'attività a cui si riferisce la richiesta;
  - `private IAccount account`: attributo che rappresenta la relazione con l'account dell'utente che ha richiesto l'iscrizione;
  - `private LocalDate dataRichiesta`: attributo che rappresenta la data in cui l'utente ha effettuato la richiesta (nel formato mm/gg/yyyy).

- Metodo/i:

- `public String getIdRichiesta()`;
- `public void accetta()`: operazione che permette l'accettazione dell'utente all'attività con cambio di stato in accettato, se lo stato in cui versa lo permette. In caso contrario verrà sollevata un'eccezione del tipo `UnsupportedOperationException`;
- `public void rifiuta()`: operazione che permette il rifiuto dell'utente all'attività con cambio di stato in rifiutato, se lo stato in cui versa lo permette. In caso contrario verrà

---

sollevata un'eccezione del tipo  
UnsupportedOperationException;

- public IAttivita getAttivita();
- public IAccount getMittente();
- public LocalDate getDataRichiesta();
- public IStatoRichiesta getStatoRichiesta();
- public void setStatoRichiesta(IStatoRichiesta statoRichiesta);

- **BuilderRichiesta:** classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Richiesta* rendendo immutabili determinati attributi.
  - Attributo/i:
    - private String idRichiesta;
    - private IStatoRichiesta statoRichiesta;
    - private IAttivita attivita;
    - private IAccount account;
    - private LocalDate dataRichiesta.
  - Metodo/i:
    - public BuilderRichiesta setIdRichiesta(String idRichiesta);
    - public BuilderRichiesta setAttivita(IAttivita attivita);
    - public BuilderRichiesta setAccount(IAccount account);
    - public BuilderRichiesta setDataRichiesta(LocalDate dataRichiesta);
    - public IRichiesta build();
    - public BuilderRichiesta generaldRichiesta();

- 
- **StatoRichiestaFactory:** classe che implementa il Factory Method Pattern che funziona come interfaccia per l'instanziazione degli stati della richiesta.
  - **Accettata:** classe che implementa l'interfaccia *IStatoRichiesta*, secondo lo *State Pattern Design* e ne rappresenta lo stato di accettazione.
    - Attributo/i:
      - private IRichiesta richiesta.
    - Metodo/i:
      - public void accetta(): operazione che solleva un'eccezione di UnsupportedOperationException poiché è già nello stato di accettazione;
      - public void rifiuta(): operazione che solleva un'eccezione di UnsupportedOperationException poiché è già nello stato di accettazione.
  - **Rifiutata:** classe che implementa l'interfaccia *IStatoRichiesta*, secondo lo *State Pattern Design* e ne rappresenta lo stato di rifiuto.
    - Attributo/i:
      - private IRichiesta richiesta.
    - Metodo/i:
      - public void accetta(): operazione che solleva un'eccezione di UnsupportedOperationException poiché è già nello stato di rifiuto;
      - public void rifiuta(): operazione che solleva un'eccezione di UnsupportedOperationException poiché è già nello stato di rifiuto.



- 
- **InSospeso:** classe che implementa l’interfaccia *IStatoRichiesta* e, secondo lo *State Pattern Design* e ne rappresenta lo stato di “*pending*”.
    - Attributo/i:
      - private *IRichiesta* richiesta;
    - Metodo/i:
      - public void accetta(): operazione che permette l’accettazione dell’utente all’attività, la sua aggiunta alla lista dei partecipanti dell’attività e il cambio di stato in accettato;
      - public void rifiuta(): operazione che permette il rifiuto dell’utente all’attività e il cambio di stato in rifiutato.
  - **Feedback:** classe che rappresenta e gestisce il feedback rilasciato da un utente relativo ad un’attività chiusa.
    - Attributo/i:
      - private *IAccount* account: attributo che rappresenta la relazione con l’account che rilascia il feedback;
      - private *IAttivita* tempAtt: attributo che rappresenta la relazione con l’attività alla quale il feedback è correlato;
      - private String idFeedback: attributo che permette l’identificazione univoca del feedback. L’utilizzo del tipo String è dovuto all’unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;



- 
- `private String commento`: attributo che rappresenta un commento che l'utente potrà inserire nel feedback, non nullo e di lunghezza non maggiore a 500 caratteri;
  - `private LocalDate data`: attributo che rappresenta la data in cui viene creato il feedback;
  - `private Integer valutazione`: attributo che rappresenta il valore della valutazione espresso in forma numerica. Il valore è compreso in un range da 0 a 5 (dove 0 è il minimo e 5 il massimo).
  - Metodo/i:
    - `public String getIdFeedback()`;
    - `public IAccount getAutore()`;
    - `public IAttivita getAttivita()`;
    - `public Integer getValutazione()`;
    - `public String getCommento()`;
    - `public LocalDate getData()`.
  - **BuilderFeedback**: classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Feedback* rendendo immutabili determinati attributi.
    - Attributo/i:
      - `private IAccount account`;
      - `private IAttivita tempAtt`;
      - `private String idFeedback`;
      - `private String commento`;
      - `private LocalDate data`;
      - `private Integer valutazione`.



- 
- Metodo/i:
    - public BuilderFeedback setAttivita(IAttivita attivita);
    - public BuilderFeedback setAccount(IAccount account);
    - public BuilderFeedback setIdFeedback(String idFeedback);
    - public BuilderFeedback setCommento(String commento): operazione che permette di impostare il commento relativo al feedback. Solleva un'eccezione di tipo IllegalArgumentException se la lunghezza del campo non è compresa tra 0 e 500 caratteri.
    - public BuilderFeedback setValutazione(Integer valutazione): operazione che permette di impostare la valutazione del feedback espressa in numero. Solleva un'eccezione di tipo IllegalArgumentException se il valore del campo non è compreso tra 0 e 5;
    - public BuilderFeedback setData(LocalDate data);
    - public IFeedback build();
    - public BuilderFeedback generaldFeedback();
  - **Itinerario:** classe che rappresenta e gestisce le informazioni relative alla caratterizzazione di un'attività.
    - Attributo/i:
      - private String idItinerario: attributo che permette l'identificazione univoca dell'itinerario. L'utilizzo del tipo String è dovuto all'unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;



- 
- `private String descrizioneAttivita:` attributo che rappresenta la descrizione testuale dell'attività;
  - `private LocalDate dataIncontro:` attributo che rappresenta la data dell'incontro prestabilita (in formato mm/gg/yyyy);
  - `private LocalTime oraIncontro:` attributo che rappresenta l'ora dell'incontro prestabilita (in formato hh:mm);
  - `private Double xIncontro:` attributo che rappresenta la latitudine del punto di incontro;
  - `private Double yIncontro:` attributo che rappresenta la longitudine del punto di incontro;
  - `private String luogo:` attributo che rappresenta il nome in forma testuale del luogo di incontro;
  - `private Set<String> lingua:` attributo che rappresenta l'insieme univoco delle lingue utilizzate per lo svolgimento dell'attività;
  - `private Retribuzione retribuzione:` attributo che rappresenta la forma di retribuzione per lo svolgimento dell'attività;
  - `private Double valoreRetribuzione:` attributo che rappresenta la quantità di denaro se la retribuzione è del tipo "retribuita".
- **Metodo/i:**
    - `public String getIdItinerario();`
    - `public Double getvaloreRetribuzione();`
    - `public LocalDate getDataIncontro();`



- 
- `public LocalTime getOraIncontro();`
  - `public void setDataIncontro(LocalDate dataIncontro);`
  - `public Set<String> getLingua();`
  - `public Retribuzione getRetribuzione();`
  - `public void setOraIncontro(LocalTime oraIncontro);`
  - `public String getDescrizioneAttivita();`
  - `public Double getXIncontro();`
  - `public Double getYIncontro();`
  - `public void setCoordinateIncontro(Double x, Double y);`
  - `public String getLuogo();`
  - `public void setLuogo(String luogo).`
  - **BuilderItinerario:** classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Itinerario* rendendo immutabili determinati attributi.
    - Attributo/i:
      - `private String idItinerario;`
      - `private String descrizioneAttivita;`
      - `private LocalDate dataIncontro;`
      - `private LocalTime oraIncontro;`
      - `private Double xIncontro;`
      - `private Double yIncontro;`
      - `private String luogo;`
      - `private Set<String> lingua;`
      - `private Retribuzione retribuzione;`
      - `private Double valoreRetribuzione.`
    - Metodo/i:



- 
- `public BuilderItinerario setIdItinerario(String idItinerario);`
  - `public BuilderItinerario setDescrizioneAttivita(String descrizioneAttivita): operazione che permette di impostare il campo della descrizione dell'attività. Solleva un'eccezione del tipo IllegalArgumentException se la lunghezza del campo non è compreso tra 0 e 500 caratteri.`
  - `public BuilderItinerario setDataIncontro(LocalDate dataIncontro);`
  - `public BuilderItinerario setOraIncontro(LocalTime oraIncontro);`
  - `public BuilderItinerario setXIncontro(Double xIncontro);`
  - `public BuilderItinerario setYIncontro(Double yIncontro);`
  - `public BuilderItinerario setLuogo(String luogo);`
  - `public BuilderItinerario setLingua(Set<String> lingua);`
  - `public BuilderItinerario setRetribuzione(Retribuzione r, Double v): operazione che permette di impostare il valore di retribuzione dell'itinerario. Solleva un'eccezione di tipo IllegalArgumentException se il valore di valoreRetribuzione è negativo;`
  - `public Itinerario build();`
  - `public BuilderItinerario generaldItinerario();`
- **Attivita:** classe che rappresenta e gestisce l'attività e che implementa il *Proxy Pattern Design*.
    - Attributo/i:



- 
- `private LocalDate dataApertura`: attributo che rappresenta la data dell'apertura dell'attività (nel formato mm/gg/yyyy);
  - `private LocalDate dataModifica`: attributo che rappresenta la data di modifica dell'attività (nel formato mm/gg/yyyy);
  - `private String idAttivita`: attributo che permette l'identificazione univoca dell'attività. L'utilizzo del tipo `String` è dovuto all'unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;
  - `private String titolo`: attributo che rappresenta il titolo da inserire all'attività creata;
  - `private IItinerario itinerario`: attributo che rappresenta la relazione con l'itinerario;
  - `private IStatoAttivita statoAttivita`: attributo che rappresenta la relazione con lo stato dell'attività;
  - `private Set<IAccount> partecipanti`: attributo che rappresenta un insieme di account univoci che sono iscritti all'attività;
  - `private IAccount creatore`: attributo che rappresenta l'account creatore dell'attività.
- **Metodo/i:**
    - `public IAccount getCreatore()`;
    - `public Double getValoreRetribuzione()`;
    - `public String getTitolo()`;

- 
- public LocalDate getDataIncontro();
  - public LocalTime getOraIncontro();
  - public String getLingua();
  - public Retribuzione getRetribuzione();
  - public IStatoAttivita getStatoAttivita();
  - public Set<IAccount> getPartecipanti();
  - public String getIdAttivita();
  - public LocalDate getDataApertura();
  - public LocalDate getDataModifica();
  - public String getLuogo();
  - public void setLuogo(String luogo);
  - public void setDataModifica(LocalDate dataModifica);
  - public void setPartecipanti(Set<IAccount> partecipanti);
  - public void setStatoAttivita(IStatoAttivita statoAttivita);
  - public void modificaPuntoIncontro(Double x, Double y);
  - public void modificaDataIncontro(LocalDate dataIncontro);
  - public void modificaOraIncontro(LocalTime oraIncontro);
  - public String getDescrizioneAttivita();
  - public Double getXIncontro();
  - public Double getYIncontro();
  - public void aggiungiPartecipante(IAccount a);
  - public void eliminaPartecipante(IAccount a);
  - public void chiudiAttivita();
  - public Boolean isIscrivibile(IAccount a).

- 
- **BuilderAttivita:** classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Attivita* rendendo immutabili determinati attributi.
    - Attributo/i:
      - private LocalDate dataApertura;
      - private LocalDate dataModifica;
      - private String idAttivita;
      - private String titolo;
      - private Itinerario itinerario;
      - private IStatoAttivita statoAttivita;
      - private Set<IAccount> partecipanti;
      - private IAccount creatore.
    - Metodo/i:
      - public BuilderAttivita setIdAttivita(String idAttivita);
      - public BuilderAttivita setTitolo(String titolo): operazione che permette di impostare il titolo dell'attività. Solleva un'eccezione di tipo `IllegalArgumentException` se la lunghezza del campo non è compresa tra 0 e 100 caratteri.
      - public BuilderAttivita setItinerario(Itinerario itinerario);
      - public BuilderAttivita setCreatore(IAccount creatore);
      - public BuilderAttivita setDataApertura(LocalDate dataApertura);
      - public BuilderAttivita setDataModifica(LocalDate dataModifica);
      - public IAttivita build();

- 
- public BuilderAttivita generaldAttivita();
  - **StatoAttivitaFactory**: classe che implementa il Factory Method Pattern che funziona come interfaccia per l'istanziazione degli stati dell'attività.
  - **ProxyAttivita**: classe che implementa il *Proxy Pattern Design* che funziona come interfaccia per *Attivita* e permette di accedere ad alcune sue funzionalità attraverso dei permessi.
    - Attributo/i:
      - private static final String *PERMISSIONGLOBETROTTER* = "Globetrotter": permesso per l'account Globetrotter;
      - private static final String *PERMISSIONCICERONE* = "Cicerone": permesso per l'account Cicerone;
      - private static final String *ERRORMSG* = "Permessi insufficienti per l'utilizzo di questa operazione": messaggio di errore dovuto alla mancanza di permessi;
      - private Attivita attivita: attributo che rappresenta la relazione con l'attività a cui si riferisce lo stato.
      - private String ruolo: attributo che rappresenta il ruolo che un account può assumere.
    - Metodo/i:
      - public IAccount getCreatore();
      - public Double getValoreRetribuzione();
      - public String getTitolo();
      - public LocalDate getDataIncontro();
      - public LocalTime getOraIncontro();

- 
- `public String getLingua();`
  - `public Retribuzione getRetribuzione();`
  - `public IStatoAttivita getStatoAttivita();`
  - `public Set<IAccount> getPartecipanti();`
  - `public String getIdAttivita();`
  - `public LocalDate getDataApertura();`
  - `public LocalDate getDataModifica();`
  - `public String getLuogo();`
  - `public void setLuogo(String luogo);`
  - `public void setDataModifica(LocalDate dataModifica);`
  - `public void setPartecipanti(Set<IAccount> partecipanti);`
  - `public void setStatoAttivita(IStatoAttivita statoAttivita);`
  - `public void modificaPuntoIncontro(Double x, Double y);`
  - `public void modificaDataIncontro(LocalDate dataIncontro);`
  - `public void modificaOralIncontro(LocalTime oralIncontro);`
  - `public String getDescrizioneAttivita();`
  - `public Double getXIncontro();`
  - `public Double getYIncontro();`
  - `public void aggiungiPartecipante(IAccount a);`
  - `public void eliminaPartecipante(IAccount a);`
  - `public void chiudiAttivita();`
  - `public Boolean isIscrivibile(IAccount a).`

- 
- **Modificato:** classe che implementa l'interfaccia *IStatoAttivita*, secondo lo *State Pattern Design* e ne rappresenta lo stato di modificato.
    - Attributo/i:
      - private IAttivita attivita: attributo che rappresenta la relazione con l'attività a cui si riferisce lo stato.
    - Metodo/i:
      - public void aggiungi(IAccount a): operazione che permette l'aggiunta dell'account all'insieme dei partecipanti dell'attività.
      - public void elimina(IAccount a): operazione che permette la rimozione dell'account all'insieme dei partecipanti dell'attività.
      - public void modificaPunto(Double x, Double y): operazione che permette la modifica del punto d'incontro;
      - public void modificaData(LocalDate d): operazione che permette la modifica della data d'incontro;
      - public void modificaOra(LocalTime t): operazione che permette la modifica dell'ora dell'incontro;
      - public Boolean isIscrivibile(IAccount a): operazione che permette di verificare se l'account può iscriversi all'attività restituendo un *true* se è possibile e *false* in caso contrario.
  - **Aperto:** classe che implementa l'interfaccia *IStatoAttivita*, secondo lo *State Pattern Design* e ne rappresenta lo stato di apertura.



- 
- Attributo/i:
    - private IAttivita attivita: attributo che rappresenta la relazione con l'attività a cui si riferisce lo stato.
  - Metodo/i:
    - public void aggiungi(IAccount a): operazione che permette l'aggiunta dell'account all'insieme dei partecipanti dell'attività.
    - public void elimina(IAccount a): operazione che permette la rimozione dell'account all'insieme dei partecipanti dell'attività.
    - public void modificaPunto(Double x, Double y): operazione che permette la modifica del punto d'incontro;
    - public void modificaData(LocalDate d): operazione che permette la modifica della data d'incontro;
    - public void modificaOra(LocalTime t): operazione che permette la modifica dell'ora dell'incontro;
    - public Boolean isIscrivibile(IAccount a): operazione che permette di verificare se l'account può iscriversi all'attività restituendo un *true* se è possibile, *false* in caso contrario.
  - **Chiuso:** classe che implementa l'interfaccia *IStatoAttivita*, secondo lo *State Pattern Design* e ne rappresenta lo stato di chiusura.
    - Attributo/i:
      - private IAttivita attivita: attributo che rappresenta la relazione con l'attività a cui si riferisce lo stato.

- 
- Metodo/i:
    - public void aggiungi(IAccount a): operazione che solleva un'eccezione di tipo IllegalArgumentException;
    - public void elimina(IAccount a): operazione che solleva un'eccezione di tipo IllegalArgumentException;
    - public void modificaPunto(Double x, Double y): operazione che solleva un'eccezione di tipo IllegalArgumentException;
    - public void modificaData(LocalDate d): operazione che solleva un'eccezione di tipo IllegalArgumentException;
    - public void modificaOra(LocalTime t): operazione che solleva un'eccezione di tipo IllegalArgumentException;
    - public Boolean isIscrivibile(IAccount a): operazione che solleva un'eccezione di tipo IllegalArgumentException.
  - **Account:** classe che rappresenta e gestisce le informazioni relative alla digitalizzazione di utente sulla piattaforma.
    - Attributo/i:
      - private String idAccount: attributo che permette l'identificazione univoca dell'attività. L'utilizzo del tipo String è dovuto all'unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;
      - private String email: attributo che rappresenta l'indirizzo di posta elettronica per le comunicazioni sulla piattaforma e lo scambio di informazioni;



- 
- `private String password`: attributo che rappresenta la password univoca usata per l'accesso all'account sulla piattaforma;
  - `private IAnagrafica anagrafica`: attributo che rappresenta la relazione con la classe Anagrafica;
  - `private String ruolo`: attributo che rappresenta il ruolo dell'account: "Cicerone", "Globetrotter";
  - `private Long contatoreNotifiche`: attributo che rappresenta il numero di notifiche non ancora lette collegate all'account.
  - Metodo/i:
    - `public String getIdAccount()`;
    - `public String getEmail()`;
    - `public String getPassword()`;
    - `public IAnagrafica getAnagrafica()`;
    - `public Long getContatoreNotifiche()`;
    - `public String getRuolo()`;
    - `public void cambiaPassword(String password)`;
    - `public void aggiungiLingua(String lingua)`;
    - `public void modificaResidenza(String paese)`;
    - `public void modificaNumeroDiTelefono(String numeroDiTelefono)`;
    - `public String visualizzaRuolo()`: attributo che restituisce il ruolo dell'account aggiornato all'ultimo accesso;

- 
- **BuilderAccount:** classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Account* rendendo immutabili determinati attributi.
    - Attributo/i:
      - private String idAccount;
      - private String email;
      - private String password;
      - private String ruolo;
      - private IAnagrafica anagrafica;
      - private Long contatoreNotifiche.
    - Metodo/i:
      - public BuilderAccount setIdAccount(String idAccount);
      - public BuilderAccount setEmail(String email);
      - public BuilderAccount setPassword(String password);
      - public BuilderAccount setAnagrafica(IAnagrafica anagrafica);
      - public BuilderAccount setRuolo(String ruolo);
      - public IAccount build();
      - public BuilderAccount generaldAccount();
      - public BuilderAccount hashPassword(String password).
  - **IdGenerator:** classe *Utility* che permette di generare gli id tutte le entità principali.
    - Metodo/i:
      - public String generald().

- 
- **HashnSalt:** classe *Utility* (con metodi di tipo *static*) che permette di codificare e convalidare le password utilizzando la tecnica di *hash* e *salt*.
    - Metodo/i:
      - `public String encode(String password, String salt):` operazione che prende in input la password dell'utente concatenata al *salt* per generare l'*hash* della password;
      - `public String generaSalt():` operazione che genera casualmente una stringa alfanumerica da 29 caratteri che rappresenta il *salt*;
      - `public String estrapolaSalt(String hash):` operazione che prende in input la password hashata e restituisce la stringa di *salt*.
  - **RicercaOdierna:** classe che implementa l'interfaccia */GestoreRicerca* che permette di filtrare una lista di attività.
    - Attributo/i:
      - `private List<IAttivita> listaAttivita;`
      - `private LocalDate dataOdierna.`
    - Metodo/i:
      - `public List<IAttivita> ricerca():` operazione che restituisce una lista di attività filtrate per data odierna.
  - **RicercaDecoratore:** classe astratta che rappresenta l'implementazione del *Decorator Pattern Design*.
    - Attributo/i:



- 
- private IGestoreRicerca ricerca: attributo che deriva dall'implementazione del Decorator Pattern Design tramite la relazione tra RicercaDecoratore e IGestoreRicerca;
  - Metodo/i:
    - public abstract List<IAttivita> ricerca();
  - **RicercaLuogo:** classe che implementa *RicercaDecoratore* e che funziona da wrapper ad altre classi che implementano *IGestoreRicerca* in modo tale da aggiungere ulteriori filtri e scremare la ricerca.
    - Attributo/i:
      - private String luogo.
    - Metodo/i:
      - public List<IAttivita> ricerca(): operazione che restituisce una lista di attività filtrate per luogo.
  - **RicercaData:** classe che implementa *RicercaDecoratore* e che funziona da wrapper ad altre classi che implementano *IGestoreRicerca* in modo tale da aggiungere ulteriori filtri e scremare la ricerca.
    - Attributo/i:
      - private LocalDate dataRicerca.
    - Metodo/i:
      - public List<IAttivita> ricerca(): operazione che restituisce una lista di attività filtrate per data.
  - **RicercaRetribuzione:** classe che implementa *RicercaDecoratore* e che funziona da wrapper ad altre classi che implementano



---

*IGestoreRicerca* in modo tale da aggiungere ulteriori filtri e scremare la ricerca.

- Metodo/i:
  - `public List<IAttivita> ricerca():` operazione che restituisce una lista di attività filtrate per retribuzione.
- **RicercaLingua:** classe che implementa *RicercaDecoratore* e che funziona da wrapper ad altre classi che implementano *IGestoreRicerca* in modo tale da aggiungere ulteriori filtri e scremare la ricerca.
  - Attributo/i:
    - `private String lingua.`
  - Metodo/i:
    - `public List<IAttivita> ricerca():` operazione che restituisce una lista di attività filtrate per lingua.
- **Anagrafica:** classe che rappresenta e gestisce le informazioni relative alla caratterizzazione di un utente.
  - Attributo/i:
    - `private String idAnagrafica:` attributo che permette l'identificazione univoca dell'anagrafica. L'utilizzo del tipo `String` è dovuto all'unicità del campo infatti questo tipo permette di avere un campo di valori univoci maggiori agli altri tipi;
    - `private String nome:` attributo che rappresenta il nome dell'utente;
    - `private String cognome:` attributo che rappresenta il cognome dell'utente;

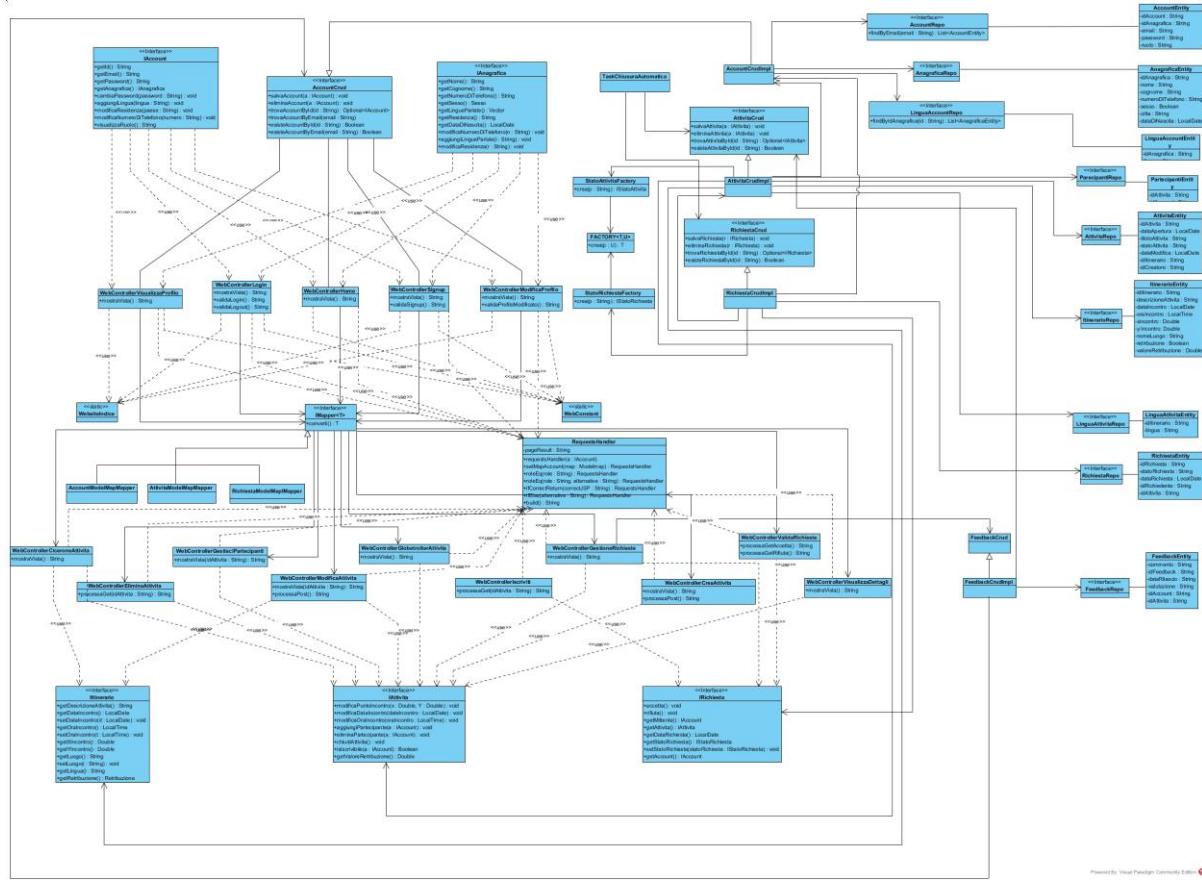


- 
- `private String numeroDiTelefono;` attributo che rappresenta il numero di telefono dell'utente in forma testuale perché ne rende più semplice la verifica;
  - `private Sesso sesso;` attributo che rappresenta il sesso dell'utente;
  - `private Set<String> lingueParlate;` attributo che rappresenta l'insieme delle lingue parlate;
  - `private String residenza;` attributo che rappresenta il paese di residenza;
  - `private LocalDate dataDiNascita;` attributo che rappresenta la data di nascita.
- Metodo/i:
    - `public String getIdAnagrafica();`
    - `public String getNome();`
    - `public String getCognome();`
    - `public String getNumeroDiTelefono();`
    - `public Sesso getSesso();`
    - `public Set<String> getLingueParlate();`
    - `public String getResidenza();`
    - `public LocalDate getDataDiNascita();`
    - `public void modificaNumeroDiTelefono(String numeroDiTelefono);` operazione che permette di modificare il numero di telefono. Solleva un'eccezione di tipo `IllegalArgumentException` se il numero di telefono non è una stringa numerica con lunghezza 10.

- 
- `public void aggiungiLingueParlate(String lingua):`  
operazione che permette l'aggiunta di una nuova lingua alla lista di lingue parlate. Solleva un'eccezione di tipo `IllegalArgumentException` se la lingua è già presente nell'insieme e ha una lunghezza non compresa tra 0 e 100 caratteri.
  - `public void modificaResidenza(String r):` operazione che permette la modifica del paese di residenza. Solleva un'eccezione di tipo `IllegalArgumentException` se la lunghezza del paese non è compreso tra 0 e 100 caratteri.
  - **BuilderAnagrafica:** classe che implementa il *Builder Pattern Design* per creare un'istanza della classe *Anagrafica* rendendo immutabili determinati attributi.
    - Attributo/i:
      - `private String idAnagrafica;`
      - `private String nome;`
      - `private String cognome;`
      - `private String numeroDiTelefono;`
      - `private Sesso sesso;`
      - `private Set<String> lingueParlate;`
      - `private String residenza;`
      - `private LocalDate dataDiNascita.`
    - Metodo/i:
      - `public BuilderAnagrafica setIdAnagrafica(String idAnagrafica);`



- 
- `public BuilderAnagrafica setNome(String nome):`  
operazione che permette l'impostazione del nome.  
Solleva un'eccezione di tipo `IllegalArgumentException` se  
il campo non comincia con lettera maiuscola e non è più  
lungo di quattro caratteri;
  - `public BuilderAnagrafica setCognome(String cognome):`  
operazione che permette l'impostazione del cognome.  
Solleva un'eccezione di tipo `IllegalArgumentException` se  
il campo non comincia con lettera maiuscola e non è più  
lungo di quattro caratteri;
  - `public BuilderAnagrafica setNumeroDiTelefono(String numeroDiTelefono):` operazione che permette di  
modificare il numero di telefono. Solleva un'eccezione di  
tipo `IllegalArgumentException` se il numero di telefono  
non è una stringa numerica con lunghezza 10;
  - `public BuilderAnagrafica setSesso(Sesso sesso);`
  - `public BuilderAnagrafica setLingueParlate(Set<String> LingueParlate);`
  - `public BuilderAnagrafica setResidenza(String residenza);`
  - `public BuilderAnagrafica setDataDiNascita(LocalDate dataDiNascita);`
  - `public IAnagrafica build();`
  - `public BuilderAnagrafica generaldAnagrafica().`
- **ErrorMessage:** classe `utils` utilizzata dalle classi di business che  
conterrà attributi costanti per i messaggi di errore.



- **AccountCrudImpl**: classe che implementa l'interfaccia `AccountCrud` che permette di eseguire le operazioni di crud sull'Account mappandolo alla rappresentazione corrispettiva all'interno del database.
- **AttivitàCrudImpl**: classe che implementa l'interfaccia `AttivitàCrud` che permette di eseguire le operazioni di crud sull'Attività mappandola alla rappresentazione corrispettiva all'interno del database.
- **RichiestaCrudImpl**: classe che implementa l'interfaccia `RichiestaCrud` che permette di eseguire le operazioni di crud sulla Richiesta

---

mappandola alla rappresentazione corrispettiva all'interno del database.

- **FeedbackCrudImpl:** classe che implementa l'interfaccia *FeedbackCrud* che permette di eseguire le operazioni di crud sul Feedback mappandolo alla rappresentazione corrispettiva all'interno del database.
- **TaskChiusuraAutomatica:** classe che deriva da *SpringBoot* e che chiude automaticamente alle 23:59 le attività che hanno data di incontro odierna.
- **WebControllerVisualizzaProfilo:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *VisualizzaProfilo*.
- **WebControllerLogin:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *Login*.
- **WebControllerHome:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *Home*.
- **WebControllerSignup:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *Signup*.
- **WebControllerModificaProfilo:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *ModificaProfilo*.



- 
- **WebsitelIndice:** classe statica che ha attributi pubblici che rappresentano le costanti degli indici dei siti.
  - **WebConstant:** classe statica che ha attributi pubblici che rappresentano attributi html visualizzati nelle viste.
  - **AccountModelMapMapper:** classe che permette di mappare l'*Account* in un dato visualizzabile all'interno delle viste.
  - **AttivitaModelMapMapper:** classe che permette di mappare l'*Attivita* in un dato visualizzabile all'interno delle viste.
  - **RichiestaModelMapMapper:** classe che permette di mappare la *Richiesta* in un dato visualizzabile all'interno delle viste.
  - **WebControllerCiceroneAttivita:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *CiceroneAttivita*.
    - Metodo/i:
      - public String mostraVista().
  - **WebControllerEliminaAttivita:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *EliminaAttivita*.
    - Metodo/i:
      - public String processaGet(String idAttivita).
  - **WebControllerGestisciPartecipanti:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *GestisciPartecipanti*.
    - Metodo/i:
      - public String mostraVista(String idAttivita).



- 
- **WebControllerModificaAttivita:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *ModificaAttivita*.
    - Metodo/i:
      - public String mostraVista(String idAttivita);
      - public String processaPost();
  - **WebControllerGlobetrotterAttivita:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *GlobetrotterAttivita*.
    - Metodo/i:
      - public String mostraVista();
  - **WebControllerIscriviti:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *Iscriviti*.
    - Metodo/i:
      - public String processaGet(String idAttivita);
  - **WebControllerGestioneRichieste:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *GestioneRichieste*.
    - Metodo/i:
      - public String mostraVista();
  - **WebControllerCiceroneRichieste:** classe servlet configurata automaticamente da Spring framework per il binding tra servizi e url del sito. Il servizio bindato è *CiceroneRichieste*.
    - Metodo/i:
      - public String mostraVista();



- 
- public String processalscrizione(String idAccount);
  - public String processaRifiuto(String idAccount).
  - **StatoRichiestaFactory**: classe che implementa il pattern creazionale *Factory Pattern Design* che permette la creazione degli stati della *Richiesta*:
    - Metodo/i:
      - public IStatoRichiesta crea(String p).
  - **StatoAttivitaFactory**: classe che implementa il pattern creazionale *Factory Pattern Design* che permette la creazione degli stati dell' *Attivita*:
    - Metodo/i:
      - public IStatoAttivita crea(String p).
  - **AccountRepo**: interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Account* nel database.
  - **AccountEntity**: classe che rappresenta una tupla della tabella Account all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
    - Attributo/i:
      - private String idAccount;
      - private String idAnagrafica;
      - private String email;
      - private Integer numeroNotifiche;
      - private String password;

- 
- `private String ruolo;`
  - **AnagraficaRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Anagrafica* nel database.
  - **AnagraficaEntity:** classe che rappresenta una tupla della tabella Anagrafica all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
    - Attributo/i:
      - `private String idAnagrafica;`
      - `private String nome;`
      - `private String cognome;`
      - `private String numeroDiTelefono;`
      - `private Boolean sesso;`
      - `private String citta;`
      - `private LocalDate dataDiNascita.`
  - **LinguaAccountRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *LinguaAccount* nel database.
  - **LinguaAccountEntity:** classe che rappresenta una tupla della tabella LinguaAccount all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e

---

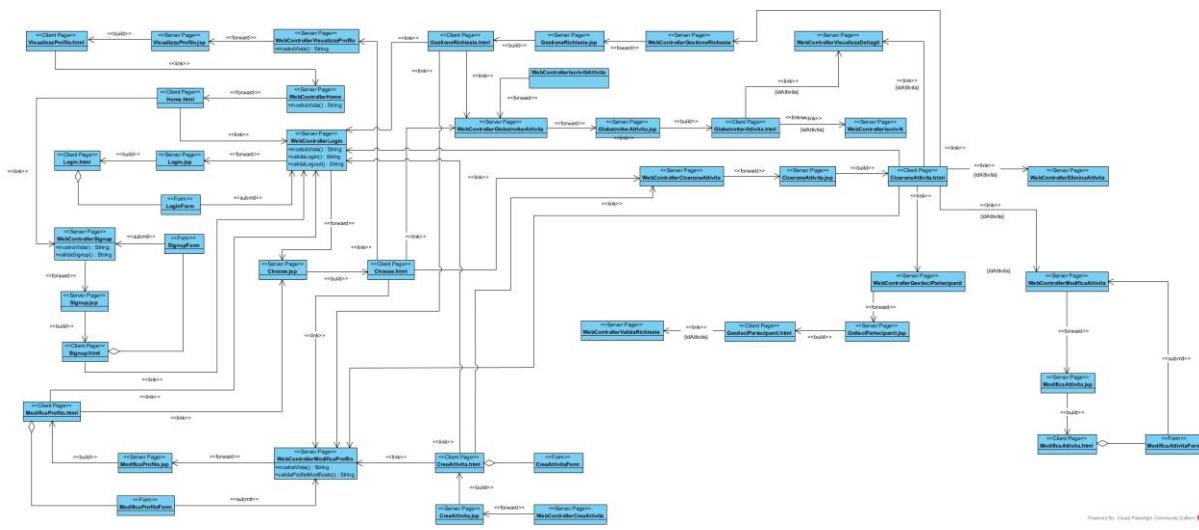
setters per permettere ad *Hibernate* di impostarne automaticamente il valore.

- Attributo/i:
  - private String idAnagrafica;
- **PartecipantiRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Partecipanti* nel database.
- **PartecipantiEntity:** classe che rappresenta una tupla della tabella Partecipanti all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
  - Attributo/i:
    - private String idAttivita;
    - private String idAccount.
- **AttivitaRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Attivita* nel database.
- **AttivitaEntity:** classe che rappresenta una tupla della tabella Attivita all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
  - Attributo/i:

- 
- `private String idAttivita;`
  - `private LocalDate dataApertura;`
  - `private String titoloAttivita;`
  - `private String statoAttivita;`
  - `private LocalDate dataModifica;`
  - `private String idItinerario;`
  - `private String idCreatore.`
  - **FeedbackRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Feedback* nel database.
  - **FeedbackEntity:** classe che rappresenta una tupla della tabella Feedback all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
    - Attributo/i:
      - `private String commento;`
      - `private String idFeedback;`
      - `private String dataRilascio;`
      - `private Integer valutazione;`
      - `private String idAccount;`
      - `private String idAttivita.`
  - **ItinerarioRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Itinerario* nel database.

- 
- **ItinerarioEntity:** classe che rappresenta una tupla della tabella Itinerario all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
    - Attributo/i:
      - private String idItinerario;
      - private String descrizioneAttivita;
      - private LocalDate dataIncontro;
      - private LocalTime oraIncontro;
      - private Double xIncontro;
      - private Double yIncontro;
      - private String nomeLuogo;
      - private Boolean retribuzione;
      - private Double valoreRetribuzione.
  - **LinguaAttivitaRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *LinguaAttivita* nel database.
  - **LinguaAttivitaEntity:** classe che rappresenta una tupla della tabella LinguaAttivita all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
    - Attributo/i:
      - private String idItinerario;

- `private String lingua.`
- **RichiestaRepo:** interfaccia che estende la funzionalità di Repository Crud di Spring framework e che permette di autogenerare a run-time la classe corrispondente per effettuare operazioni di crud sulla tabella *Richiesta* nel database.
- **RichiestaEntity:** classe che rappresenta una tupla della tabella Richiesta all'interno del database. Ogni attributo corrisponde ad un campo della tabella e per ognuno di essi ci sono getters e setters per permettere ad *Hibernate* di impostarne automaticamente il valore.
  - Attributo/i:
    - `private String idRichiesta;`
    - `private String statoRichiesta;`
    - `private LocalDate dataRichiesta;`
    - `private String idRichiedente;`
    - `private String idAttivita.`



- 
- **Server Pages:** classi servlet configurate automaticamente da Spring framework per il binding tra servizi e url dei siti.
    - *Controllers:*
      - WebControllerHome;
      - WebControllerVisualizzaProfilo;
      - WebControllerLogin;
      - WebControllerSignup;
      - WebControllerModificaProfilo;
      - WebControllerCiceroneAttivita;
      - WebControllerEliminaAttivita;
      - WebControllerModificaAttivita;
      - WebControllerGestisciPartecipanti;
      - WebControllerGlobetrotterAttivita;
      - WebControllerIscriviti;
      - WebControllerGestioneRichieste;
      - WebControllerCreaAttivita;
      - WebControllerValidaRichieste;
      - WebControllerVisualizzaDettagli;
    - *Files .jsp:*
      - VisualizzaProfilo.jsp: genera la pagina VisualizzaProfilo.html mediante la mappatura dei dati del WebControllerVisualizzaProfilo;
      - Login.jsp: genera la pagina Login.html mediante la mappatura dei dati del WebControllerLogin;



- 
- Choose.jsp: genera la pagina Choose.html mediante la mappatura dei dati del WebControllerLogin e del WebControllerSignup;
  - Signup.jsp: genera la pagina Signup.html mediante la mappatura dei dati del WebControllerSignup;
  - ModificaProfilo.jsp: genera la pagina ModificaProfilo.html mediante la mappatura dei dati del WebControllerModificaProfilo;
  - CiceroneAttivita.jsp: genera la pagina CiceroneAttivita.html;
  - ModificaAttivita.jsp: genera la pagina ModificaAttivita.html mediante la mappatura dei dati del WebControllerModificaAttivita;
  - GestisciPartecipanti.jsp: genera la pagina GestisciPartecipanti.html;
  - GlobetrotterAttivita.jsp: genera la pagina GlobetrotterAttivita.html mediante la mappatura dei dati delle Attività che il Globetrotter può ricercare;
  - GestioneRichieste.jsp: genera la pagina GestioneRichieste.html;
  - CreaAttivita.jsp: genera la pagina CreaAttivita.html.
- **Client Pages:**
    - *Home.html*: vista statica generata da Home.jsp;
    - *VisualizzaProfilo.html*: vista statica generata da VisualizzaProfilo.jsp;
    - *Login.html*: vista statica generata da Login.jsp;

- 
- *Choose.html*: vista statica generata da Choose.jsp;
  - *Signup.html*: vista statica generata da Signup.jsp;
  - *ModificaProfilo.html*: vista statica generata da ModificaProfilo.jsp;
  - *CiceroneAttivita.html*: vista statica generata da CiceroneAttivita.jsp;
  - *ModificaAttivita.html*: vista statica generata da ModificaAttivita.jsp;
  - *GestisciPartecipanti.html*: vista statica generata da GestisciPartecipanti.jsp;
  - *GlobetrotterAttivita.html*: vista statica generata da GlobetrotterAttivita.jsp;
  - *GestioneRichieste.html*: vista statica generata da GestioneRichieste.jsp;
  - *CreaAttivita.html*: vista statica generata da CreaAttivita.jsp.

- **Forms:**

- *LoginForm*: componente di una pagina web che permette agli utenti l'inserimento dei dati di login che verranno inviati al server per essere processati;
- *SignupForm*: componente di una pagina web che permette agli utenti l'inserimento dei dati di signup che verranno inviati al server per essere processati;
- *ModificaProfiloForm*: componente di una pagina web che permette agli utenti l'inserimento dei dati delle

---

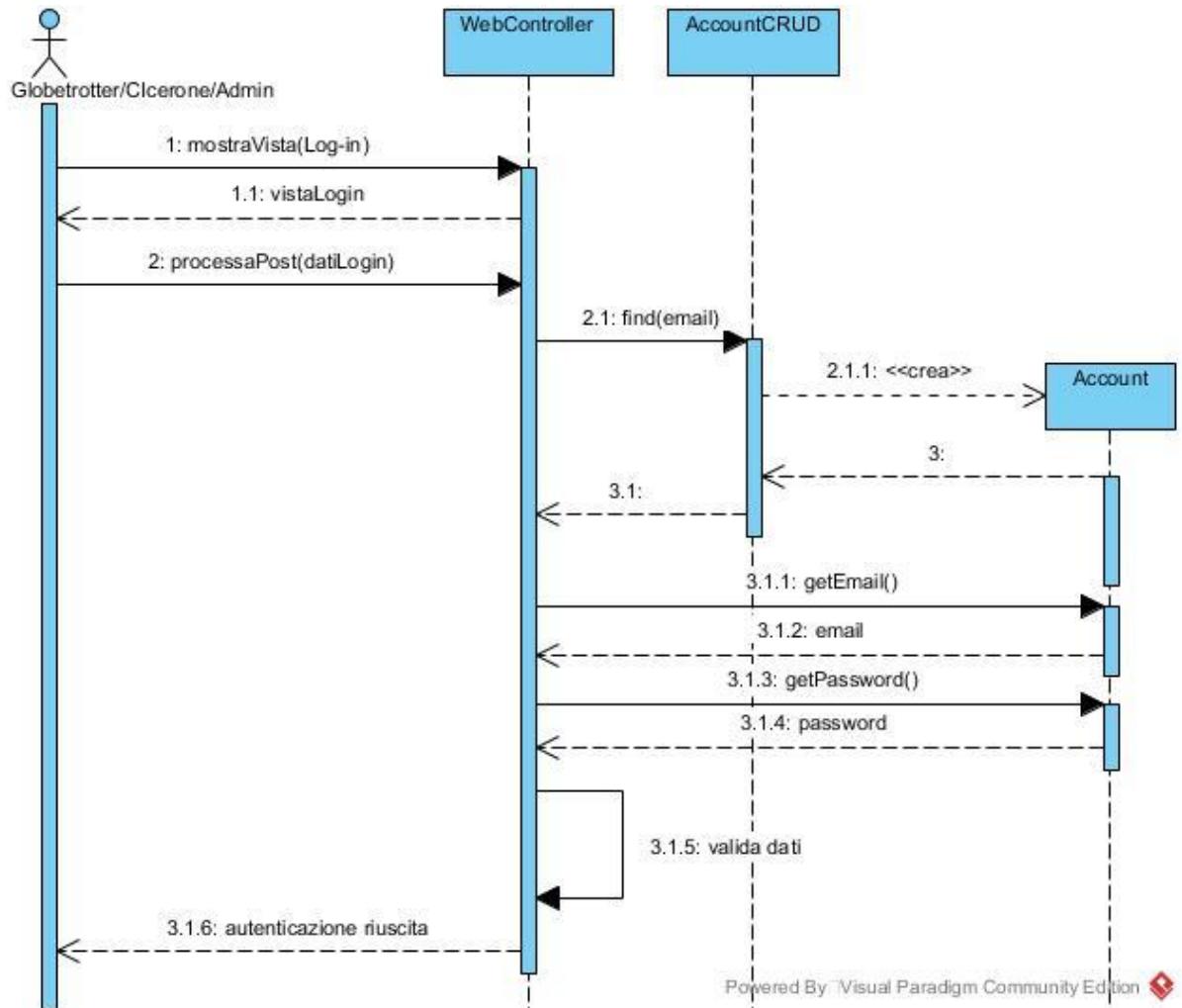
modifiche da effettuare sul profilo che verranno inviati al server per essere processati;

- *ModificaAttivitaForm*: componente di una pagina web che permette agli utenti l'inserimento dei dati delle modifiche da effettuare sull'attività che verranno inviati al server per essere processati;
- *CreaAttivitaForm*: componente di una pagina web che permette agli utenti l'inserimento dei dati per la creazione di un'attività che verranno inviati al server per essere processati.

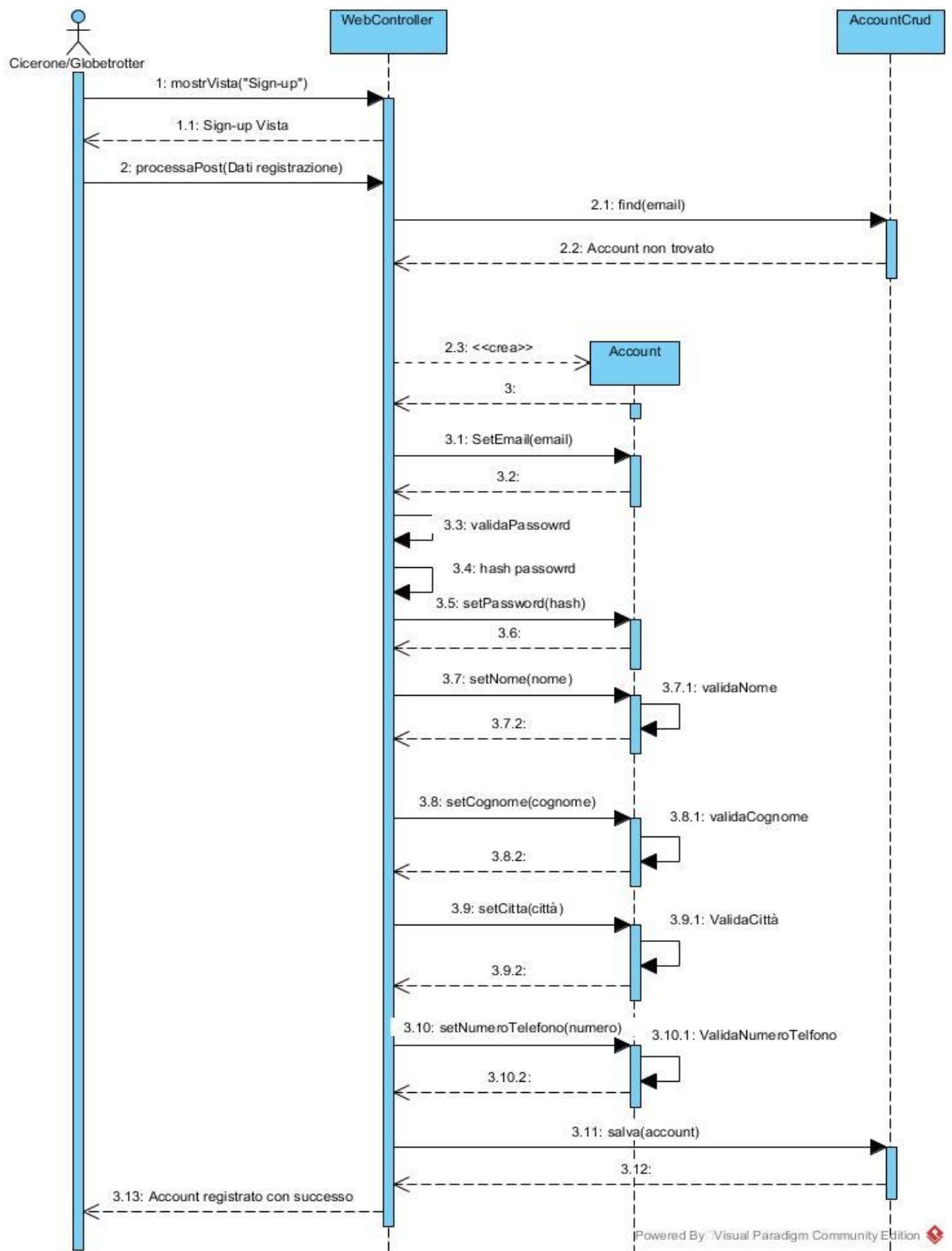


## 3.5 Diagrammi di sequenza

- Autentica Account (DS\_1)

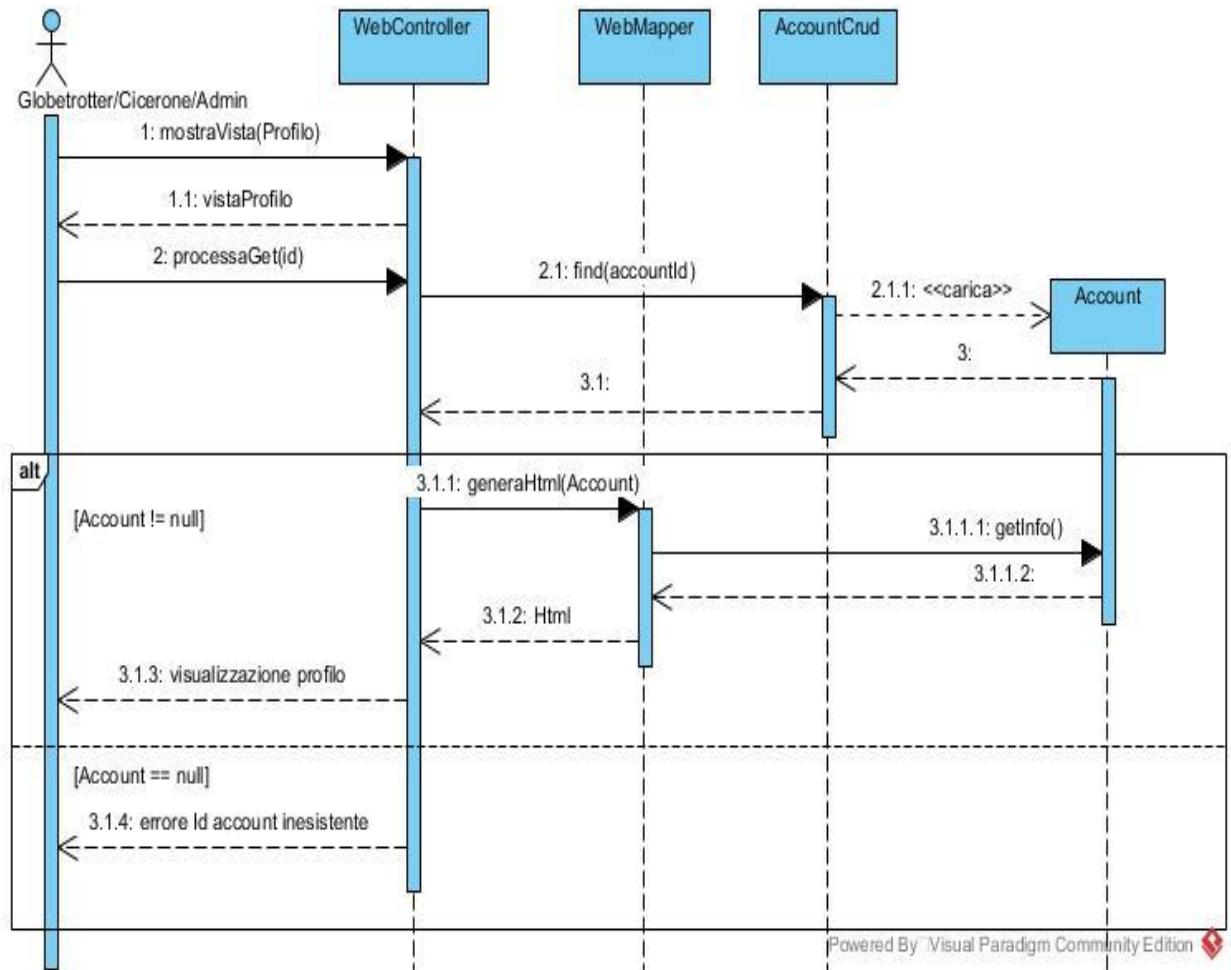


- **Registra Account (DS\_2)**

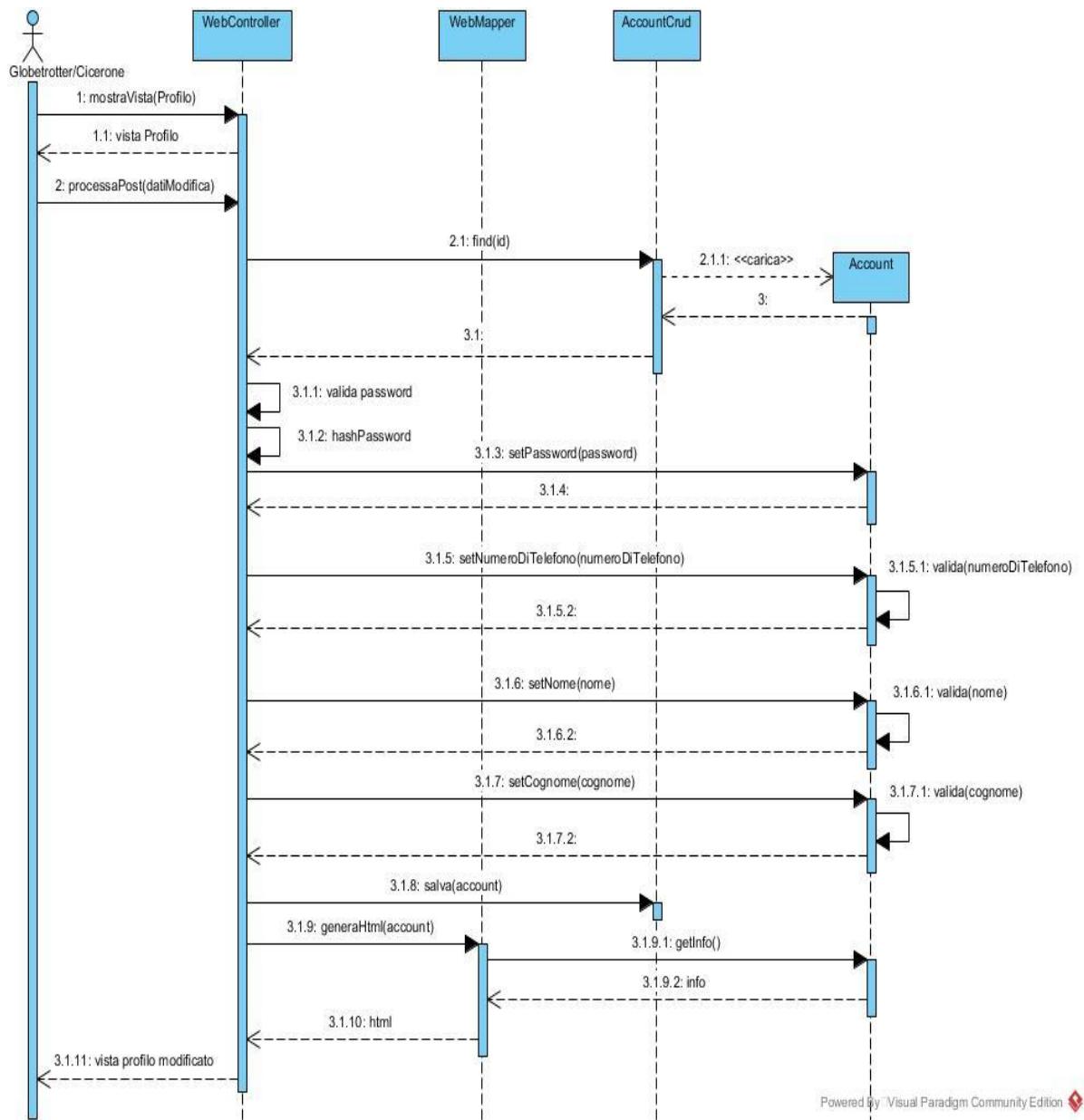




- **Visualizza Profilo (DS\_3)**

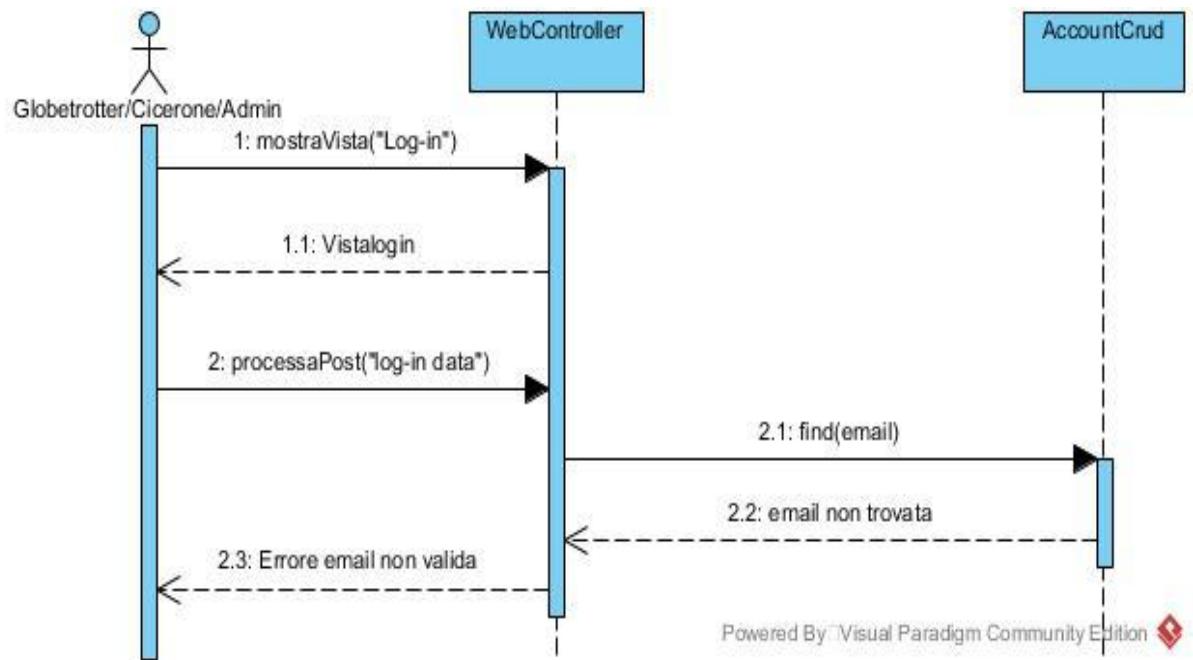


- **Modifica Profilo (DS\_4)**



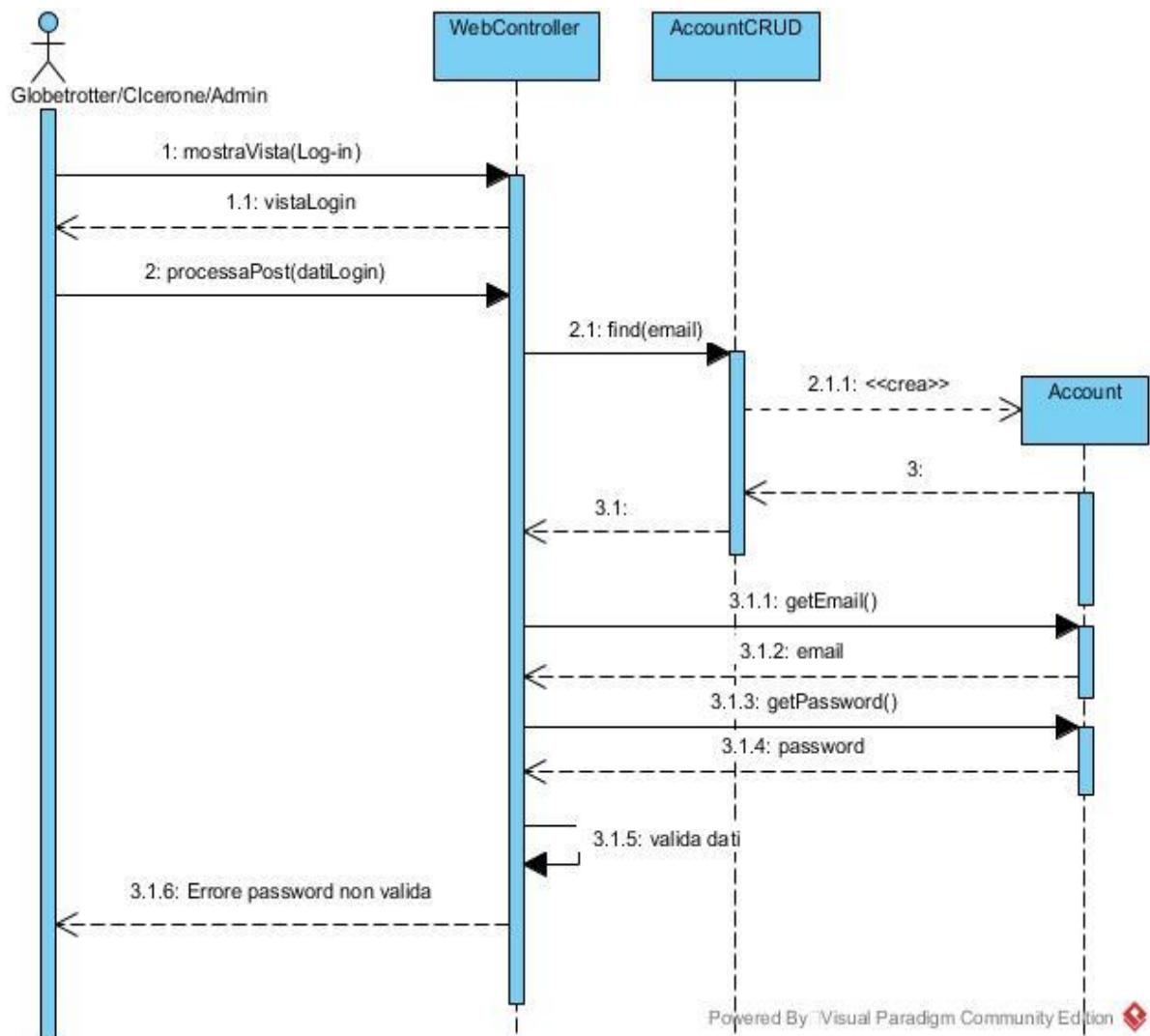
Powered by Visual Paradigm Community Edition

- Autentica Account: EmailNonValida (DS\_1.1)

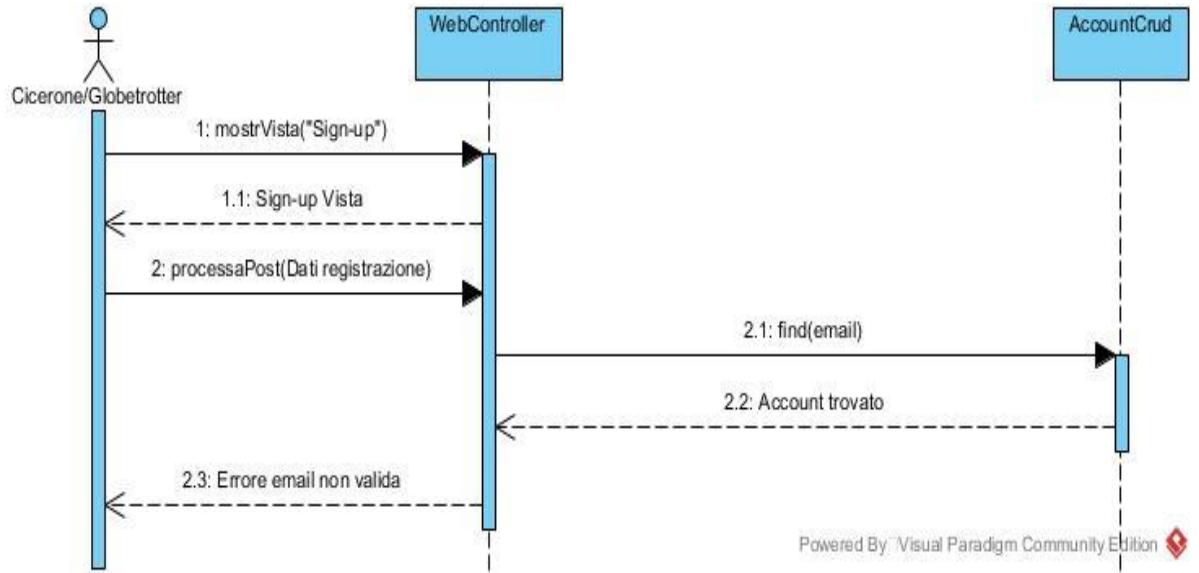




- Autentica Account: PasswordNonValida (DS\_1.2)

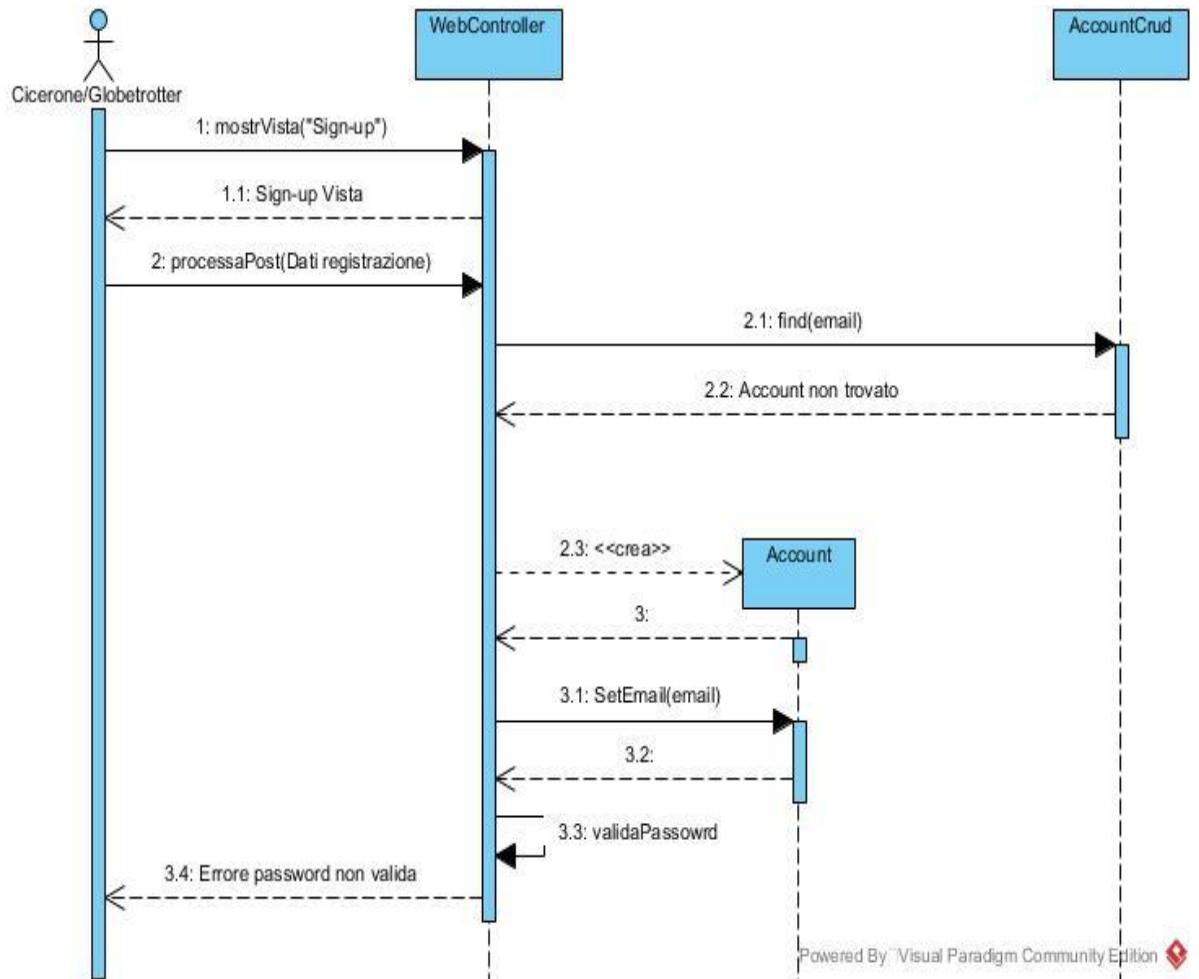


- **Registra Account: EmailNonValida (DS\_2.1)**

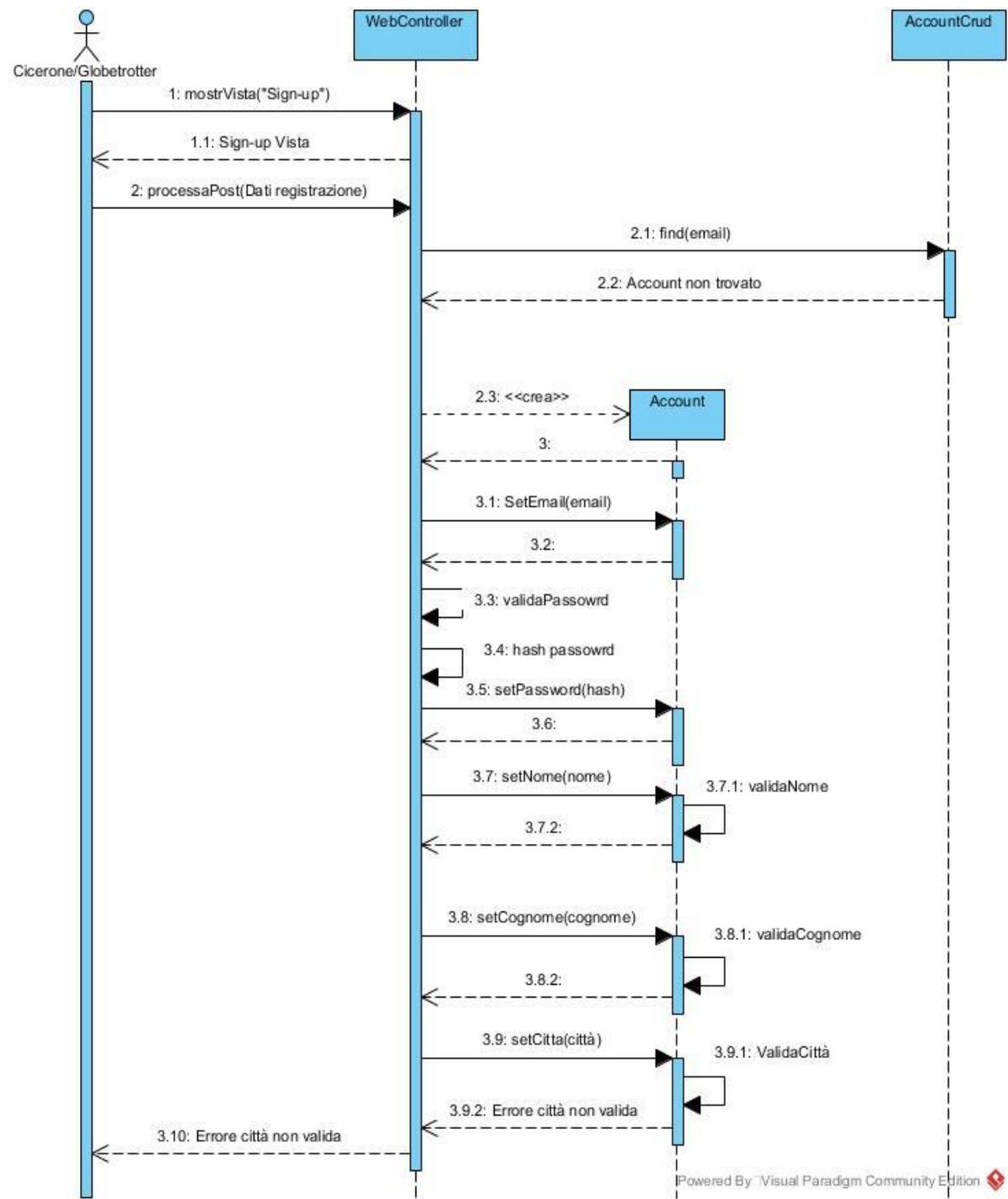




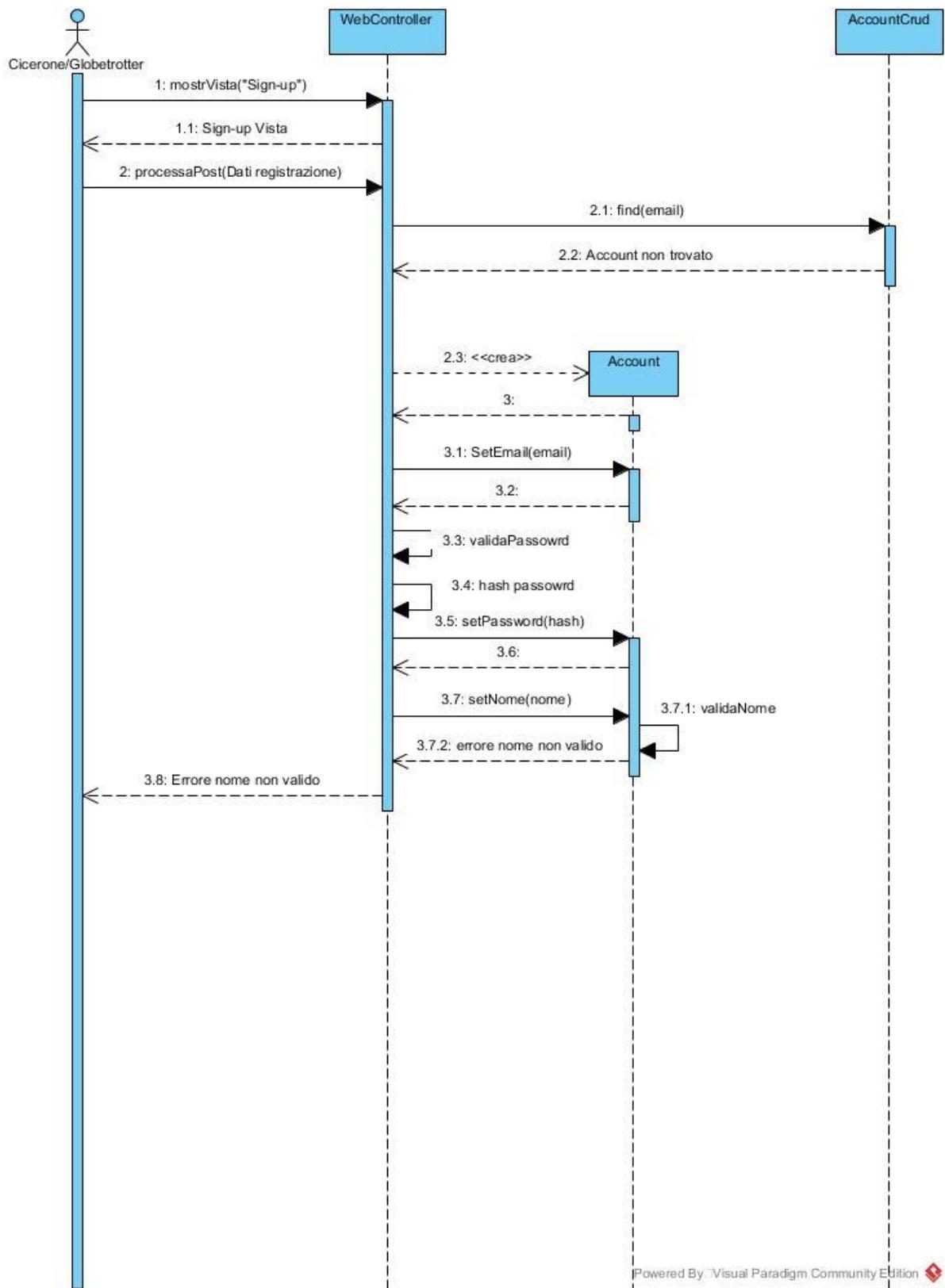
- **Registra Account: PasswordNonValida (DS\_2.2)**



- **Registra Account: CittàNonValida (DS\_2.3)**

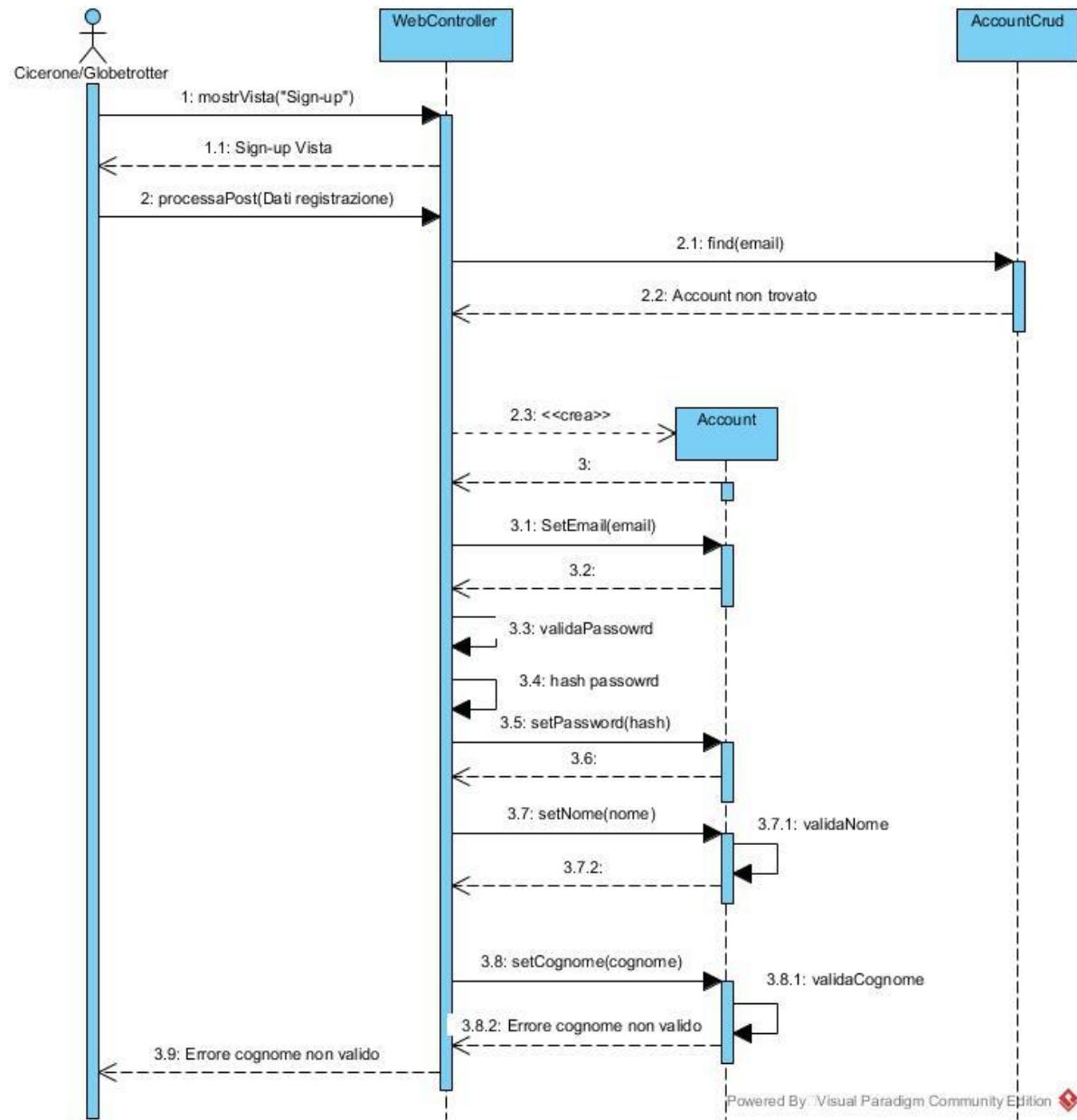


- **Registra Account: NomeNonValido (DS\_2.4)**

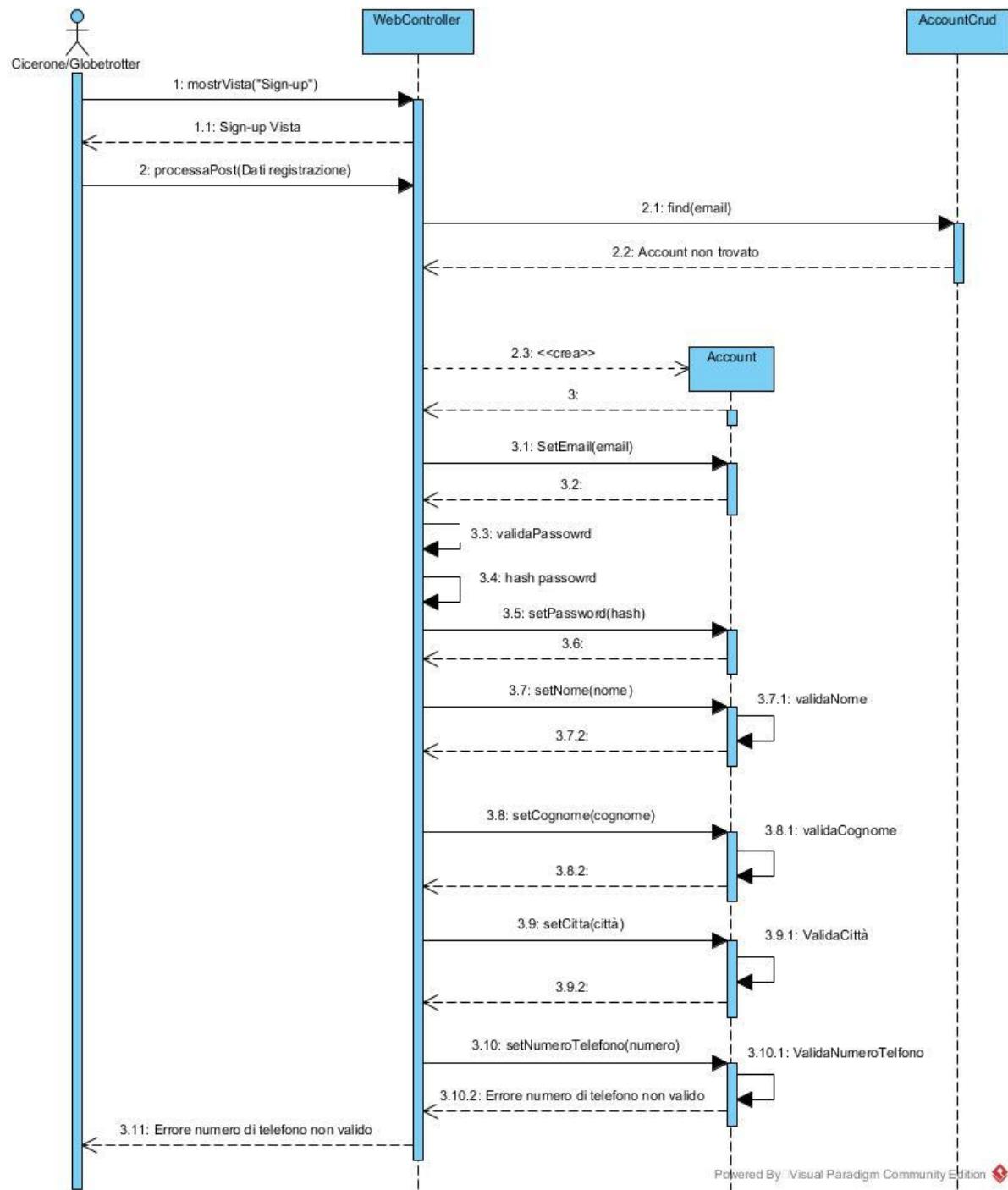




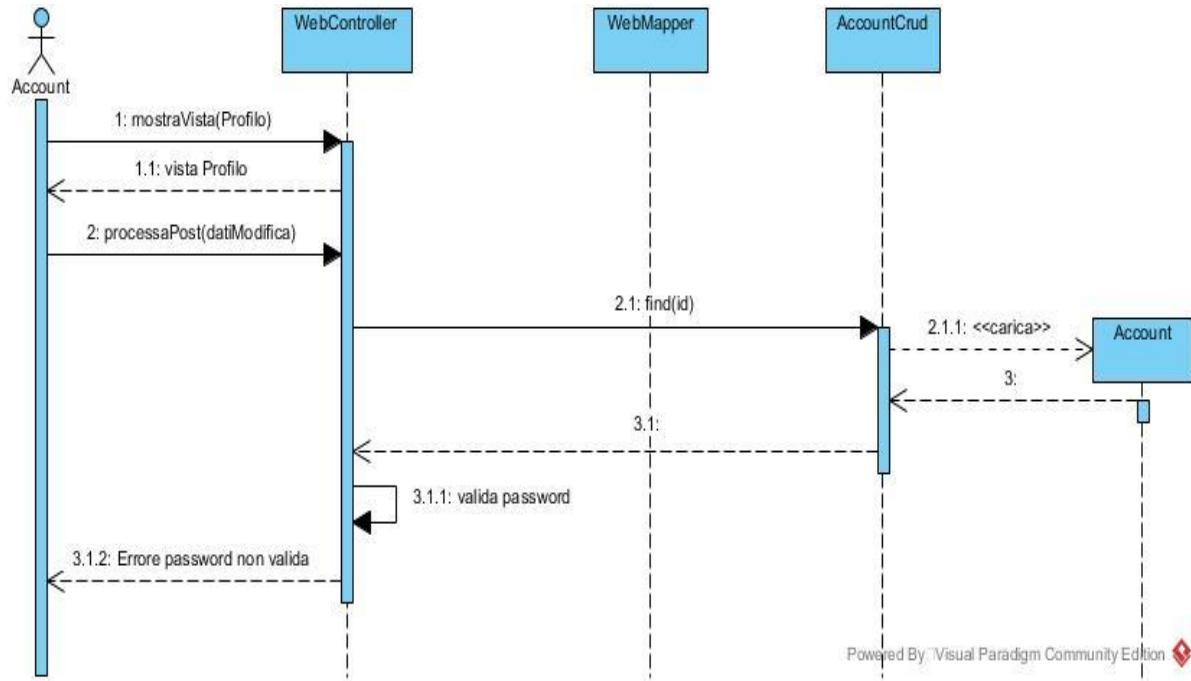
- **Registra Account: CognomeNonValido (DS\_2.5)**



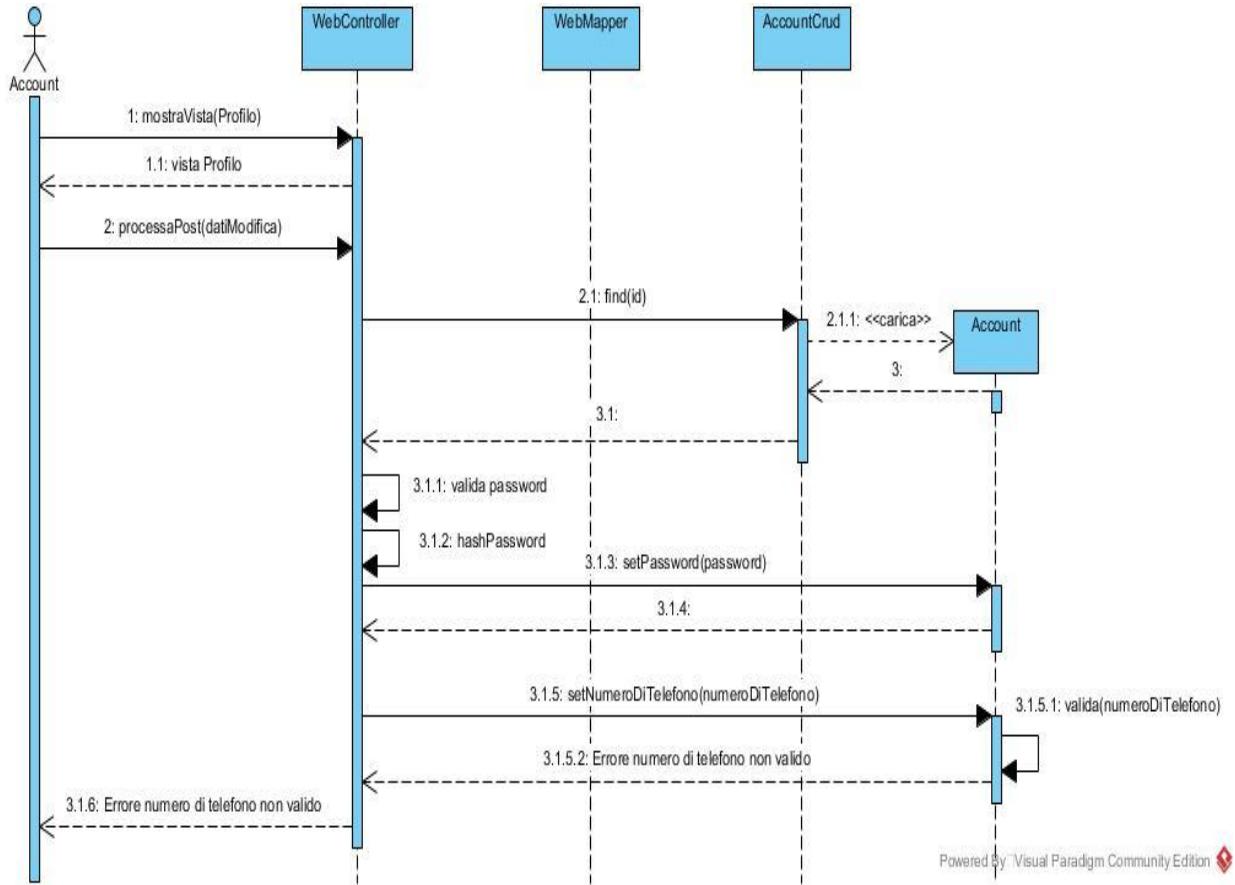
- **Registra Account: NumeroDiTelefonoNonValido (DS\_2.6)**



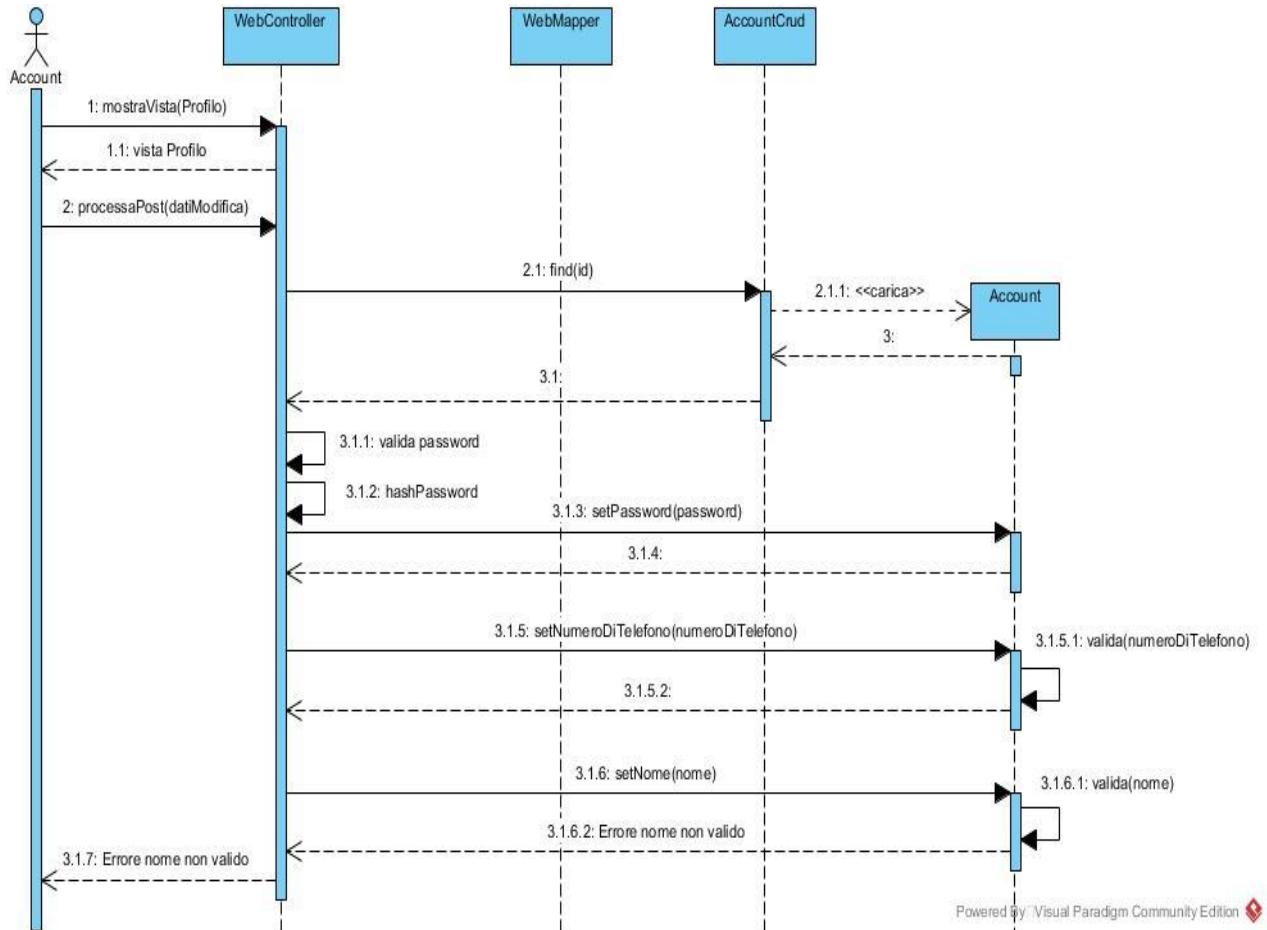
- **Modifica Profilo: PasswordNonValida (DS\_4.1)**



- **Modifica Profilo: NumeroDiTelefonoNonValido (DS\_4.2)**

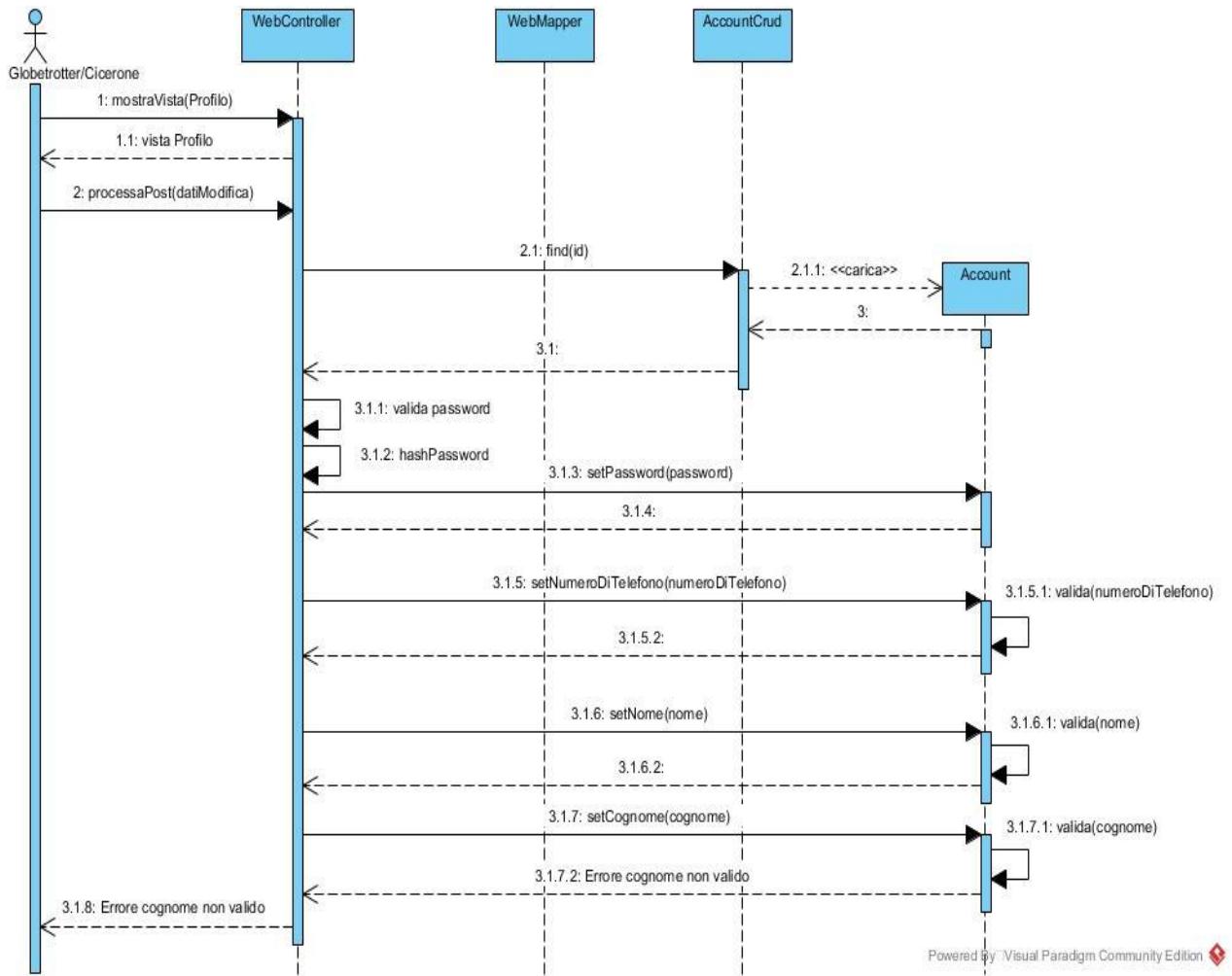


- **Modifica Profilo: NomeNonValido (DS\_4.3)**





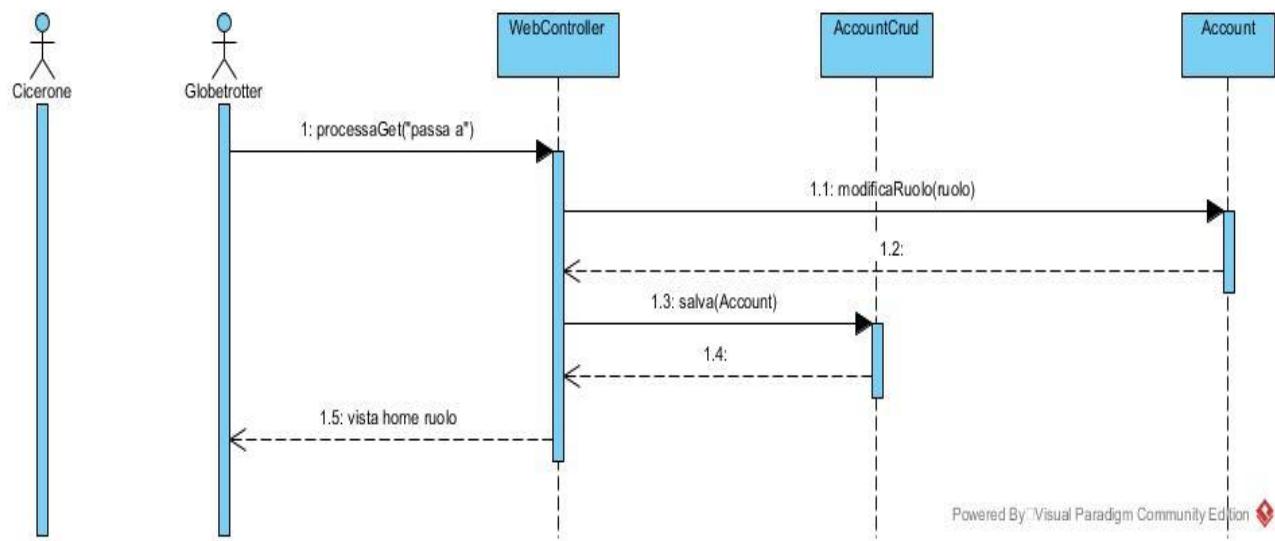
- **Modifica Profilo: CognomeNonValido (DS\_4.4)**



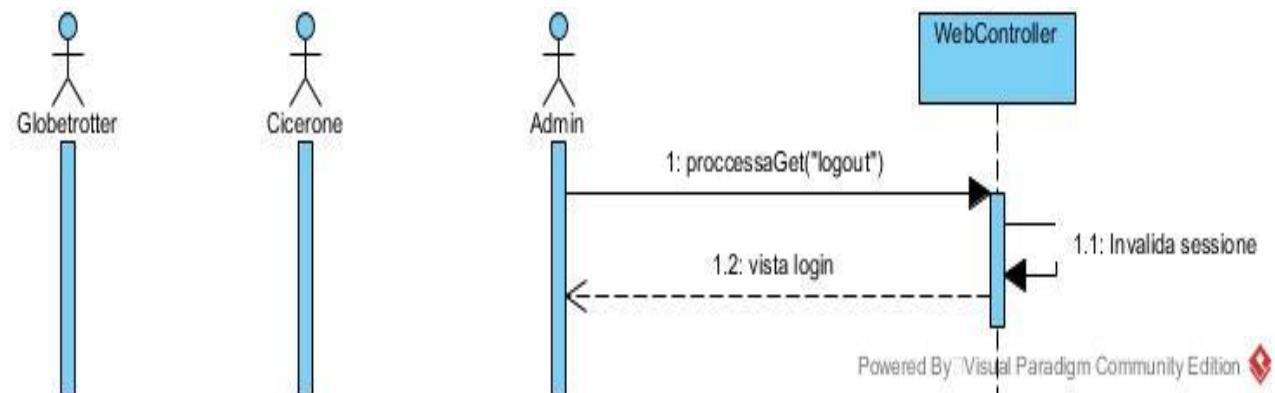
- 
- Visualizzazione piattaforma: Home (DS\_5)



- Cambia ruolo (DS\_6)

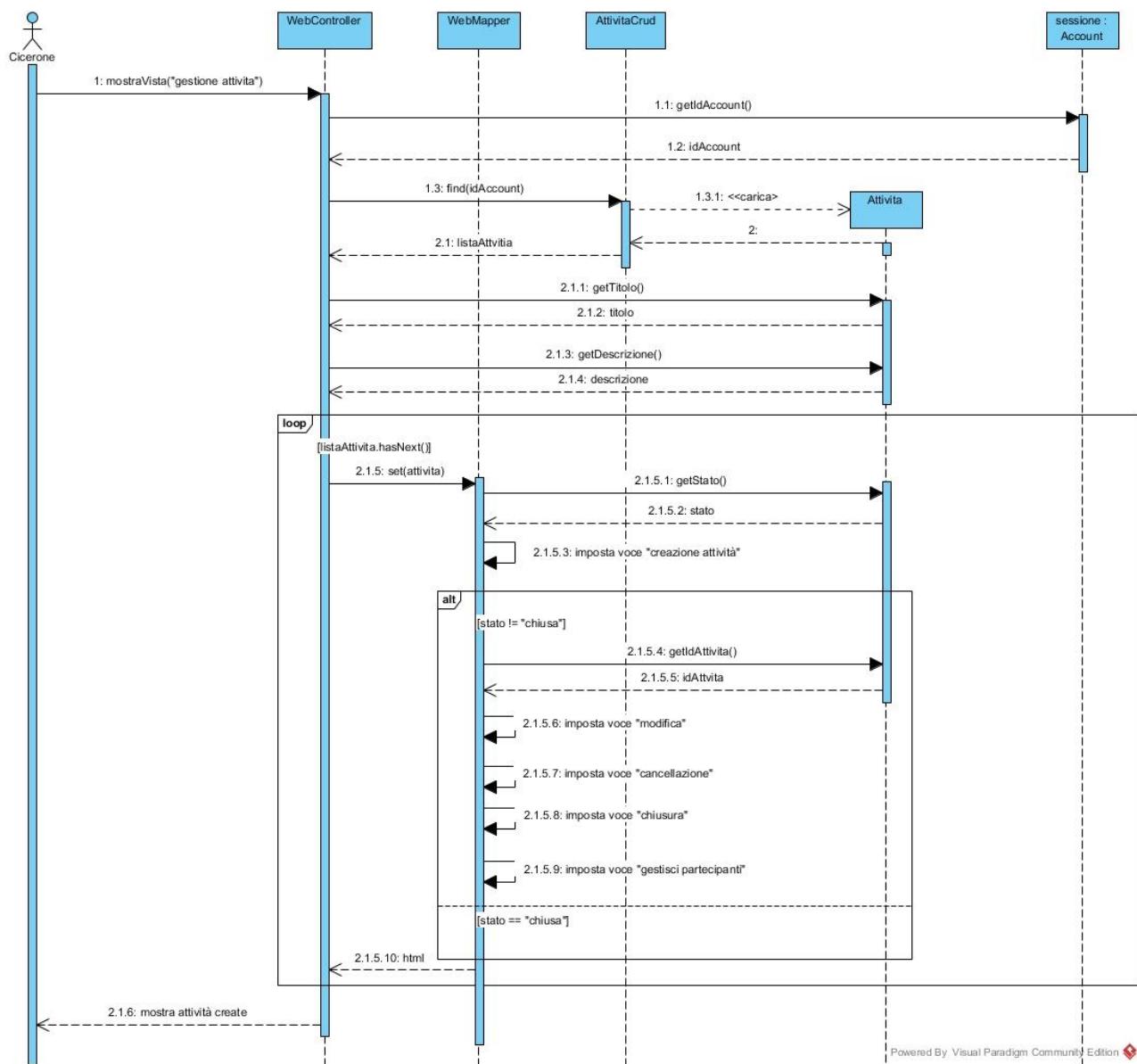


- Effettua logout (DS\_7)

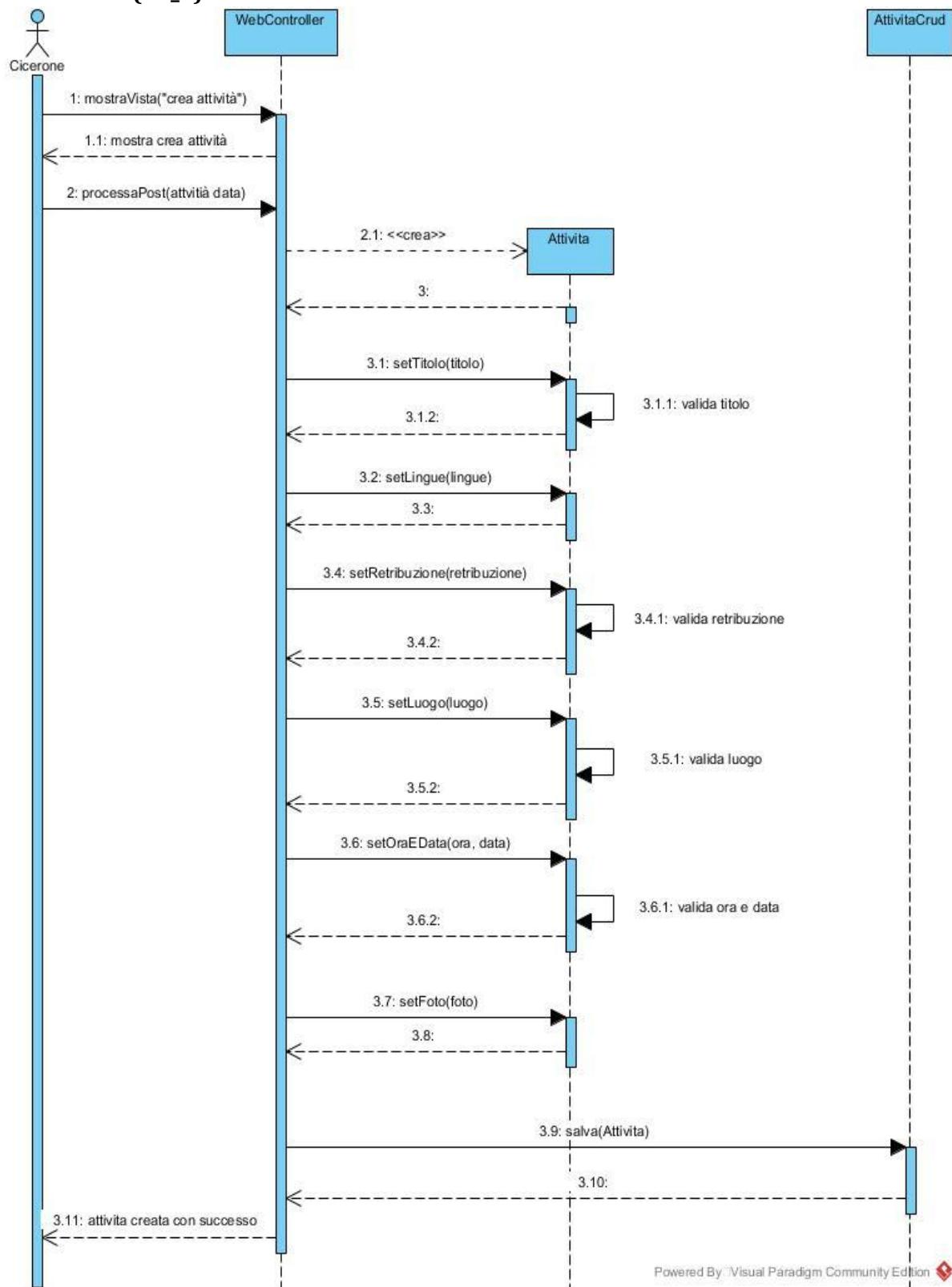




- Visualizza attività create (DS\_8)

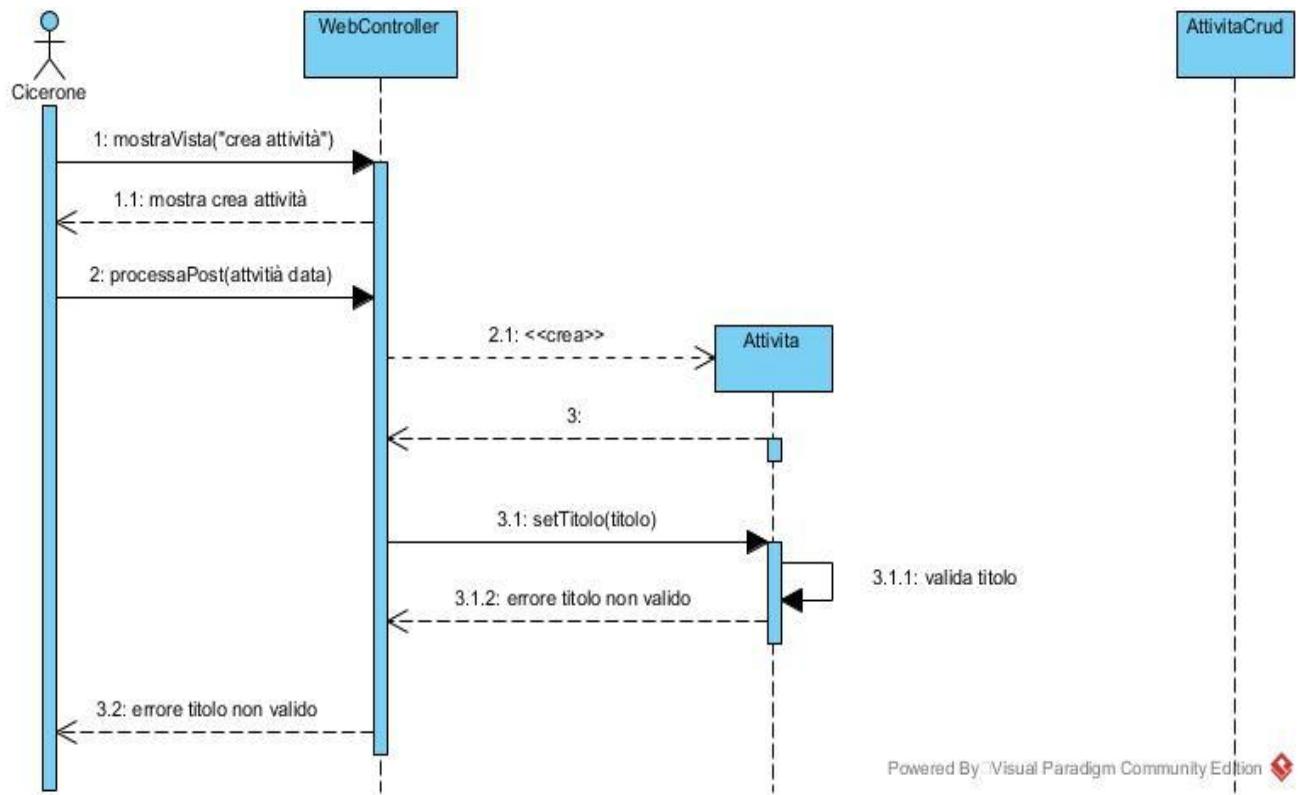


- **Crea attività (DS\_9)**



Powered By Visual Paradigm Community Edition

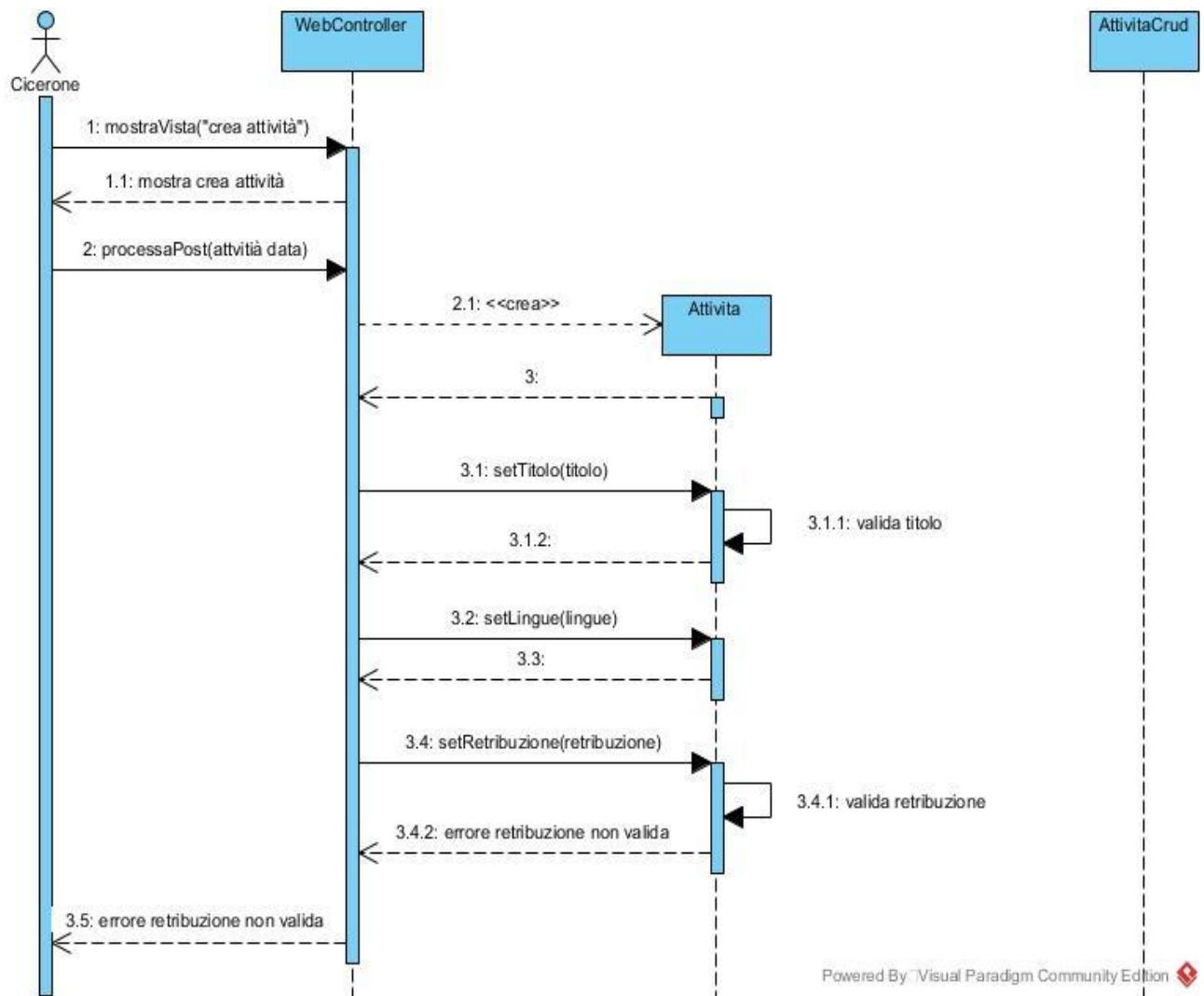
- Crea attività: TitoloNonValido (DS\_9.1)



Powered By  Visual Paradigm Community Edition

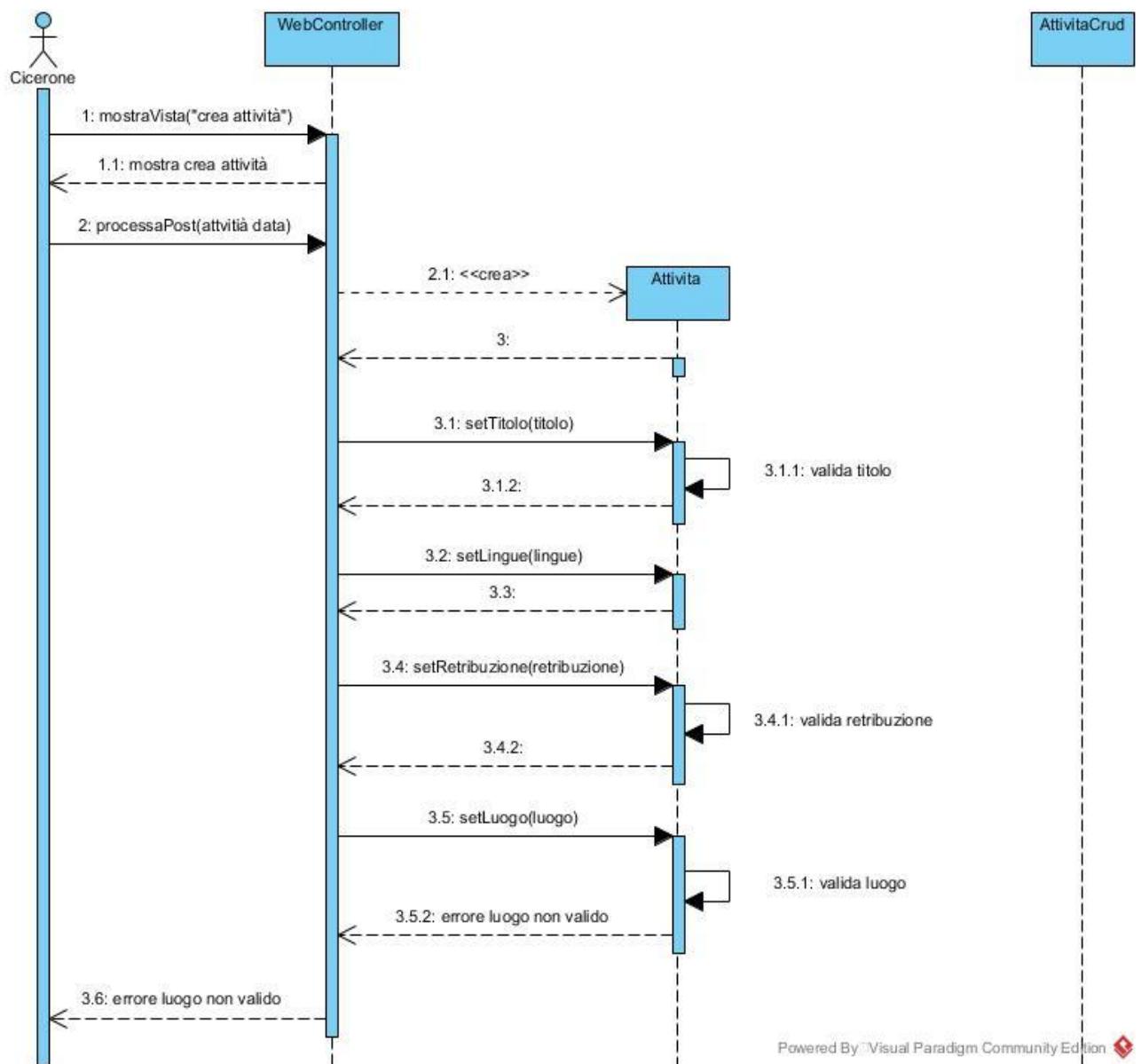


- **Crea attività: RetribuzioneNonValida (DS\_9.2)**

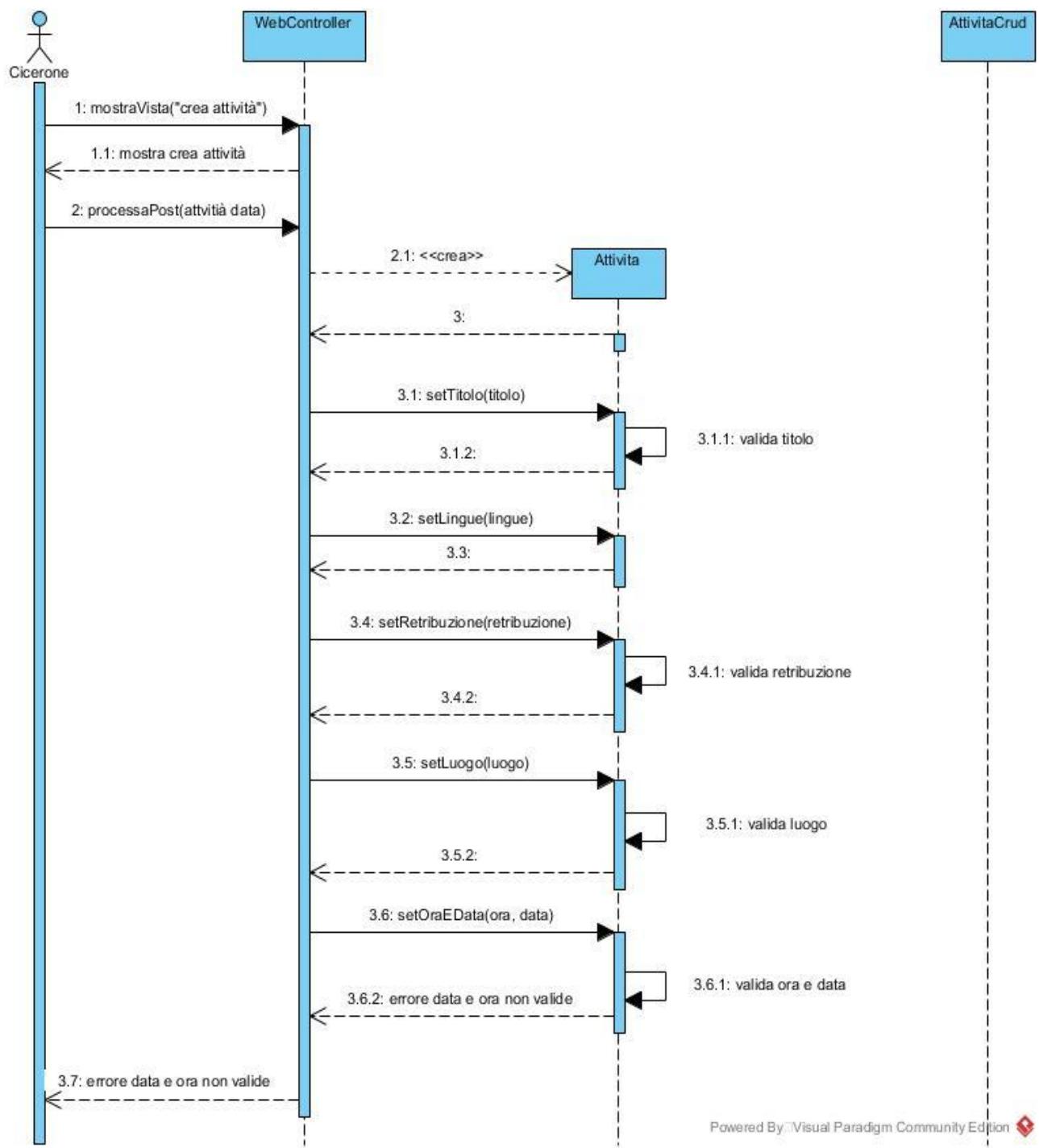




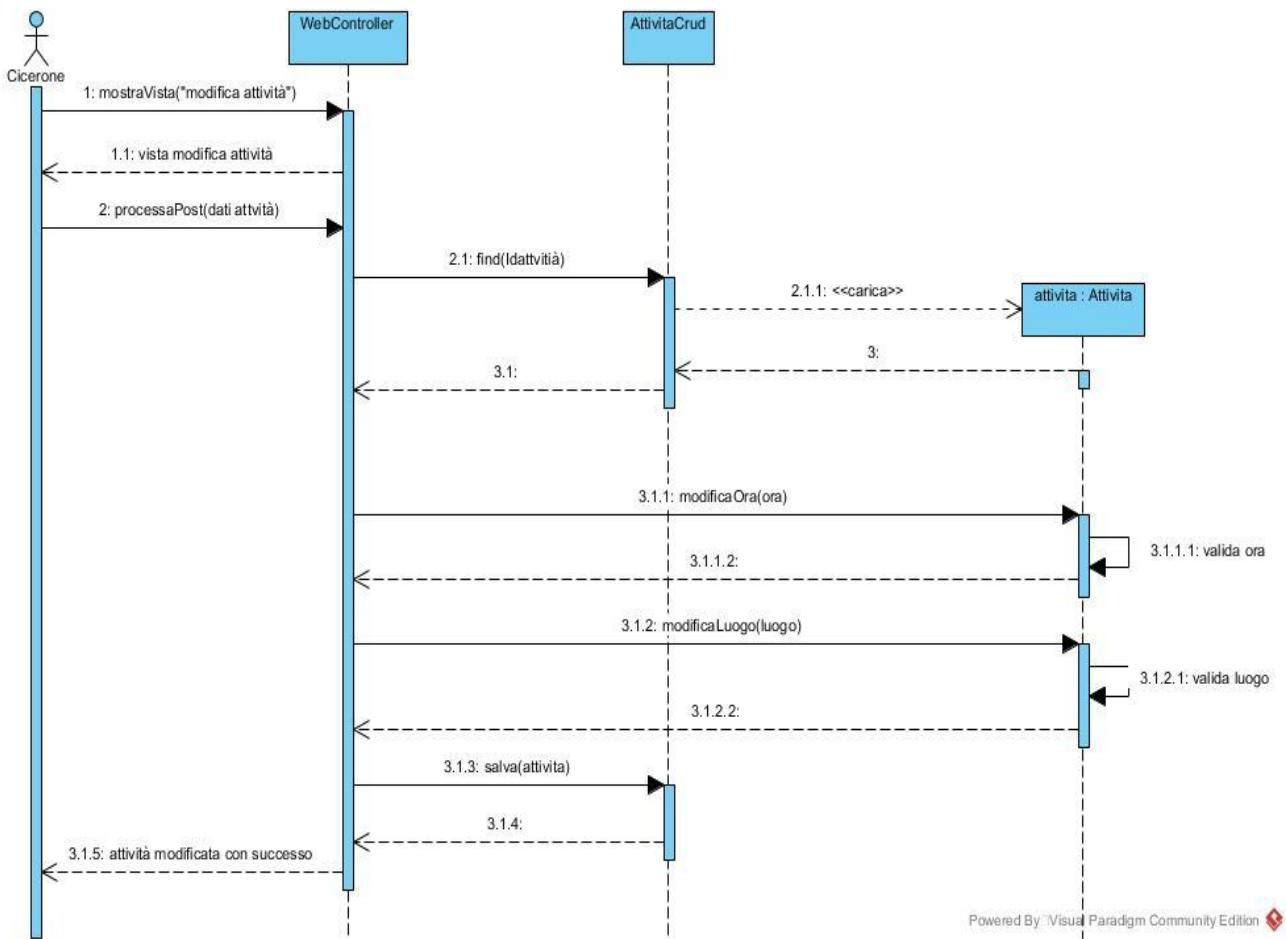
- Crea attività: LuogoNonValido (DS\_9.3)



- Crea attività: DataNonValida (DS\_9.4)

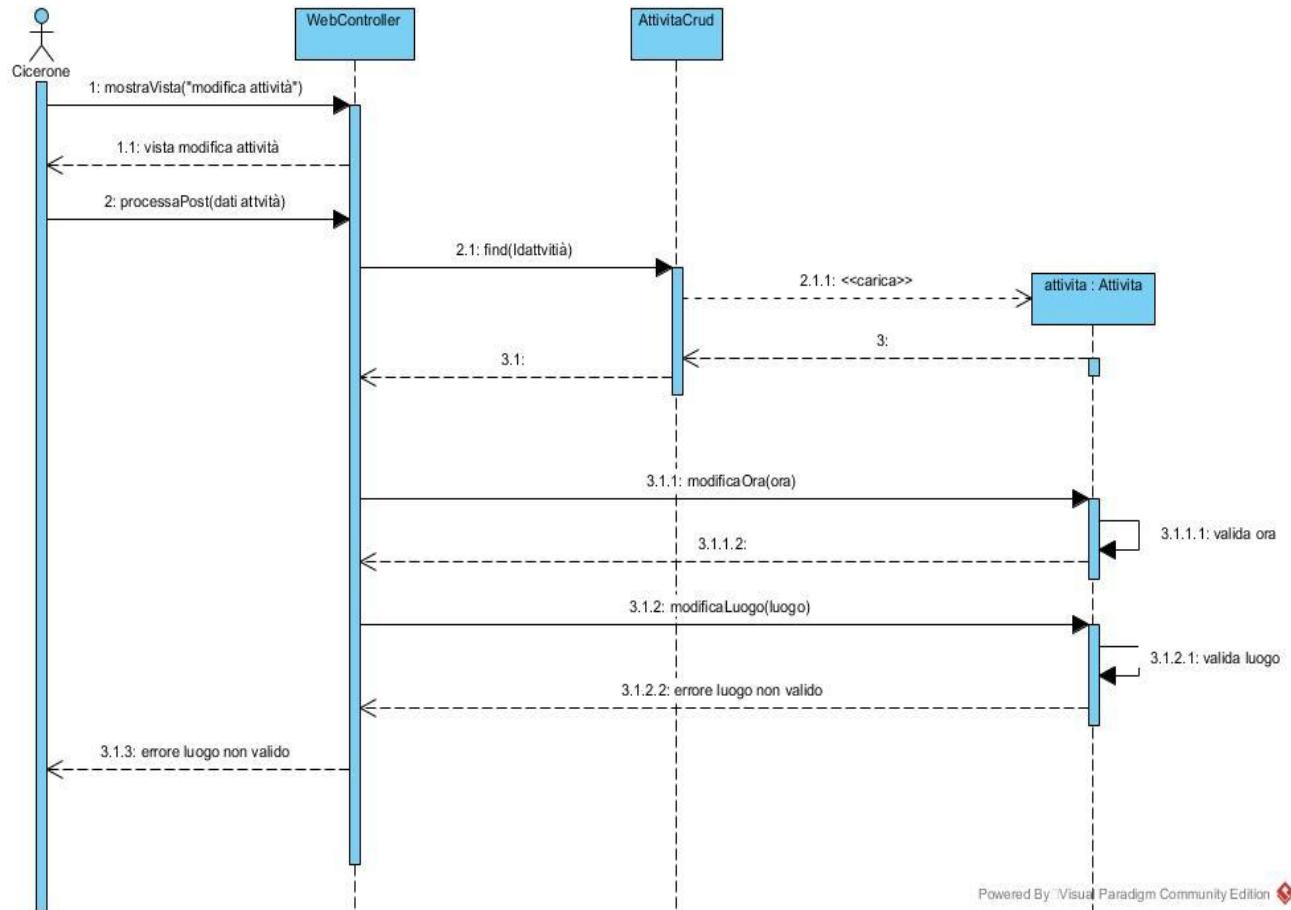


- **Modifica attività (DS\_10)**



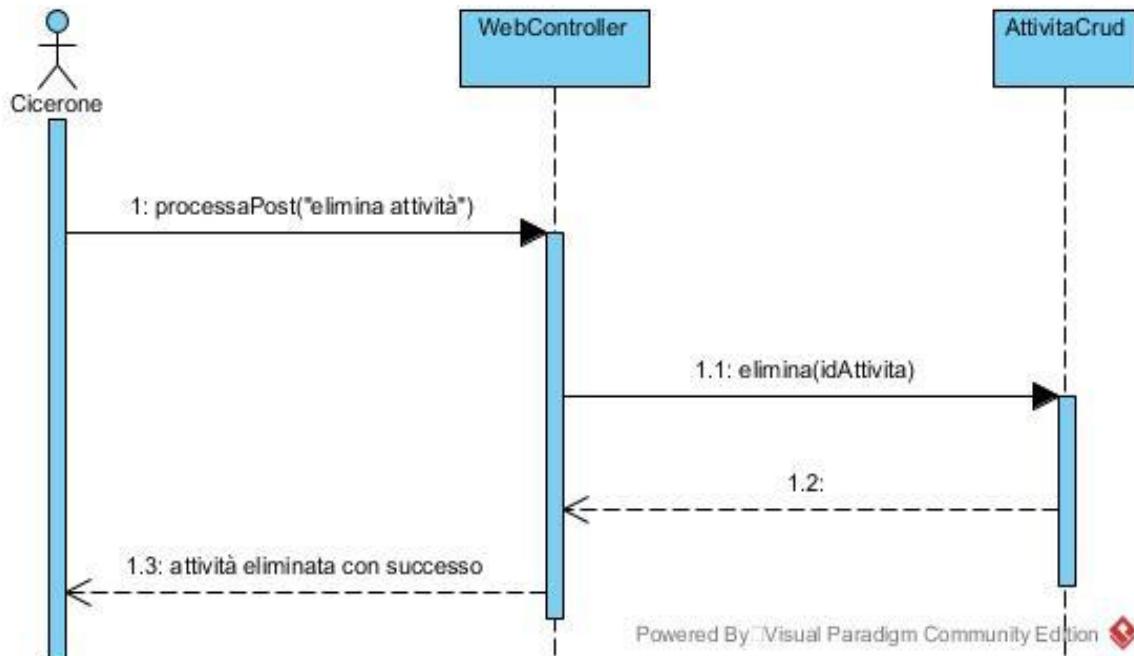
Powered By Visual Paradigm Community Edition

- **Modifica attività: LuogoNonValido (DS\_10.1)**

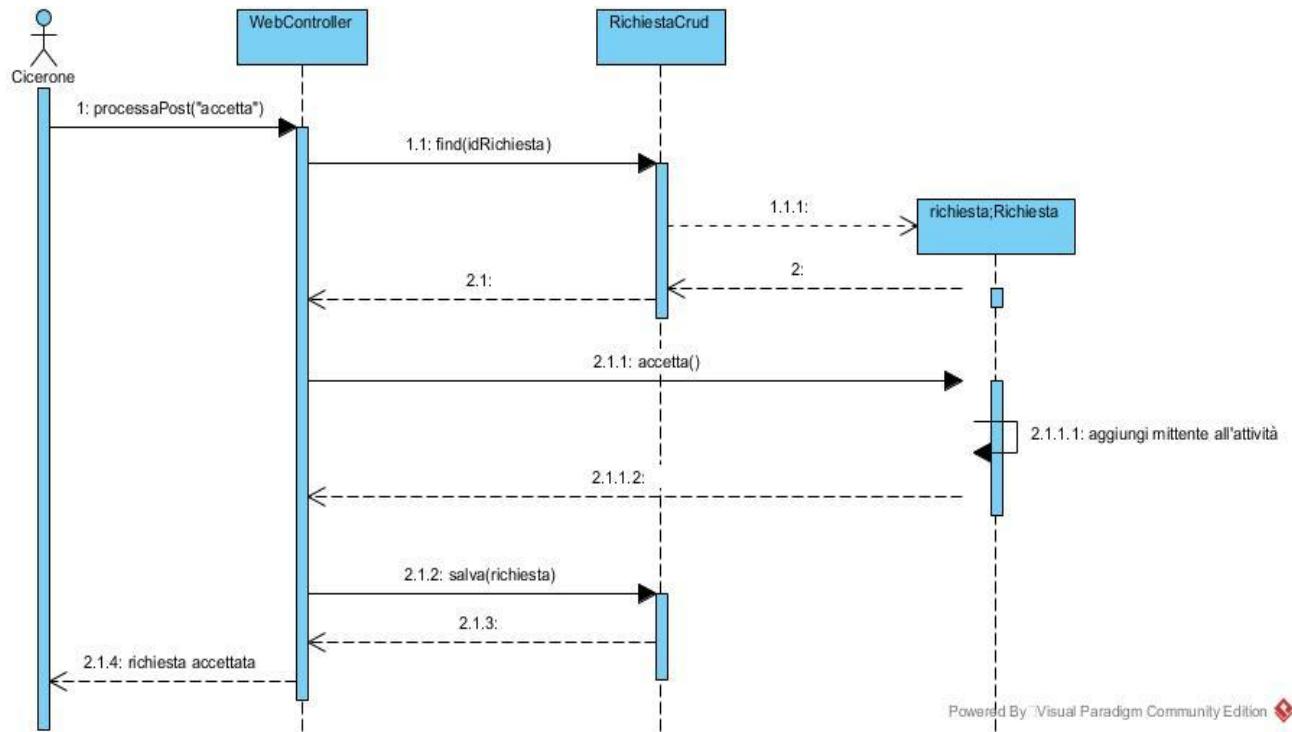


Powered By Visual Paradigm Community Edition

- **Elimina attività (DS\_11)**



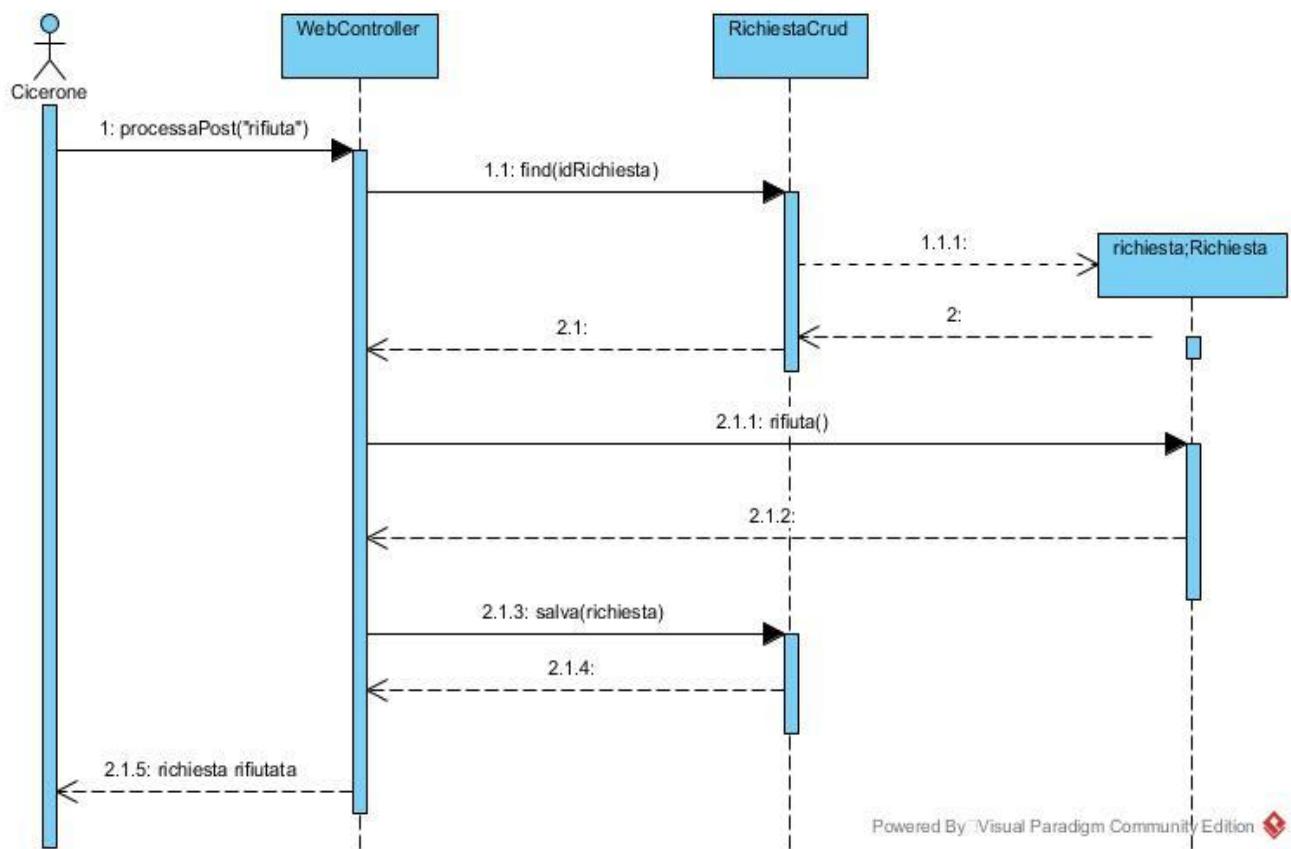
- **Accetta richiesta (DS\_12)**



Powered By Visual Paradigm Community Edition

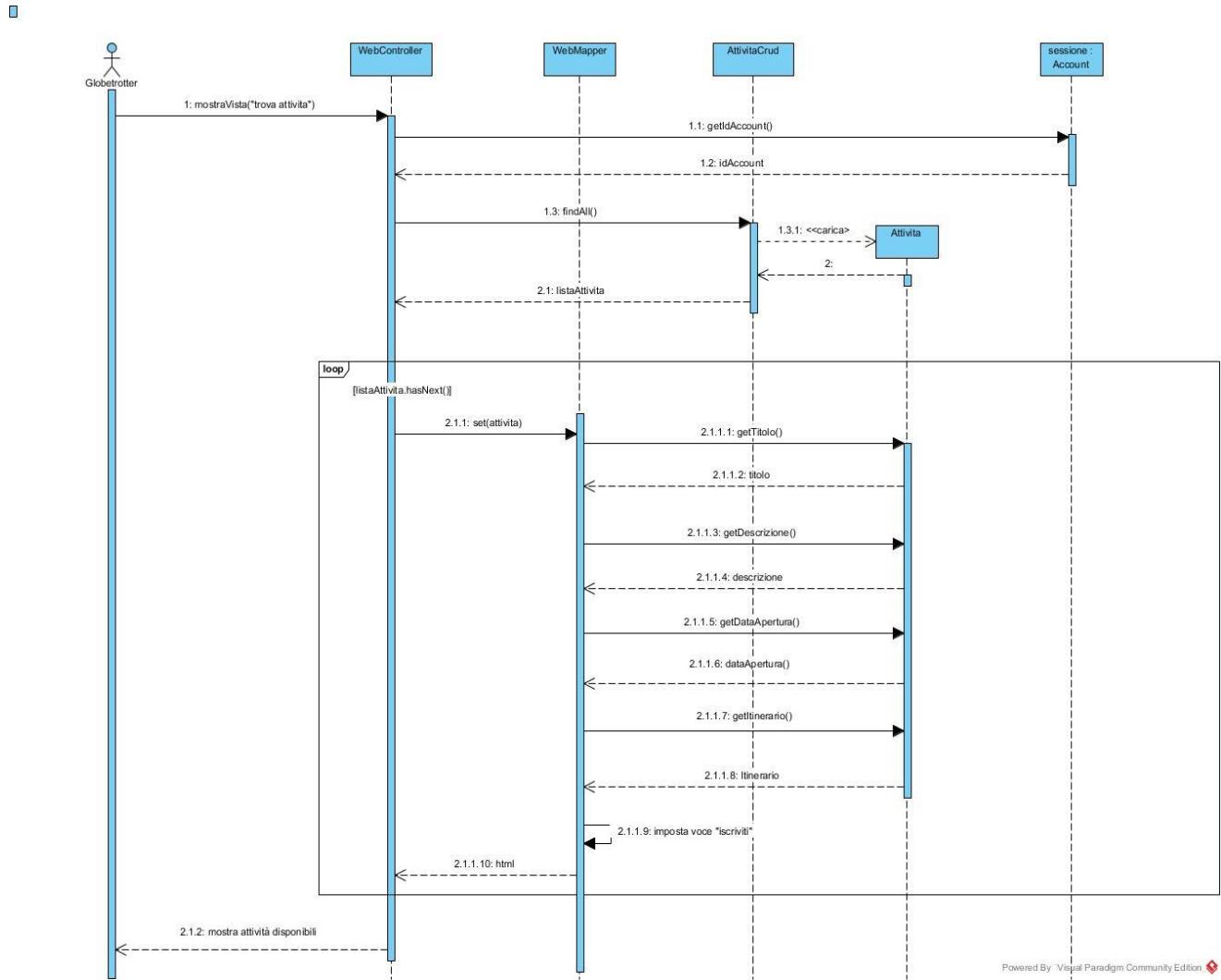


- **Rifiuta richiesta (DS\_13)**



Powered By Visual Paradigm Community Edition

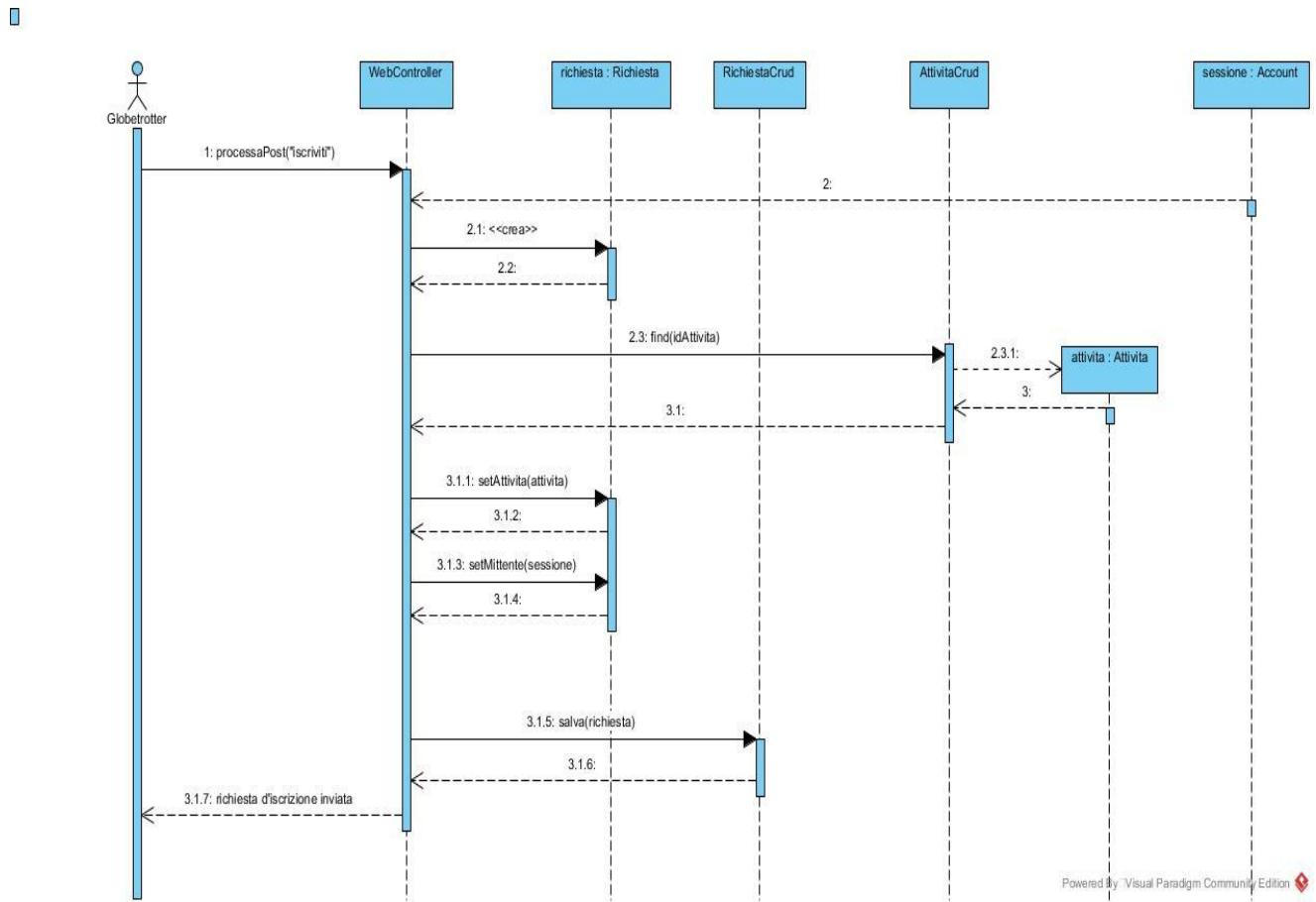
- Visualizza catalogo attività (DS\_14)



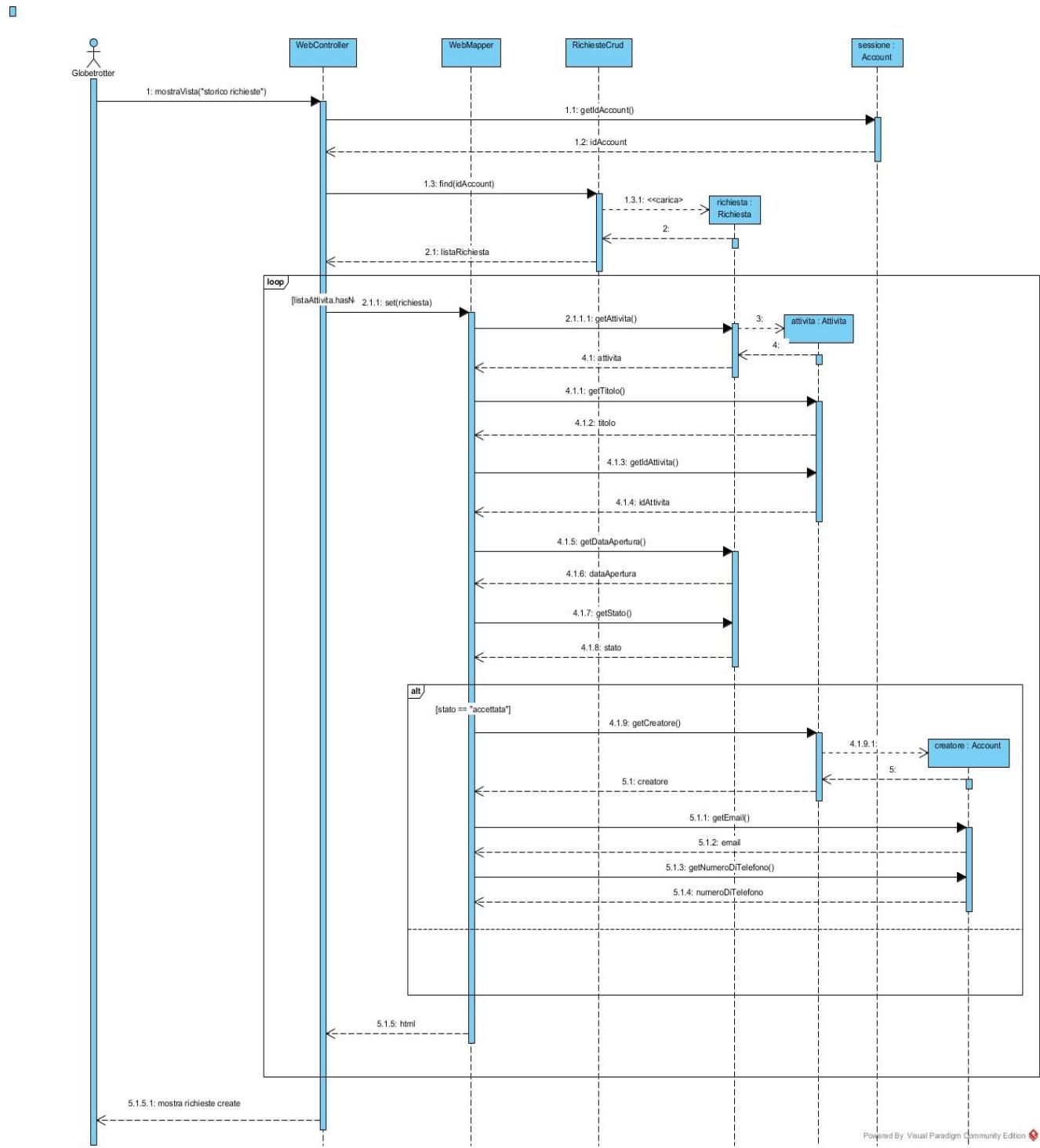
Powered By: Visual Paradigm Community Edition



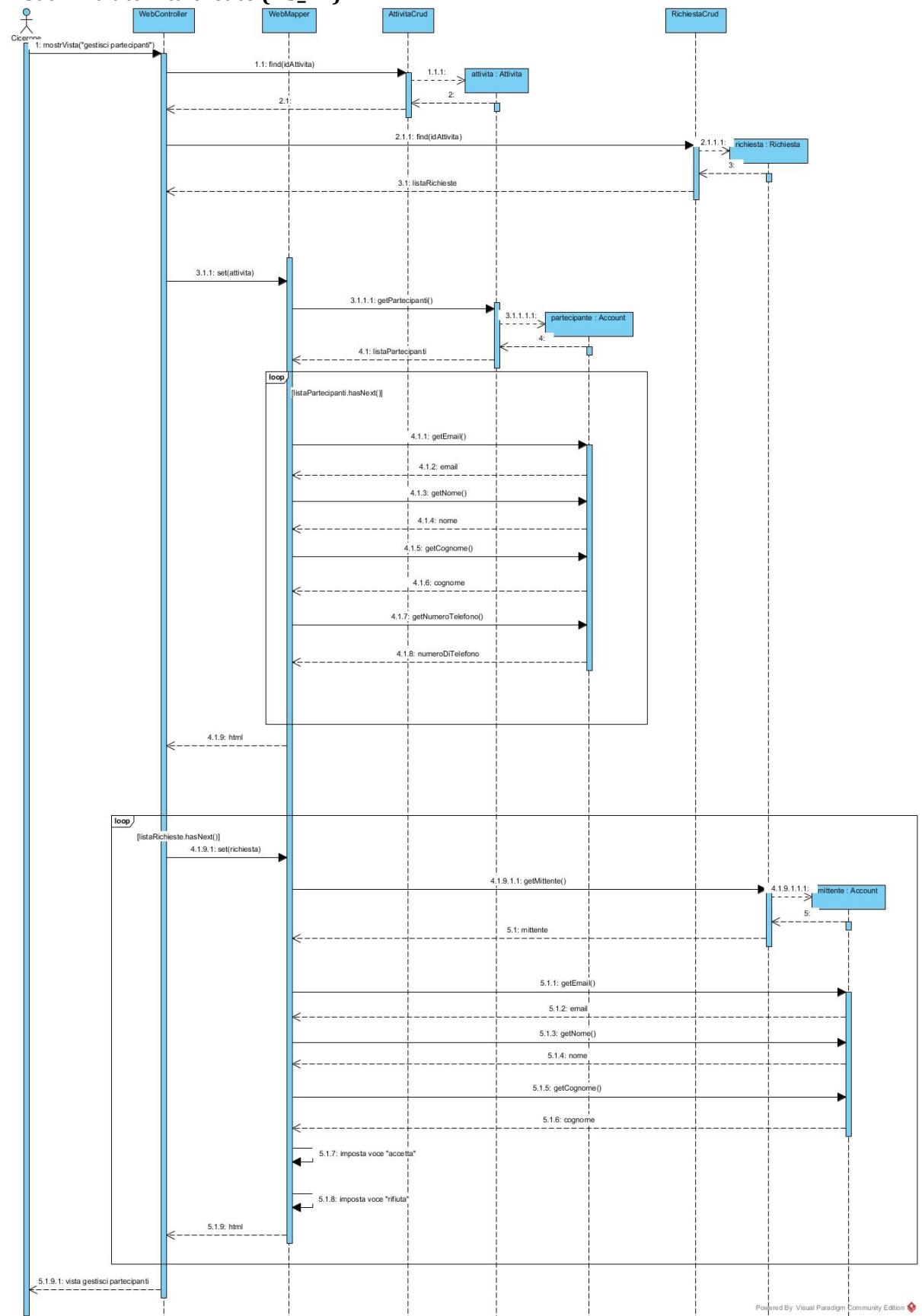
- Crea richiesta (DS\_15)



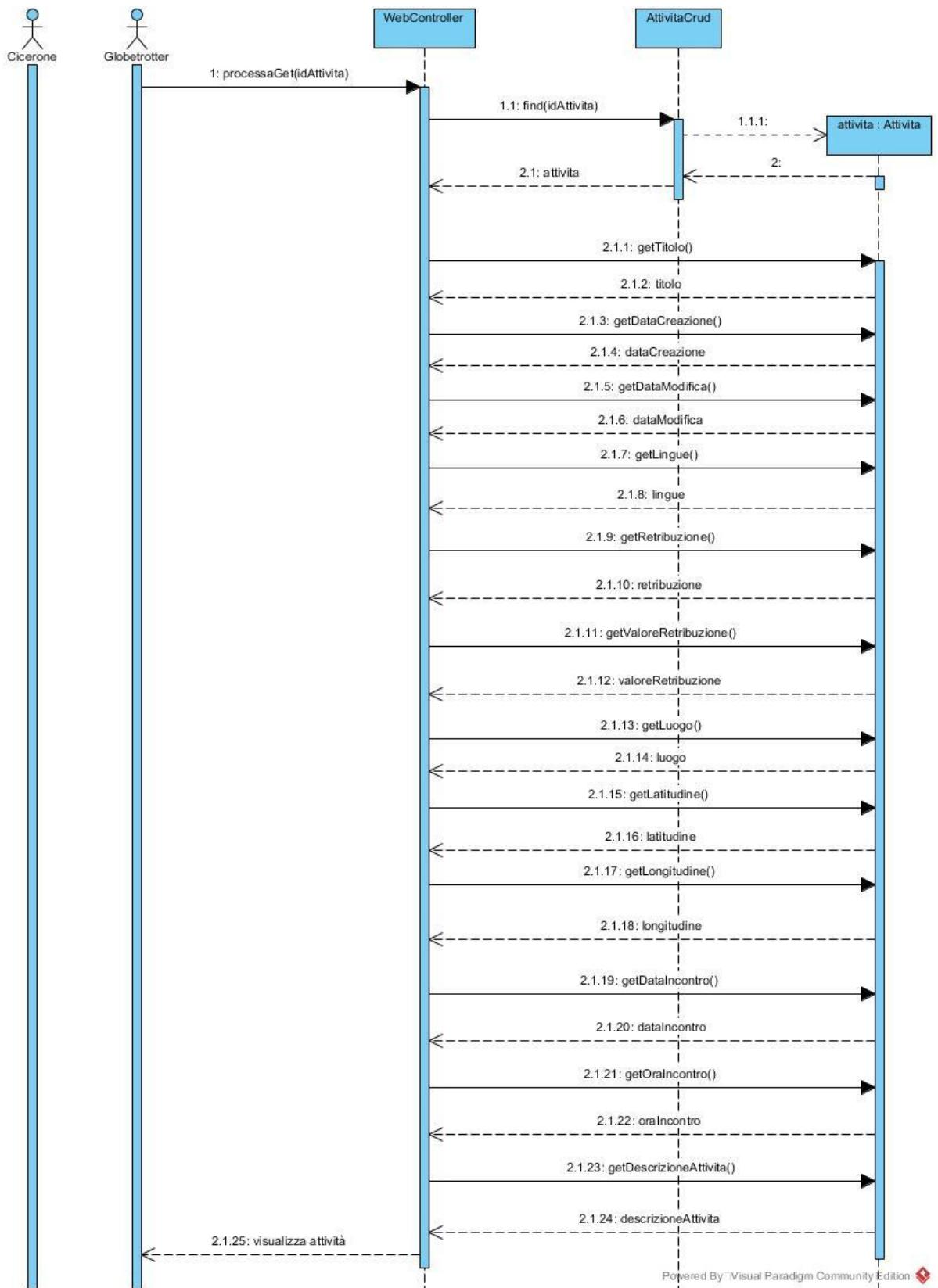
- Visualizza richieste create (DS\_16)



- Visualizza attività create (DS\_17)

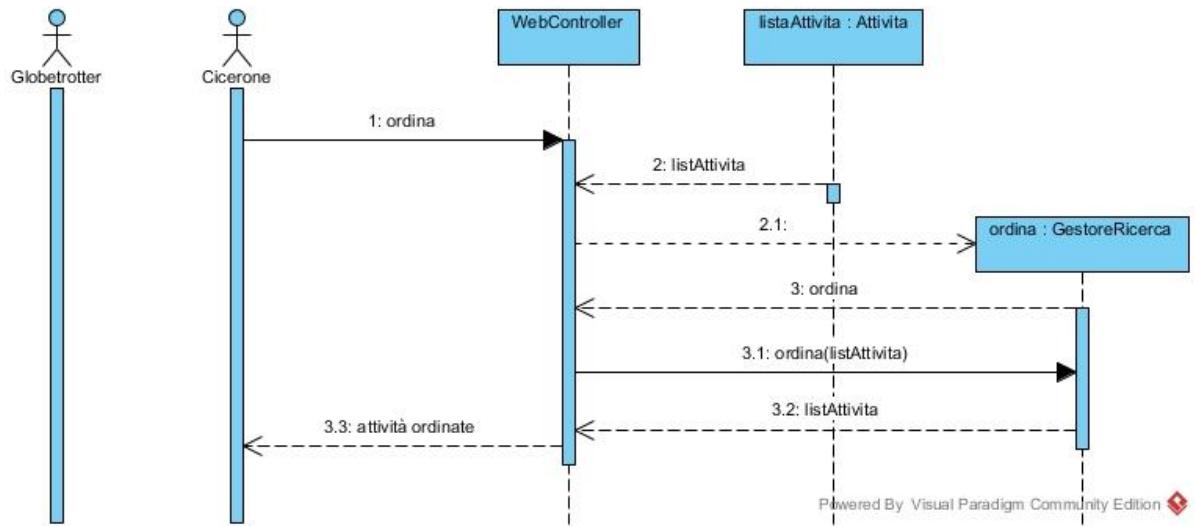


- Visualizza Dettagli (DS\_18)

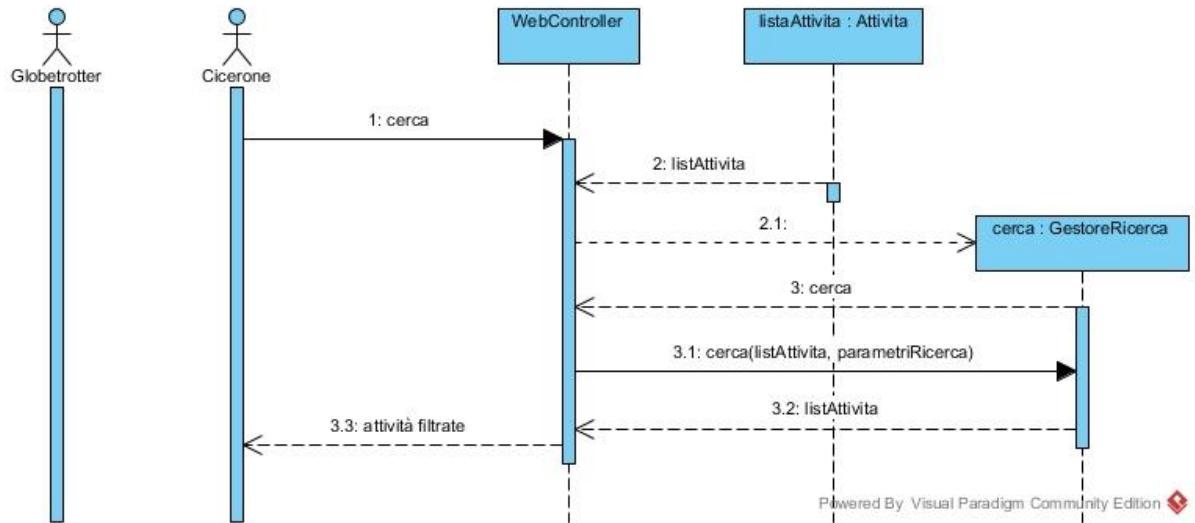




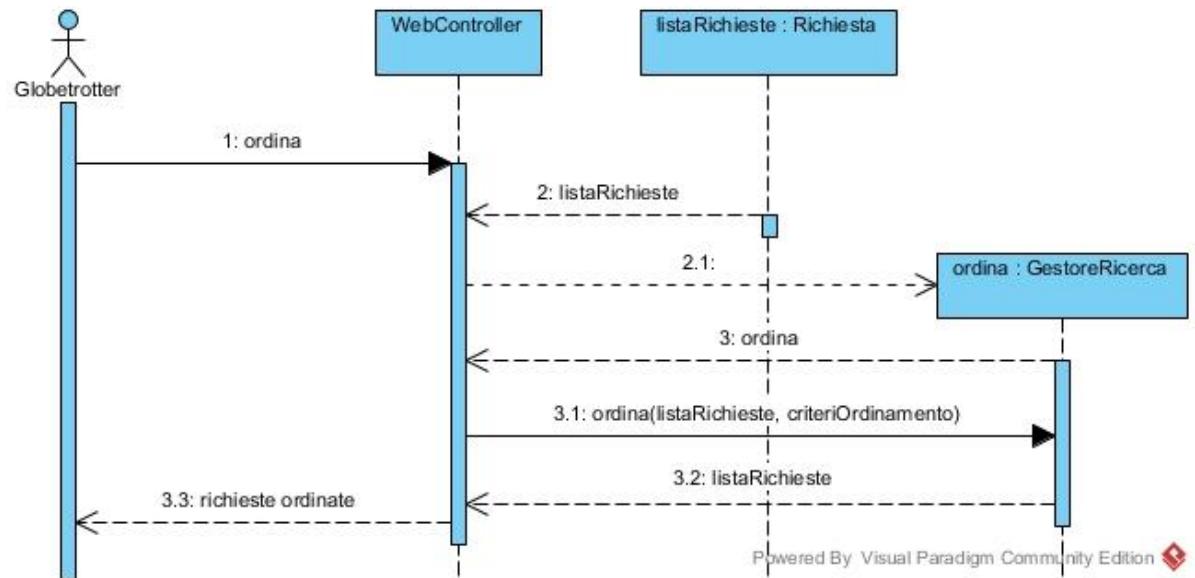
- **Ordina attività (DS\_19)**



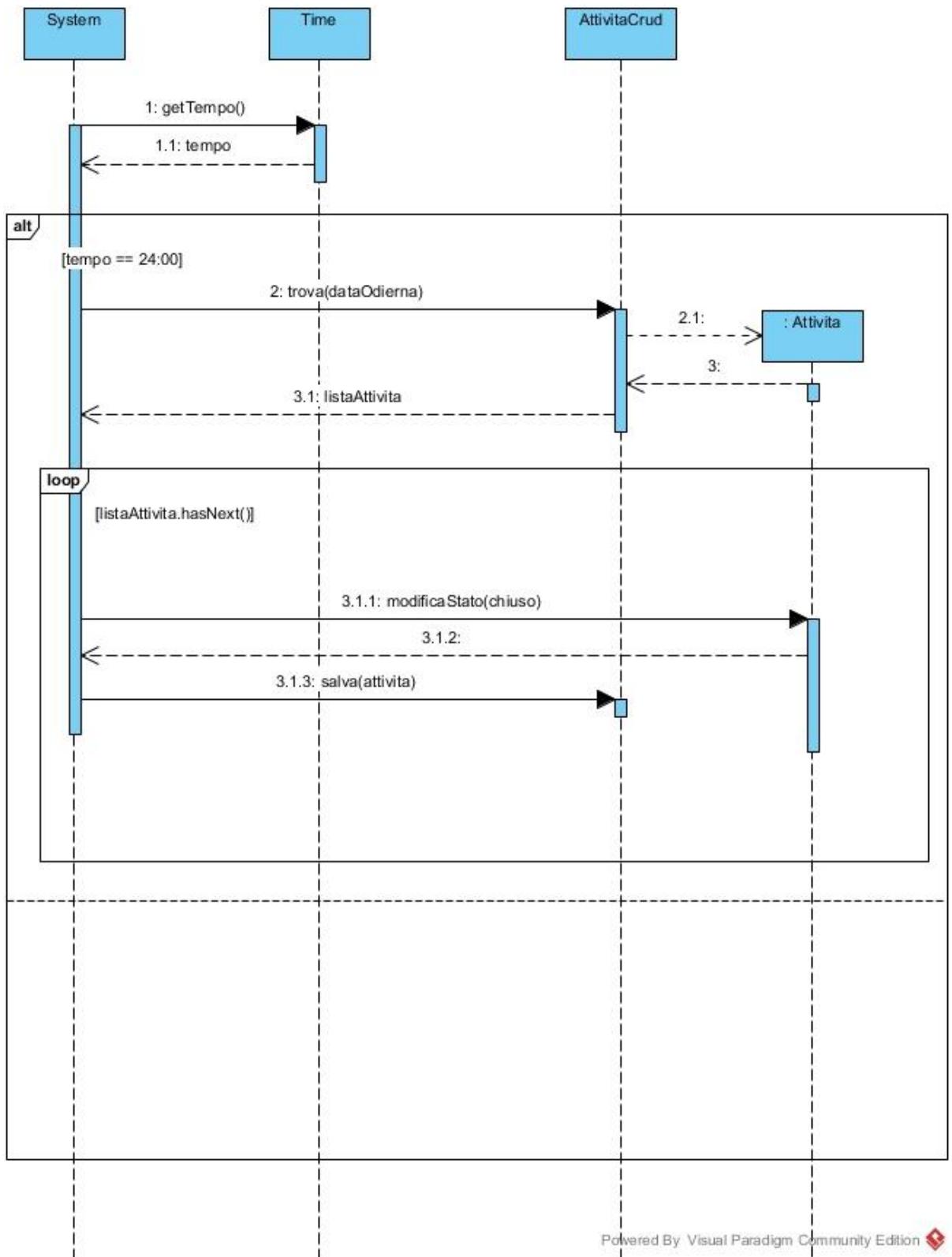
- **Cerca attività (DS\_20)**



- **Ordina richieste (DS\_21)**

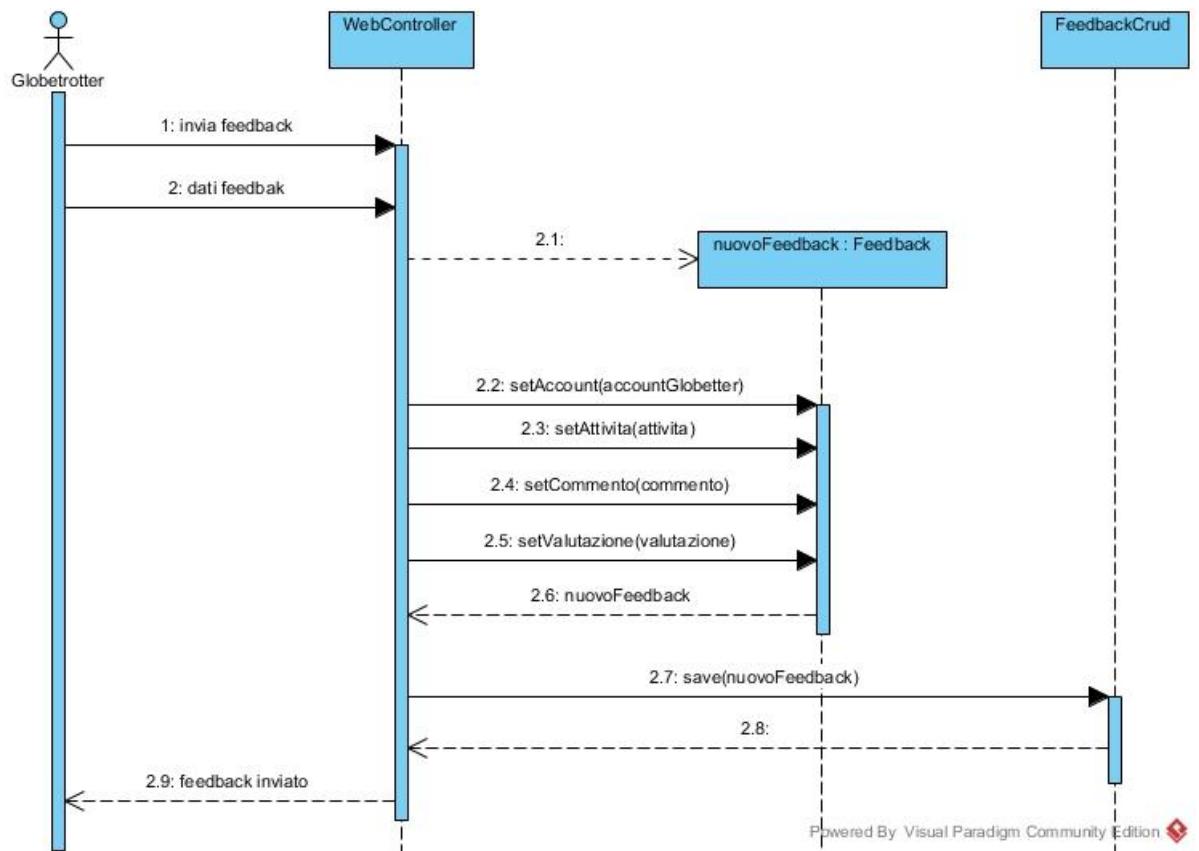


- Chiudi attività (DS\_22)



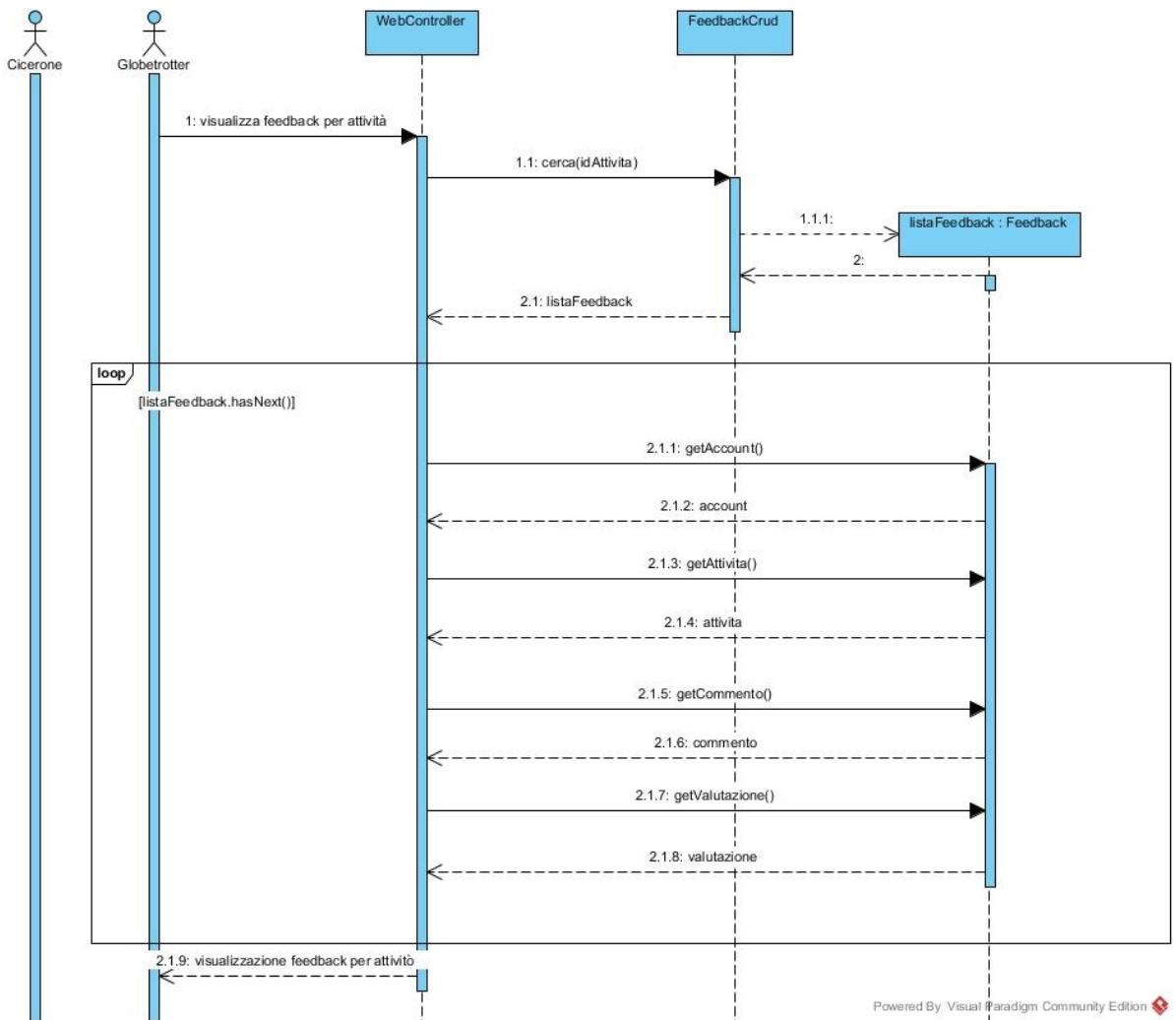


- **Invia feedback (DS\_23)**





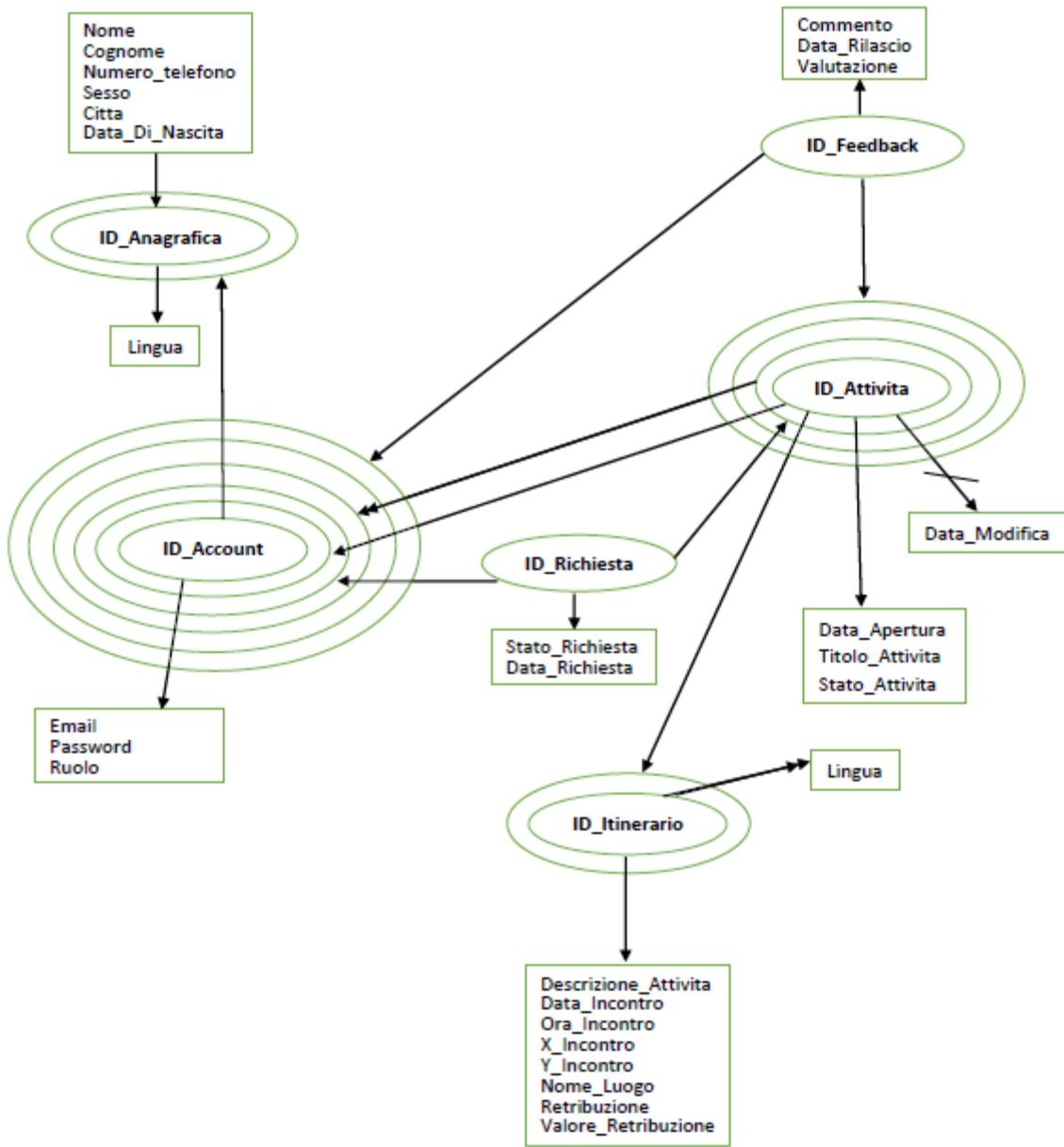
- Visualizza feedback (DS\_24)



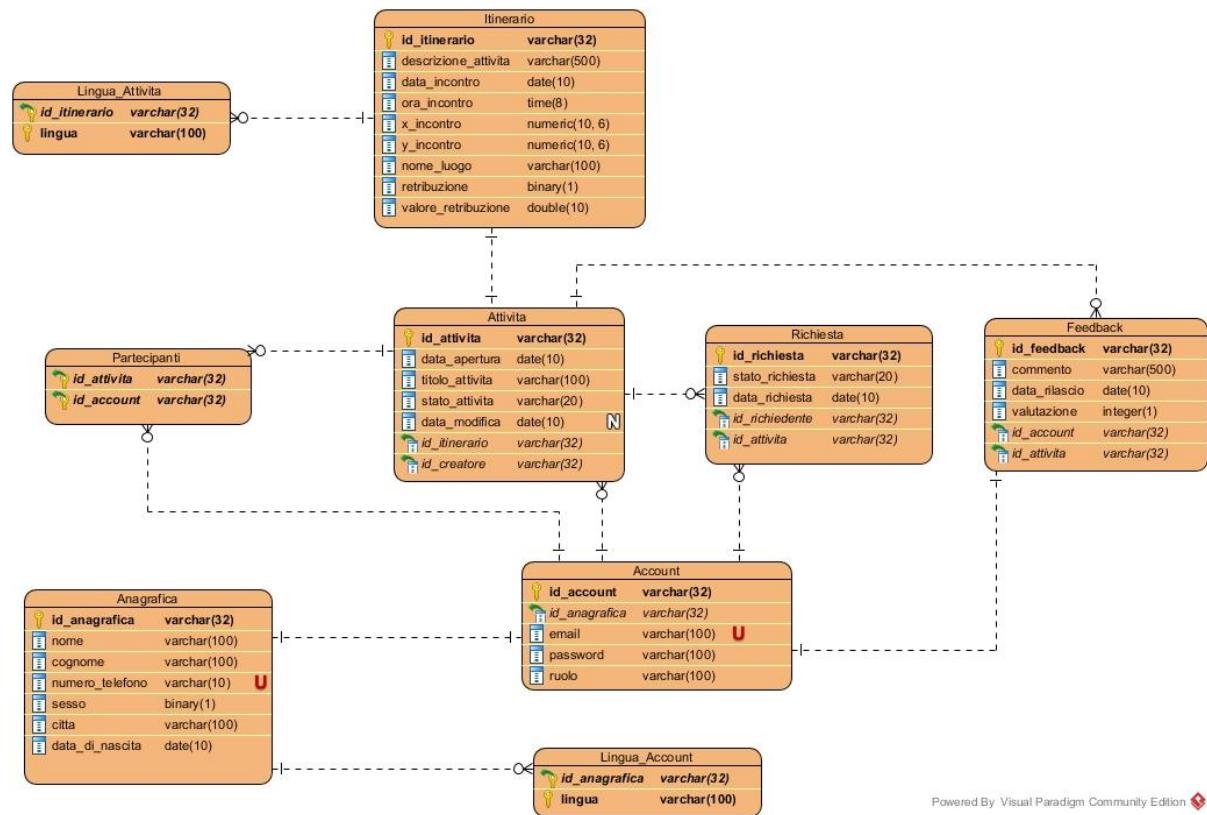
## 4. DATA DESIGN

### 4.1 Database

#### 4.1.1 Diagramma delle Dipendenze dei Dati



## 4.1.2 Modello del Database


Powered By Visual Paradigm Community Edition

## 4.1.3 Lista delle dipendenze e Dettaglio dei Dati

Le entità che avevamo bisogno di rappresentare e quindi di memorizzare in modo persistente sono principalmente sei:

- **Itinerario**: contiene i campi:
  - id\_itinerario: varchar(32) (campo chiave della tabella): rappresenta l'id dell'itinerario aggiunto dal cicerone ad una attività. Anche esso verrà generato dal lato back-end.
  - descrizione\_attivita: varchar(500): contiene la descrizione che un cicerone dà ad una attività, per dare delle info generali al globetrotter che si è iscritto;
  - data\_incontro: date(10): contiene la data in cui il cicerone fissa l'incontro con i globetrotters;

- 
- `x_incontro` e `y_incontro`: numeric(10, 6): rappresentano le coordinate geografiche espresse in decimali (arrotondate a 6 cifre dopo la virgola) del punto di incontro;
  - `nome_luogo`: varchar(100): contiene il nome del luogo individuato dalle coordinate del punto di incontro;
  - `retribuzione`: binary(1): indica se una attività ha una retribuzione per il cicerone che l'ha creata;
  - `valore_retribuzione`: double(10): a completare il concetto introdotto dal campo precedente, serve a contenere il valore in € della retribuzione, se presente;
  - **Feedback:** contiene i campi:
    - `id_feedback`: varchar(32): generato dal lato back-end, contiene l'id del feedback rilasciato da un globetrotter ad una attività ed al cicerone che l'ha creata;
    - `commento`: varchar(500): rappresenta il commento (max 500 caratteri) che il globetrotter vuole lasciare assieme alla valutazione;
    - `data_rilascio`: date(10): è la data in cui il feedback viene rilasciato;
    - `valutazione`: integer(1): un numero intero da 1 a 5 per la valutazione all'attività e al cicerone;
    - `id_account`: varchar(32): è il riferimento che contiene l'id dell'account (globetrotter) che ha lasciato il feedback;
    - `id_attivita`: varchar(32): è il riferimento che contiene l'id dell'attività a cui è stato lasciato un feedback;
  - **Attivita:** contiene i campi:
    - `id_attivita`: varchar(32) (campo chiave della tabella): rappresenta l'id dell'attività creata da un cicerone. Anche esso verrà generato dal lato back-end;
    - `data_apertura`: date(10): contiene la data in cui una attività viene creata dal cicerone;
    - `titolo_attivita`: varchar(100): contiene il titolo che il cicerone sceglie di dare ad una attività da lui creata;
    - `stato_attivita`: varchar(20): contiene lo stato di una attività fra 3 disponibili: Aperta, Modificata e Chiusa;
    - `data_modifica`: date(10) (nullable): rappresenta la data in cui una attività viene modificata in qualche suo campo dal cicerone;

- 
- id\_itinerario: varchar(32): è il riferimento all'id dell'itinerario che contiene;
  - id\_creatore\_ varchar(32): è il riferimento all'id dell'account (cicerone) che ha creato l'attività;
  - **Richiesta:** contiene i campi:
    - id\_richiesta: varchar(32) (campo chiave della tabella): anche esso generato dal lato back-end, contiene l'id della richiesta fatta da un globetrotter per partecipare ad una attività;
    - stato\_richiesta: varchar(20): contiene lo stato della richiesta fra 3 disponibili: Accettata, In sospeso e Rifiutata;
    - data\_richiesta: date(10): contiene la data in cui il globetrotter ha fatto richiesta di partecipazione ad una attività;
    - id\_richiedente: varchar(32): è il riferimento che contiene l'id dell'account (globetrotter) che ha effettuato la richiesta;
    - id\_attività: varchar(32): è il riferimento che contiene l'id dell'attività a cui è stata mandata la richiesta;
  - **Account:** contiene i campi:
    - id\_account: varchar(32) (campo chiave della tabella): generato dal back-end, contiene l'id di un account registrato nel sistema;
    - id\_anagrafica: varchar(32): è il riferimento che contiene l'anagrafica relativa all'account iscritto;
    - email: varchar(32) (unique): contiene l'email che l'account ha usato per iscriversi alla piattaforma;
    - numero\_notifiche: integer(100): è un contatore di notifiche che memorizza quante notifiche ha ricevuto un account;
    - password: varchar(100): contiene la password che l'account ha usato per iscriversi e che usa per effettuare il login;
    - ruolo: varchar(100): contiene il ruolo dell'account aggiornato all'ultimo accesso;
  - **Anagrafica:** contiene i campi:
    - id\_anagrafica: varchar(32) (campo chiave della tabella): generato dal lato back-end, contiene l'id dell'anagrafica;
    - nome: varchar(100): contiene il nome che l'utente ha immesso durante la registrazione alla piattaforma;
    - cognome: varchar(100): contiene il nome che l'utente ha immesso durante la registrazione alla piattaforma;

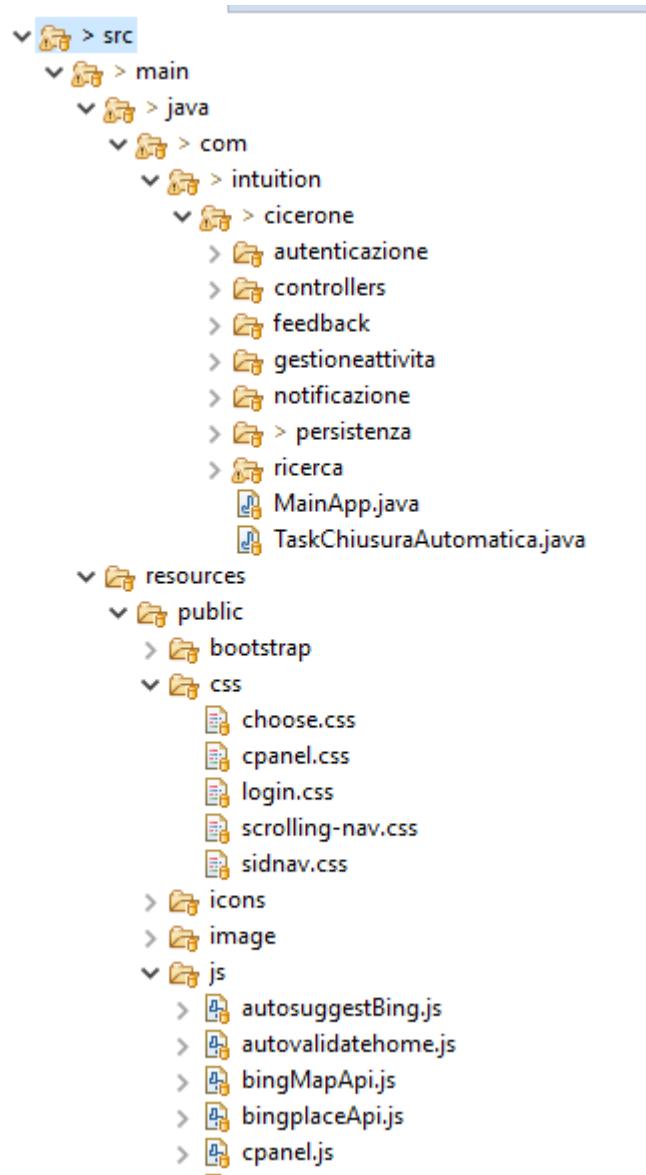
- 
- numero\_telefono: varchar(10) (unique): contiene il nome che l'utente ha immesso durante la registrazione alla piattaforma;
  - sesso: binary(1): contiene l'informazione sul genere che l'utente ha scelto durante la registrazione;
  - citta: varchar(100): contiene la città che l'utente ha inserito durante la registrazione;
  - data\_di\_nascita: date(10): contiene la data di nascita dell'utente, usata per la registrazione;

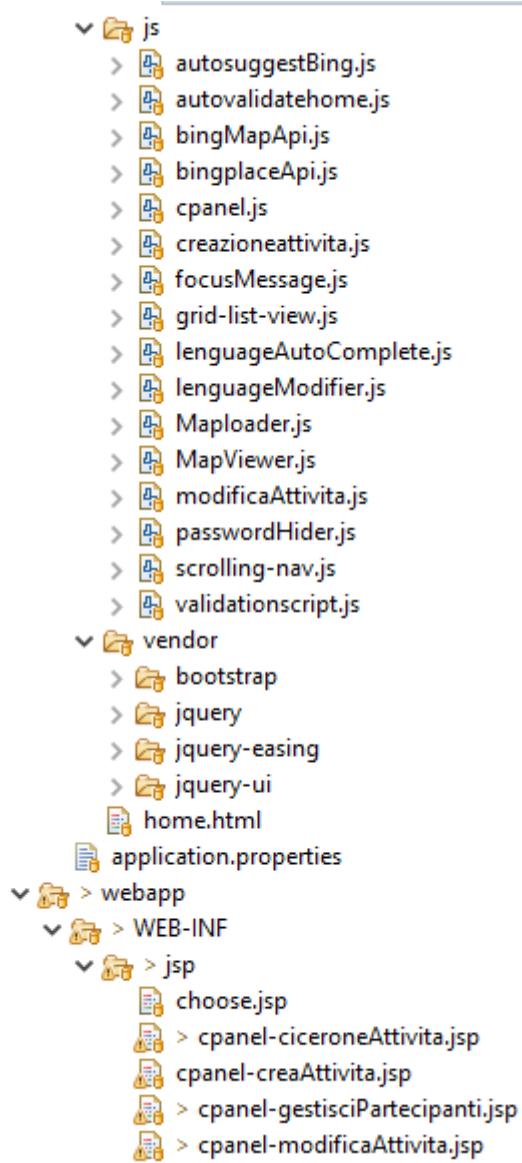
Successivamente abbiamo avuto la necessità di introdurre altre tabelle di supporto per realizzare il sistema e sono:

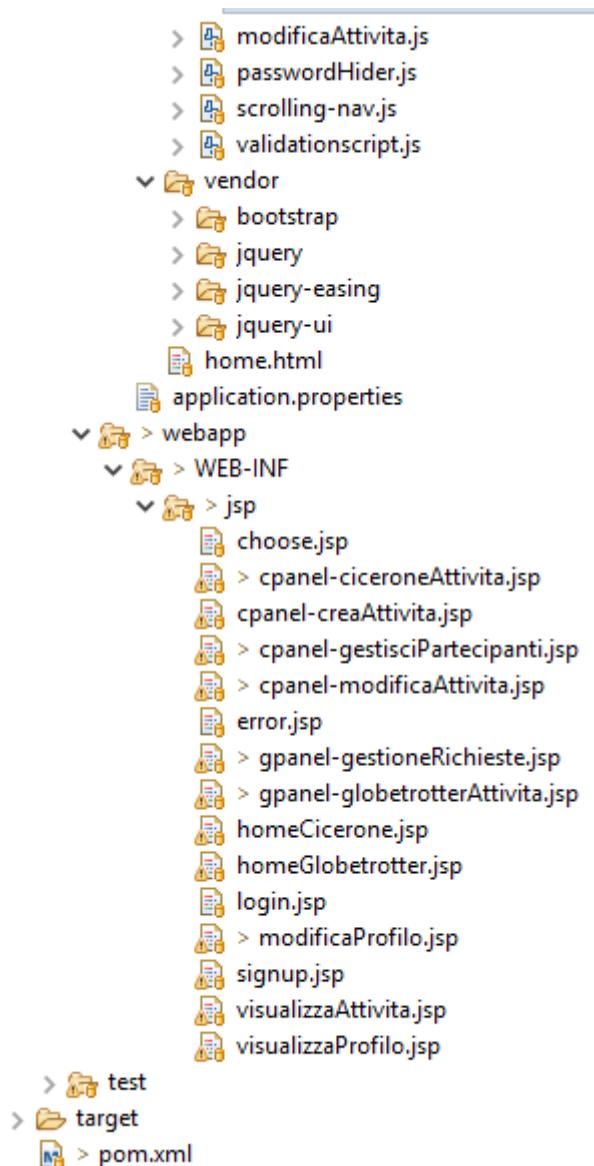
- Partecipanti;
- Lingua\_Account;
- Lingua\_Attività.



## 4.2 File System









## 5. GLOSSARIO

### 5.1 Acronimi

- **CRUD:** operazioni sui dati del database di create, retrieve, update e delete;
- **UI:** User Interface;
- **MVC:** Model–View–Controller;
- **DTO:** Data Transfer Object;
- **JSP:** Java Server Page;
- **UUID:** Universal Unique Identifier;
- **HTML:** Hyper–Text Mark–up Language;

### 5.2 Definizioni

- **Framework:** software progettato per supportare lo sviluppo di siti web dinamici, applicazioni web e servizi web. Il suo scopo è quello di alleggerire il lavoro associato allo sviluppo delle attività più comuni di un'applicazione web da parte dello sviluppatore;
- **Hash:** l'hash è una funzione che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza minore;
- **Salt:** il salt è una stringa generata casualmente e aggiunta ad una password;
- **Middleware:** insieme di software che fungono da intermediari fra strutture e programmi informatici, permettendo loro di comunicare a dispetto della diversità dei protocolli o dei sistemi operativi;
- **Mapping:** termine usato per indicare la conversione dei dati;
- **Utility Class:** classe conosciuta anche come Helper class, è una classe che contiene solo metodi statici e non può essere istanziata. Contiene, inoltre, una serie di metodi correlati, in modo che possano essere riutilizzati;
- **Wrapper Class:** una classe wrapper non è altro che una classe “involucro” che ha come unico scopo quello di contenere un valore primitivo rendendolo da un lato un oggetto e dall'altro “ornandolo” con metodi che altrimenti non avrebbero una loro naturale collocazione;
- **Binding:** termine usato per indicare la conversione dei dati;

- 
- **Servlet Class:** termine usato per indicare il collegamento tra due o più elementi;
  - **Tupla:** elemento di un database relazionale caratterizzato da uno o più attributi;
  - **Hibernate:** Hibernate è una piattaforma middleware open source per lo sviluppo di applicazioni Java, attraverso l'appoggio al relativo framework, che permette di gestire la persistenza dei dati sul database attraverso la rappresentazione e il mantenimento su database relazionale di un sistemi di oggetti Java.