



Università Politecnica delle Marche
Facoltà di Ingegneria
Corso di Laurea Triennale in Ingegneria Informatica e
dell'Automazione

Progetto per l'esame di ingegneria del software 2021/2022
Gestione di una campagna di Dungeons & Dragons

Indice



1. Descrizione
 - 1.1. Glossario di progetto
2. Gestione dei requisiti
 - 2.1. Analisi dei requisiti
 - 2.1.1. Requisiti funzionali
 - 2.1.2. Requisiti non funzionali
 - 2.2. Casi d'uso
 - 2.2.1. Gestione Login
 - 2.2.2. Miscellanea
 - 2.2.3. Gestione Giocatore
 - 2.2.4. Gestione Master
 - 2.3. Matrice di Mapping
3. Analisi
 - 3.1. Classi di analisi
 - 3.2. Diagrammi di sequenza
 - 3.2.1. AggiornaPersonaggi
 - 3.2.2. CUDAppunti
 - 3.2.3. Login
 - 3.2.4. TrovaAppunti
 - 3.2.5. TrovaPersonaggi
 - 3.3. Diagrammi di attività
 - 3.3.1. Aggiorna Personaggi
 - 3.3.2. CUD Appunti
 - 3.3.3. Login
 - 3.3.4. Trova Appunti
 - 3.3.5. Trova Personaggi
4. Progettazione
 - 4.1. Classi di Progettazione
 - 4.2. Diagrammi delle macchine a stati
 - 4.2.1. Appunti
 - 4.2.2. Dado
 - 4.2.3. Scheda
 - 4.2.4. Utente
 - 4.3. Diagramma dei componenti
5. Implementazione
 - 5.1. Diagramma di deployment
 - 5.2. Mockup
6. Test
 - 6.1. PyUnit

Descrizione

Il progetto consiste nella realizzazione di un'applicazione per la gestione di una campagna del gioco da tavolo Dungeons and Dragons.



L'applicativo consentirà di caricare e gestire tutte quelle informazioni necessarie ai Giocatori e al cosiddetto Master per lo svolgimento della campagna.

In particolare ogni singolo giocatore potrà salvare e gestire una scheda contenente: la storia del personaggio, la classe, la razza, i punteggi di caratteristica, abilità, armatura, punti ferita e livello.

Il Master invece avrà la possibilità di visionare e modificare tutte le informazioni associate ai singoli giocatori oltre a poter salvare e gestire: mappa, trama, informazioni sui luoghi e informazioni sugli NPC.

Ogni giocatore dovrà dapprima creare una scheda del Personaggio con tutte le informazioni necessarie per la campagna.

Il giocatore dovrà scegliere tra le classi presenti: Guerriero, Barbaro, Ranger, Paladino, Chierico, Ladro, Monaco, Druido, Mago, Bardo, Stregone e Warlock.

Inoltre il giocatore dovrà determinare i propri punteggi di caratteristica, che sono Forza, Destrezza, Costituzione, Intelligenza, Saggezza e Carisma.

In base alle scelte di classe e punteggi di caratteristica, saranno poi determinati i vari modificatori associati ai tiri di salvezza e alle abilità, per quest'ultime scegliendo in quali avrà competenza.

Il giocatore potrà aggiornare o modificare queste informazioni in ogni momento in base agli sviluppi del proprio personaggio.

Oltre a tutto, il giocatore potrà scriversi delle note personali in uno spazio apposito (quindi non nella scheda), per tenere traccia dello svolgimento della campagna, salvarsi da parte delle informazioni che gli sono utili o per esigenze simili.

Il Master, oltre a poter visionare e modificare anch'egli le informazioni determinate dai giocatori, potrà aggiungere documenti che contengono informazioni utili alla trama della campagna. Essi potranno essere dei file testuali contenenti informazioni su alcune città o insediamenti del mondo di gioco o riguardanti NPC controllati dal Master stesso, ma anche file immagine come mappe di alcuni luoghi o dell'intero mondo di gioco. A sua discrezione, il Master potrà scegliere di mostrare alcuni di questi appunti ai giocatori, raggruppati nelle dispense, che saranno disponibili ai giocatori in qualsiasi momento.

Il programma sarà inoltre in grado di fornire statistiche su diversi aspetti che riguardano i giocatori, come il livello medio, i punti ferita medi, e altre statistiche di nota per il Master riguardanti i giocatori.

Sia il Master che i giocatori hanno la possibilità di lanciare dei dadi attraverso il programma, di cui si può decidere, a seconda dell'esigenza, il tipo di dado (determinato dal numero delle sue facce) e la quantità di dadi tirati.

Vi sono 6 tipi di dadi diversi: un tetraedro, un cubo, un ottaedro, un dado da 10 facce, un dodecaedro e un icosaedro.

La campagna si divide in più sessioni, le quali si svolgono in giornate diverse. Il Master avrà accesso ad un contatore del numero di sessioni effettuate che potrà andare ad aumentare per tenere traccia del tempo trascorso dall'inizio della campagna. Questo lo potrà fare una volta che le informazioni saranno state aggiornate alla sessione corrente sia da lui che dai giocatori.



Glossario

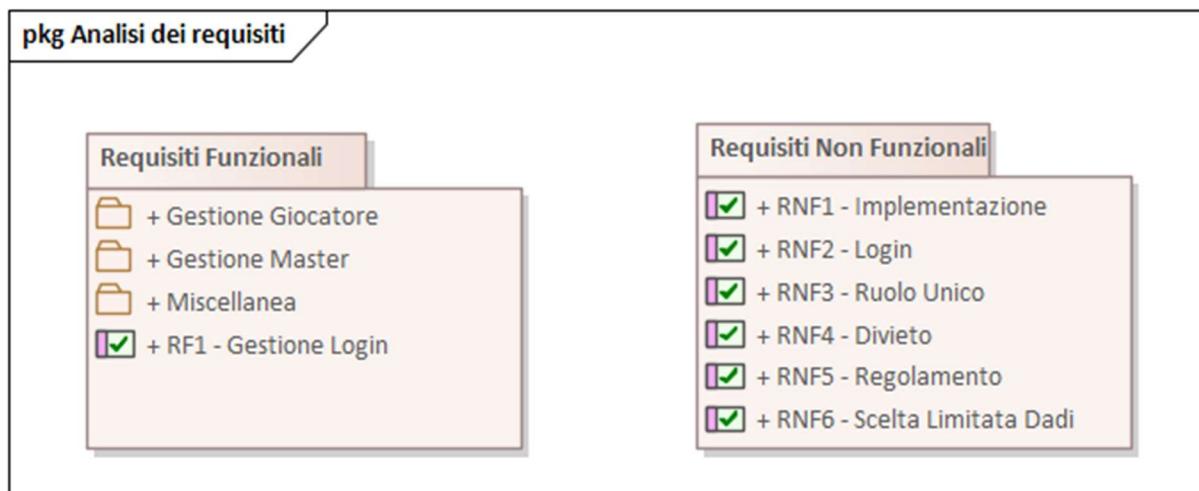


Vocaboli	Descrizione	Sinonimi	Omonimi	Tipo
Master	Amministratore della campagna e del programma	DM	nessuno	Business
Giocatore	Partecipante alla campagna	nessuno	nessuno	Business
Account giocatore	Il profilo utente del giocatore utilizzato per interagire col software	nessuno	nessuno	Tecnico
Account master	Il profilo utente del master utilizzato per interagire col software	nessuno	nessuno	Tecnico
Personaggio	L'avatar con cui un Giocatore partecipa alla campagna	PG	nessuno	Business
Scheda	L'insieme di tutti i dati di un Personaggio	nessuno	nessuno	Business
Appunti	L'insieme di mappe e informazioni testuali a disposizione del Master riguardo la campagna	nessuno	nessuno	Business
Note	Tutto ciò che eventualmente un giocatore vuole scriversi e tenere da parte	nessuno	Appunti	Business
Dispense	La parte degli appunti visualizzabili dai giocatori	nessuno	nessuno	Business
NPC	Sono i personaggi non giocanti	Non-Player Character	nessuno	Business
Contatore	Strumento a disposizione del master per segnare il numero di sessioni svolte	nessuno	nessuno	Tecnico
Sessione	Una singola giocata della campagna	nessuno	nessuno	Business
Dungeons & Dragons	Il gioco da tavolo che il programma tratta	D&D, DnD	nessuno	Business
Campagna	La macro-partita che i Giocatori cercheranno di portare a termine	nessuno	nessuno	Business
Dado	Ci sono 6 tipi di dado diversi.	nessuno	nessuno	Business

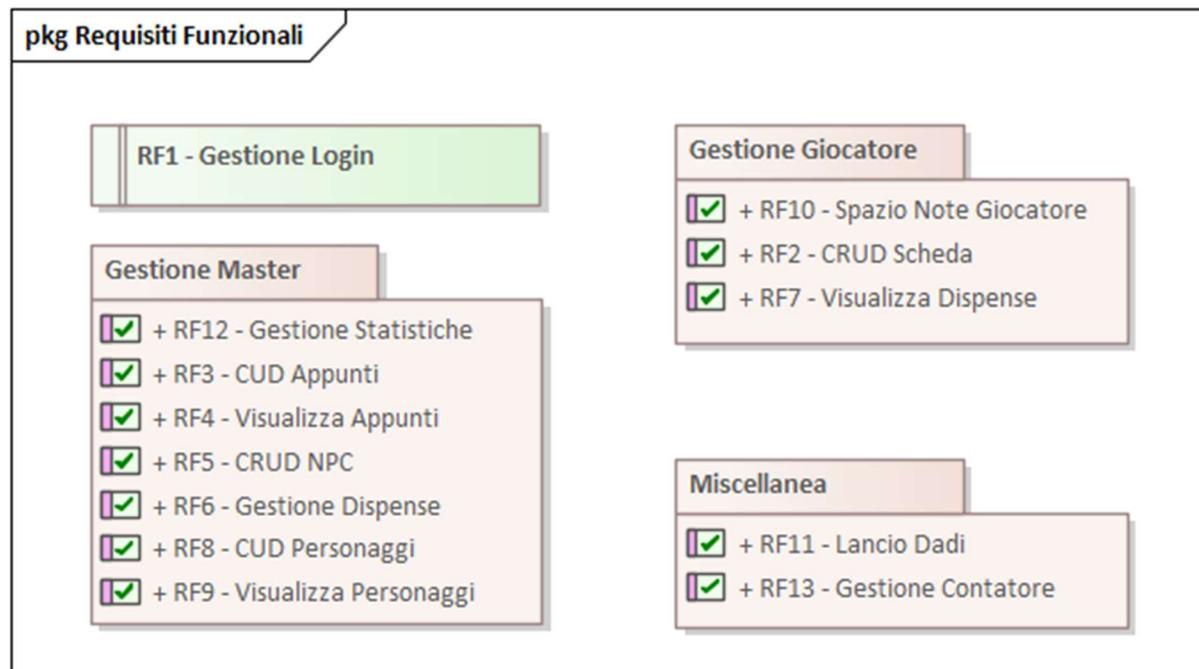
Gestione dei requisiti



Analisi dei requisiti



Requisiti funzionali



RF1 - Gestione login (da solo)

Il sistema dovrà consentire ai partecipanti di registrarsi nel sistema e di decidere username e password, modificabili successivamente dall'utente.



RF2 - CRUD Scheda (gestione giocatore)

Il sistema dovrà gestire le attività CRUD della scheda a disposizione del giocatore.

RF3 - CUD Appunti (gestione master)

Il sistema dovrà gestire le attività CUD degli appunti del master.

RF4 - Visualizza Appunti (gestione master)

Il sistema dovrà consentire al master di visualizzare gli appunti.

RF5 - CRUD NPC (gestione master)

Il sistema dovrà gestire le attività CRUD degli NPC.

RF6 - Gestione Dispense (gestione master)

Il sistema dovrà consentire al master di mostrare alcuni dei propri appunti ai giocatori a sua scelta.

RF7 - Visualizza Dispense (gestione giocatore)

Il sistema dovrà visualizzare gli appunti messi a disposizione del Master.

RF8 - CUD Personaggi (gestione master)

Il sistema dovrà consentire al Master di gestire le attività CUD di tutti i PG.

RF9 - Visualizza Personaggi (gestione master)

Il sistema dovrà consentire al Master di visualizzare tutte le informazioni riguardanti i PG.

RF10 - Spazio Note Giocatore (gestione giocatore)

Il sistema dovrà consentire ai Giocatori di scriversi degli appunti personali, in uno spazio apposito all'interno dell'account.

RF11 - Lancio dadi (miscellanea)

Il sistema dovrà consentire agli utenti di effettuare il lancio dei dadi, permettendo inoltre di lanciare un numero di dadi a piacere.

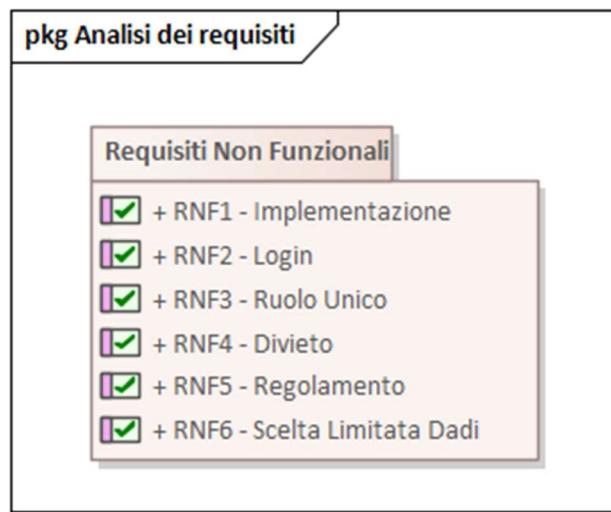
RF12 - Gestione Statistiche (gestione master)

Il sistema dovrà generare le statistiche riguardanti i dati dei personaggi su richiesta del Master.

RF13 - Gestione Contatore (miscellanea)

Il sistema dovrà consentire al master di gestire il conteggio del numero di sessioni svolte durante la campagna.

Requisiti non funzionali



RNF1 - Implementazione

Il sistema dovrà essere realizzato in tecnologia Python (e dovrà avere anche un’interfaccia grafica).

RNF2 - Login

Il sistema dovrà essere utilizzato attraverso il login dei partecipanti, i quali possono possedere un account definito da un ID univoco e immodificabile, username e password.

RNF3 - Ruolo Unico

Il sistema dovrà consentire ad ogni partecipante la gestione di un solo Personaggio o del ruolo di Master.

RNF4 - Divieto

Il sistema non consentirà ai giocatori di poter gestire i dati che appartengono agli account diversi dal proprio.

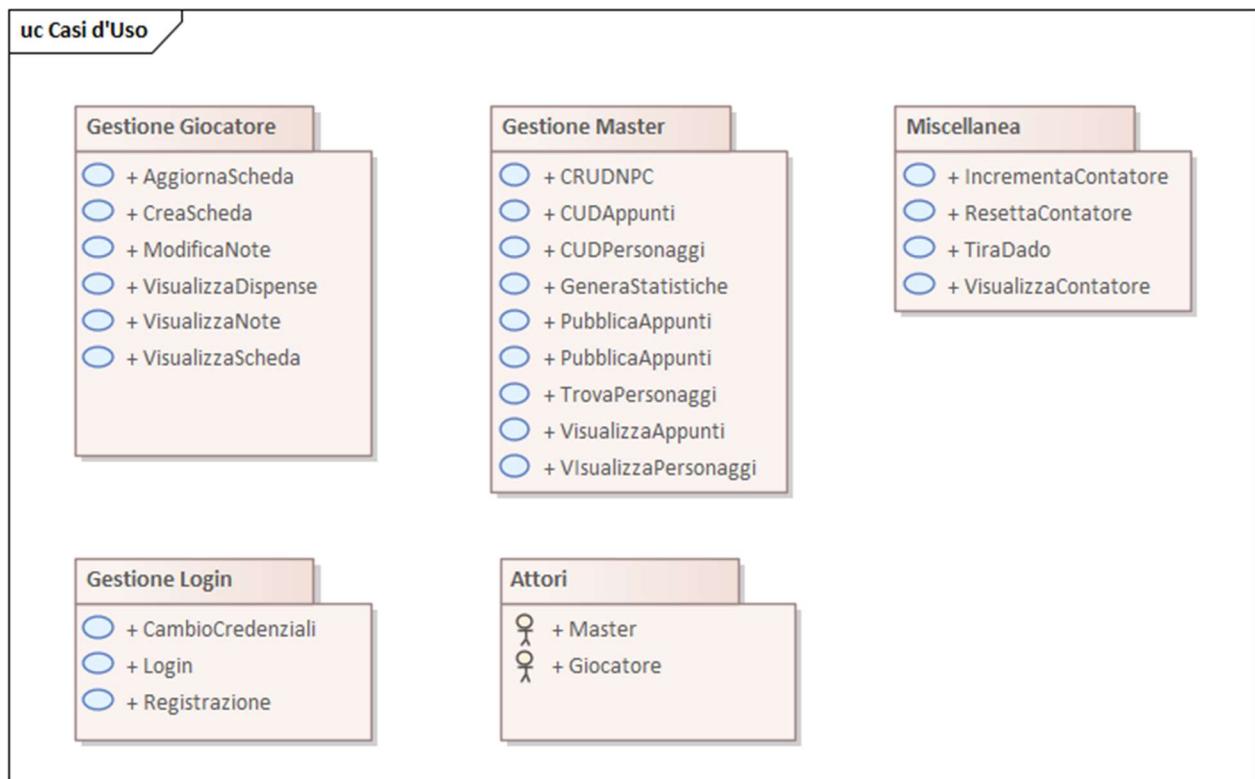
RNF5 - Regolamento

Il sistema dovrà impedire agli utenti di inserire i dati in maniera che violino il regolamento di D&D.

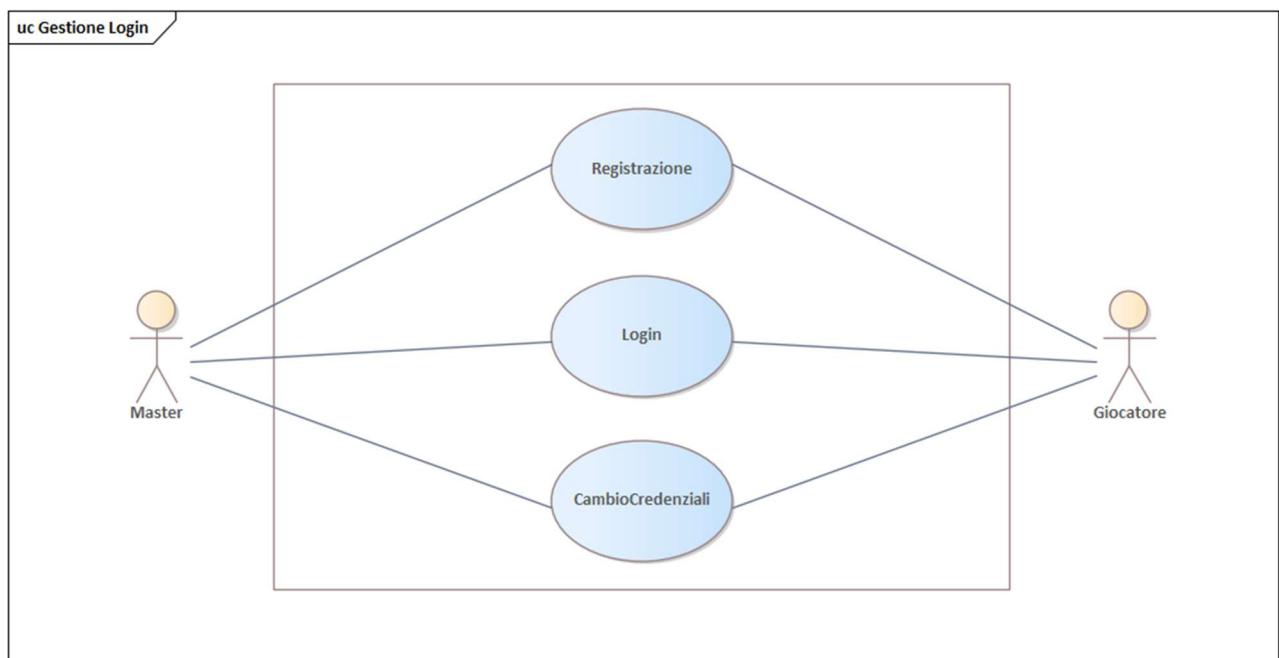
RNF6 - Scelta Limitata Dadi

Il sistema non consentirà agli utenti di lanciare dadi che non rientrano tra i 6 tipi predefiniti.

Casi d'uso



Gestione Login



1°

Caso d'uso: **Registrazione**



Breve descrizione: Questo caso d'uso consente all'utente registrarsi nel sistema.

Attori primari: Master, Giocatore.

Attori secondari: nessuno.

Precondizioni: nessuna.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole registrarsi all'interno del sistema.
2. Il sistema chiede all'attore primario di scegliere un username e una password.
3. *if* Lo username è già esistente
 - 3.1. Il sistema restituisce un errore.
4. *else*
 - 4.1. Il sistema registra i dati inseriti dall'attore primario.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

2°

Caso d'uso: **Login**

Breve descrizione: Questo caso d'uso consente all'attore primario di accedere al sistema.

Attori primari: Master, Giocatore.

Attori secondari: nessuno.

Precondizioni: Nessuna.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole accedere al sistema.
2. Il sistema chiede all'attore primario di inserire lo username e la password.
3. *if* Lo username non esiste
 - 3.1. Il sistema restituisce un errore.
4. *else if* La password non è corretta
 - 4.1. Il sistema restituisce un errore.
5. *else*

5.1. Il sistema autentica l'attore primario.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna



3°

Caso d'uso: **CambioCredenziali**

Breve descrizione: Questo caso d'uso consente all'attore primario di cambiare le proprie credenziali.

Attori primari: Master, Giocatore.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere già autenticato all'interno del sistema.

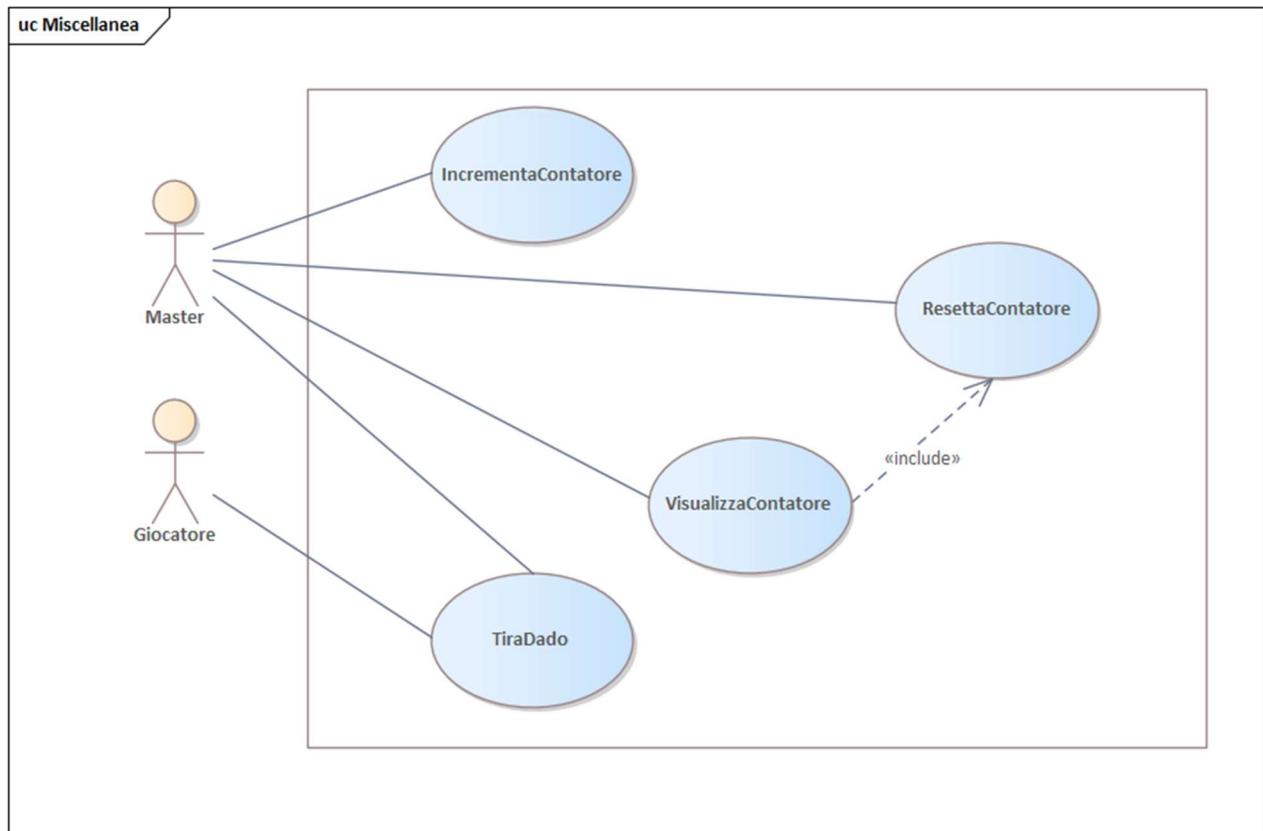
Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario desidera cambiare le proprie credenziali.
2. Il sistema chiede all'attore primario di inserire le nuove credenziali.
3. *if* Lo username è già esistente
 - 3.1. Il sistema restituisce un errore.
4. *else*
 - 4.1. Il sistema registra le nuove credenziali.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

Miscellanea



4°

Caso d'uso: **IncrementaContatore**

Breve descrizione: Questo caso d'uso consente all'attore primario di incrementare il contatore.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'utente deve essere già autenticato come Master.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario decide di incrementare il contatore.
2. Il sistema incrementa il contatore.

Postcondizioni: Il contatore è incrementato.

Autori: Rossini Alessandro, Parisi Francesco Romeo, Othmani Amir

Sequenza degli eventi alternativa: nessuna.



5°

Caso d'uso: **VisualizzaContatore**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare il contatore.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'utente deve essere già autenticato come Master.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole visualizzare il contatore.
2. *if* Il contatore è minore di 0
 - 2.1. include (ResettaContatore).
3. Il sistema visualizza il contatore.

Postcondizioni: Il contatore è visualizzato.

Sequenza degli eventi alternativa: nessuna.

6°

Caso d'uso: **ResettaContatore**.

Breve descrizione: Questo caso d'uso consente all'attore primario di resettare il contatore.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'utente deve essere già autenticato come Master.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole resettare il contatore.
2. Il sistema resetta il contatore.

Postcondizioni: Il contatore è resettato.

Sequenza degli eventi alternativa: nessuna.



7°

Caso d'uso: **TiraDado**

Breve descrizione: Questo caso d'uso consente all'attore primario di tirare i dadi.

Attori primari: Master, Giocatore.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere già autenticato all'interno del sistema.

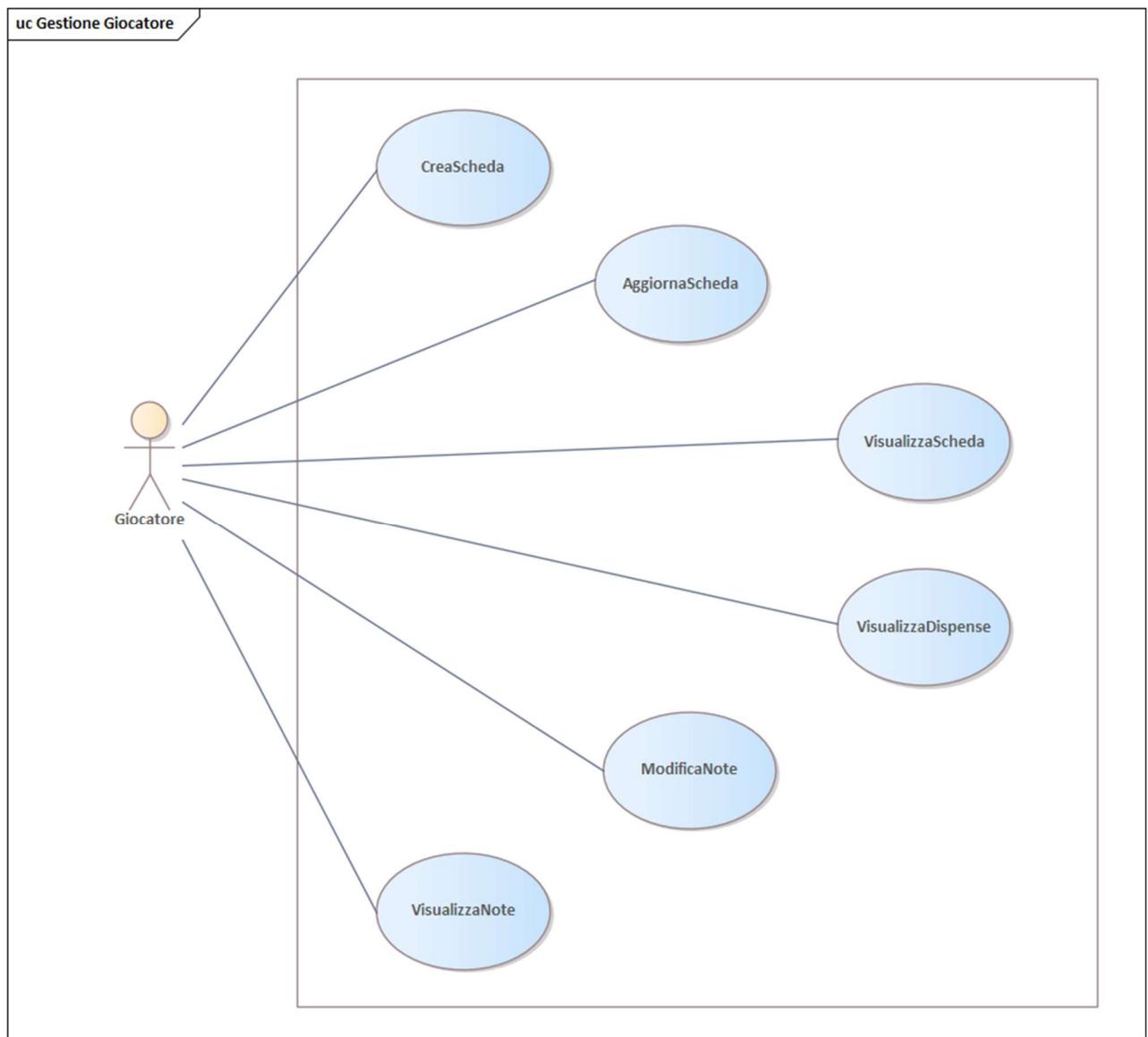
Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole tirare i dadi.
2. Il sistema chiede all'attore primario di selezionare il tipo e la quantità di dadi.
3. Il sistema effettua il tiro dei dadi scelti.
4. Il sistema visualizza il risultato.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

Gestione Giocatore



8°

Caso d'uso: **CreaScheda**

Breve descrizione: Questo caso d'uso consente all'attore primario di creare una scheda del proprio PG.

Attori primari: Giocatore.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere già autenticato all'interno del sistema.



Sequenza degli eventi principale:

1. Il caso d'uso inizia quando L'attore primario vuole creare una scheda del proprio PG.
2. Il sistema chiede all'attore primario i parametri necessari alla creazione della scheda.
3. L'attore primario inserisce i dati.
4. Il sistema crea la scheda con i dati inseriti.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

9°

Caso d'uso: **AggiornaScheda**

Breve descrizione: Questo caso d'uso consente all'attore primario di modificare la propria scheda.

Attori primari: Giocatore.

Attori secondari: nessuno.

Precondizioni:

1. L'attore primario deve essere già autenticato all'interno del sistema.
2. L'attore primario deve già possedere una scheda.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole aggiornare i dati della propria scheda.
2. Il sistema chiede all'attore primario quali parametri vuole modificare.
3. L'attore primario inserisce i dati da modificare.
4. Il sistema salva i dati acquisiti.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

10°

Caso d'uso: **VisualizzaScheda**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare la propria scheda.

Attori primari: Giocatore.



Attori secondari: nessuno.

Precondizioni:

1. L'attore primario deve essere già autenticato all'interno del sistema.
2. L'attore primario deve già possedere una scheda.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole visualizzare la propria scheda.
2. Il sistema visualizza la scheda.

Postcondizioni: La scheda è visualizzata.

Sequenza degli eventi alternativa: nessuna.

11°

Caso d'uso: **VisualizzaDispense**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare le dispense.

Attori primari: Giocatore.

Attori secondari: nessuno.

Precondizioni:

- L'attore primario deve essere già autenticato all'interno del sistema.
L'attore primario deve già possedere una scheda.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole visualizzare le dispense.
2. Il sistema visualizza le dispense.

Postcondizioni: Le dispense devono essere visualizzate.

Sequenza degli eventi alternativa: nessuna.

12°

Caso d'uso: **ModificaNote**



Breve descrizione: Questo caso d'uso consente all'attore primario di modificare le sue note personali.

Attori primari: Giocatore.

Attori secondari: nessuno.

Precondizioni: nessuna.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole appuntarsi qualcosa che gli serve.
2. Il sistema apre la finestra dedicata e consente all'attore primario di poterci scrivere quello che vuole.
3. Il sistema salva i dati acquisiti.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

13°

Caso d'uso: **VisualizzaNote**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare le proprie note.

Attori primari: Giocatore.

Attori secondari: nessuno.

Precondizioni:

1. L'attore primario deve essere già autenticato all'interno del sistema.
2. L'attore primario deve già possedere una scheda

Sequenza degli eventi principale:

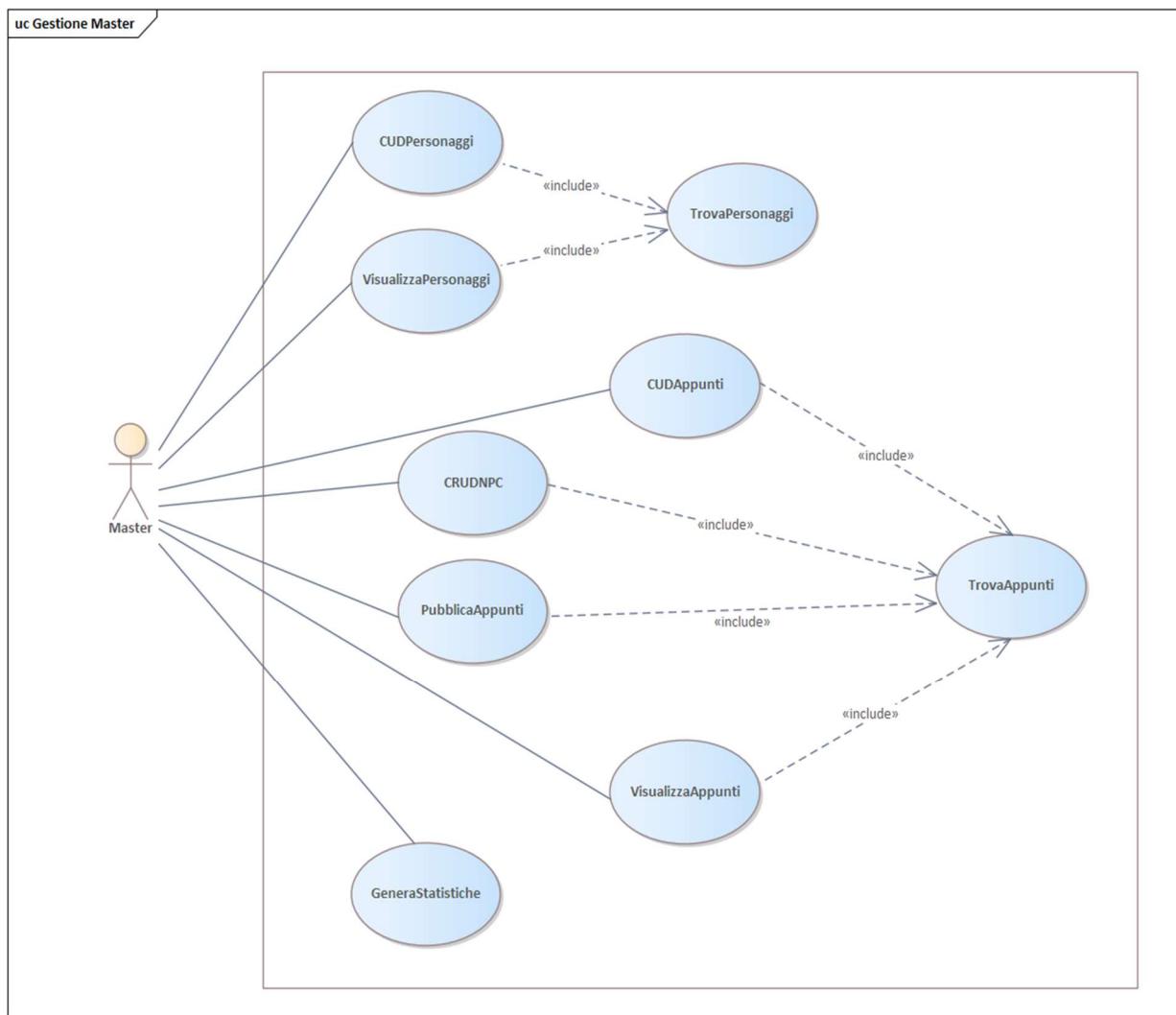
1. Il caso d'uso inizia quando l'attore primario vuole visualizzare le proprie note.
2. Il sistema visualizza le note.

Postcondizioni: Le note sono visualizzate.

Sequenza degli eventi alternativa: nessuna.



Gestione Master



14°

Caso d'uso: AggiornaPersonaggi

Breve descrizione: Questo caso d'uso consente all'attore primario di aggiornare i personaggi esistenti nella campagna.

Attori primari: Master.

Attori secondari: nessuno.

Autori: Rossini Alessandro, Parisi Francesco Romeo, Othmani Amir

Precondizioni: L'attore primario deve essere autenticato all'interno del sistema.



Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole aggiornare uno o più personaggi esistenti nella campagna.
2. include (TrovaPersonaggi).
3. Il sistema chiede i parametri da aggiornare.
4. L'attore primario inserisce i dati da modificare.
5. Il sistema salva i dati acquisiti.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

15°

Caso d'uso: **VisualizzaPersonaggi**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare i dati di tutti i personaggi esistenti nella campagna.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato nel sistema.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando vuole visualizzare i dati di uno o più personaggi esistenti nella campagna.
2. include (TrovaPersonaggi).
3. Il sistema visualizza il personaggio trovato.

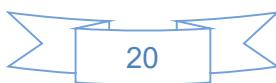
Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

16°

Caso d'uso: **TrovaPersonaggi**

Breve descrizione: Questo caso d'uso consente all'attore primario di effettuare la ricerca di un personaggio richiesto da lui.



Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato nel sistema.



Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole trovare uno specifico personaggio.
2. Il sistema chiede all'attore primario il nome del personaggio da cercare.
3. *if* Il personaggio è stato trovato
 - 3.1. Il sistema restituisce il personaggio trovato.
4. *else*
 - 4.1. Il sistema restituisce un errore.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

17°

Caso d'uso: **CUDAppunti**

Breve descrizione: Questo caso d'uso consente all'attore primario di gestire le attività CUD dei propri appunti.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato all'interno del sistema.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole gestire le attività CUD dei propri appunti.
2. *if* L'attore primario vuole creare un nuovo appunto
 - 2.1. Il sistema chiede il nome del nuovo appunto.
 - 2.2. L'attore primario inserisce il nome dell'appunto
 - 2.3. *if* è già presente un appunto con quel nome
 - 2.3.1. Il sistema restituisce un errore.
 - 2.4. Il sistema chiede all'attore primario di inserire i nuovi dati
 - 2.5. L'attore primario inserisce i dati dell'appunto.
 - 2.6. Il sistema crea l'appunto.

3. *else if* L'attore primario vuole aggiornare i propri appunti
 - 3.1. include (TrovaAppunti).
 - 3.2. Il sistema chiede all'attore primario quali sono i dati da modificare
 - 3.3. L'attore primario inserisce i dati da modificare.
 - 3.4. Il sistema salva i dati acquisiti.
4. *else if* L'attore primario vuole eliminare degli appunti
 - 4.1. include (TrovaAppunti).
 - 4.2. Il sistema elimina gli appunti trovati.



Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

18°

Caso d'uso: **CRUDNPC**

Breve descrizione: Questo caso d'uso consente all'attore primario di gestire le attività CRUD di tutti gli NPC esistenti nella campagna.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato nel sistema.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole gestire le attività CRUD di uno o più NPC esistenti nella campagna.
2. *if* L'attore primario vuole creare un nuovo NPC
 - 2.1. Il sistema chiede il nome del nuovo NPC.
 - 2.2. L'attore primario inserisce il nome del nuovo NPC.
 - 2.3. *if* è già presente un NPC con quel nome
 - 2.3.1. Il sistema restituisce un errore.
 - 2.4. Il sistema chiede all'attore primario i nuovi dati da inserire
 - 2.5. L'attore primario inserisce i dati.
 - 2.6. Il sistema crea l'NPC.
3. *else if* L'attore primario vuole visualizzare i dati di un NPC
 - 3.1. include (TrovaAppunti)
 - 3.2. Il sistema visualizza l'NPC trovato.
4. *else if* L'attore primario vuole aggiornare i dati di un NPC
 - 4.1. include (TrovaAppunti).

- 4.2. Il sistema chiede all'attore primario quali sono i dati da modificare
- 4.3. L'attore primario inserisce i dati da modificare.
- 4.4. Il sistema salva i dati acquisiti.
5. *else if* L'attore primario vuole eliminare un NPC
 - 5.1. include (TrovaAppunti).
 - 5.2. Il sistema elimina l'NPC trovato.



Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna

19°

Caso d'uso: **PubblicaAppunti**

Breve descrizione: Questo caso d'uso consente all'attore primario di pubblicare i suoi appunti.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato nel sistema.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole pubblicare parte dei suoi appunti.
2. include (TrovaAppunti).
3. Il sistema pubblica gli appunti scelti.

Postcondizioni: Gli appunti scelti dall'attore primario sono stati tramutati in dispense.

Sequenza degli eventi alternativa: nessuna

20°

Caso d'uso: **VisualizzaAppunti**

Breve descrizione: Questo caso d'uso consente all'attore primario di visualizzare gli appunti di tutti i giocatori presenti nella campagna.

Attori primari: Master.

Attori secondari: nessuno

Precondizioni: L'attore primario deve essere autenticato nel sistema.



Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole visualizzare gli appunti di uno o più giocatori presenti nella campagna.
2. include (TrovaAppunti).
3. Il sistema visualizza gli appunti del giocatore desiderato.

Postcondizioni: nessuna

Sequenza degli eventi alternativa: nessuna

21°

Caso d'uso: **TrovaAppunti**

Breve descrizione: Questo caso d'uso consente all'attore primario di effettuare la ricerca degli appunti richiesti da lui.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: L'attore primario deve essere autenticato nel sistema.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole ricercare gli appunti di cui ha bisogno.
2. Il sistema chiede all'attore primario quali appunti vuole trovare.
3. *if* Gli appunti sono stati trovati
 - 3.1. Il sistema restituisce gli appunti trovati.
4. *else*
 - 4.1. Il sistema restituisce un errore.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

22°

Caso d'uso: **GeneraStatistiche**



Breve descrizione: Questo caso d'uso consente all'attore primario di generare le statistiche riguardanti tutti i partecipanti alla campagna.

Attori primari: Master.

Attori secondari: nessuno.

Precondizioni: nessuna.

Sequenza degli eventi principale:

1. Il caso d'uso inizia quando l'attore primario vuole generare le statistiche riguardanti la campagna.
2. Il sistema genera le statistiche basate sui dati in possesso.

Postcondizioni: nessuna.

Sequenza degli eventi alternativa: nessuna.

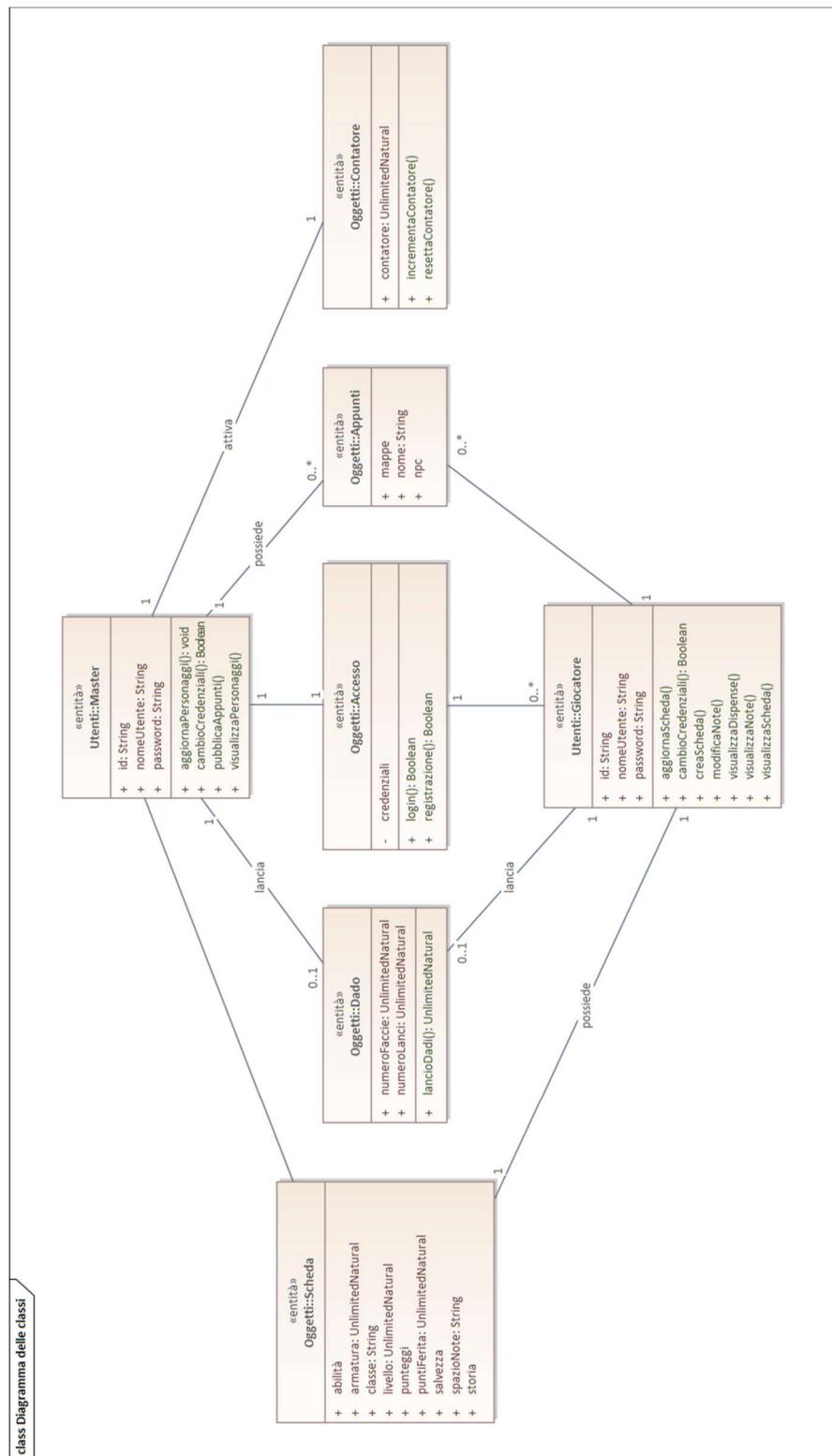
Matrice di Mapping



Target		Gestione Giocatore::RF10 - Spazio Note Giocatore	Gestione Giocatore::RF2 - CRUD Scheda	Gestione Giocatore::RF7 - Visualizza Dispense	Gestione Master::RF12 - Gestione Statistiche	Gestione Master::RF3 - CUD Appunti	Gestione Master::RF4 - Visualizza Appunti	Gestione Master::RF5 - CRUD NPC	Gestione Master::RF6 - Gestione Dispense	Gestione Master::RF8 - CUD Personaggi	Gestione Master::RF9 - Visualizza Personaggi	Miscellanea::RF11 - Lancio Dadi	Miscellanea::RF13 - Gestione Contatore	Requisiti Funzionali::RF1 - Gestione Login
Source														
Casi d'Uso::AggiornaScheda		↑												
Casi d'Uso::CambioCredenziali														↑
Casi d'Uso::CreaScheda		↑												
Casi d'Uso::CRUDNPC							↑							
Casi d'Uso::CUDAppunti				↑										
Casi d'Uso::CUDPersonaggi									↑					
Casi d'Uso::GeneraStatistiche			↑											
Casi d'Uso::IncrementaContatore													↑	
Casi d'Uso::Login														↑
Casi d'Uso::ModificaNote	↑													
Casi d'Uso::PubblicaAppunti								↑						
Casi d'Uso::Registrazione														↑
Casi d'Uso::ResettaContatore												↑		
Casi d'Uso::TiraDado													↑	
Casi d'Uso::TrovaAppunti					↑	↑	↑	↑						
Casi d'Uso::TrovaPersonaggi										↑	↑			
Casi d'Uso::VisualizzaAppunti						↑								
Casi d'Uso::VisualizzaContatore													↑	
Casi d'Uso::VisualizzaDispense			↑											
Casi d'Uso::VisualizzaNote	↑													
Casi d'Uso::VisualizzaPersonaggi											↑			
Casi d'Uso::VisualizzaScheda		↑												

Analisi

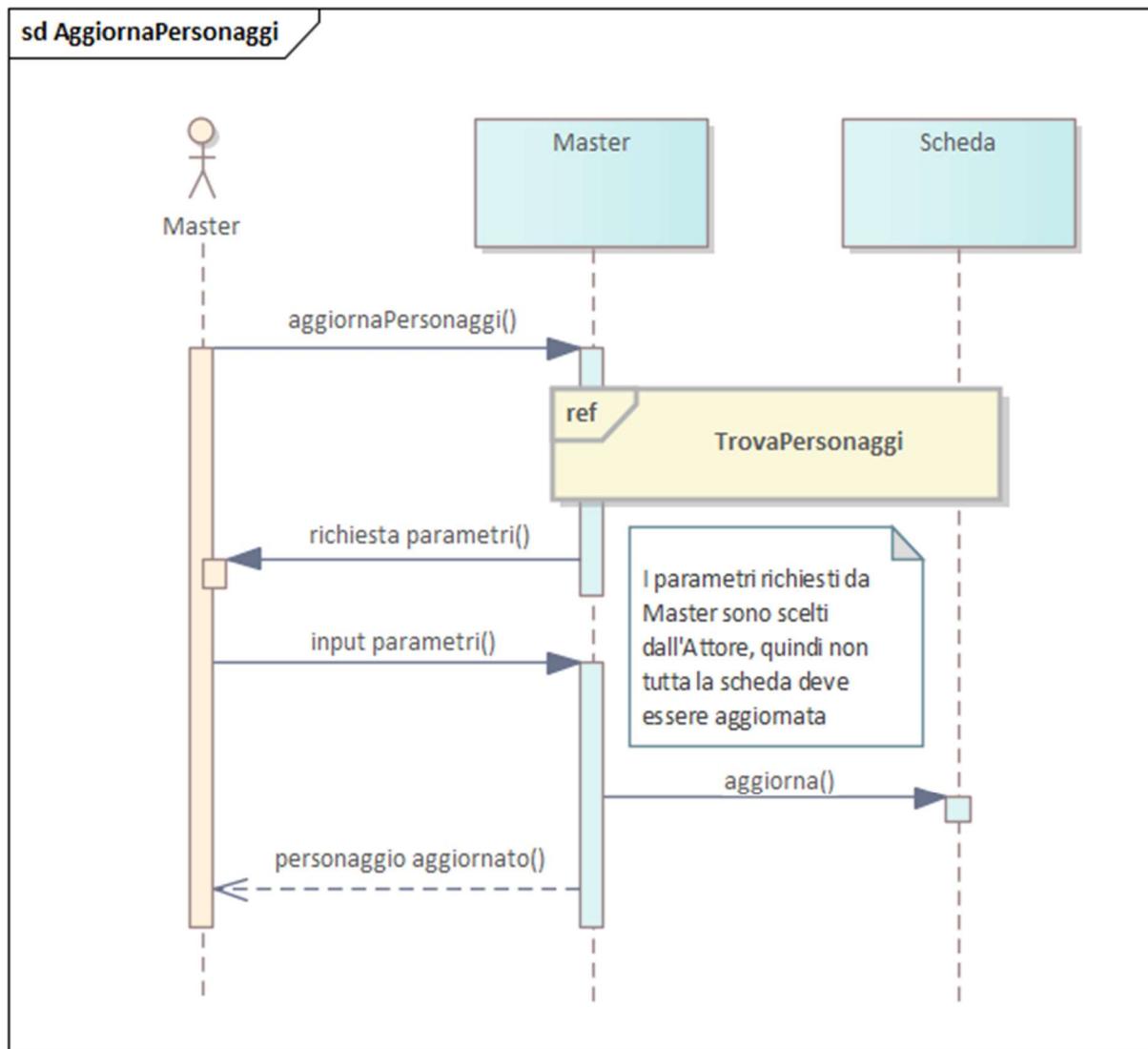
Classi di analisi



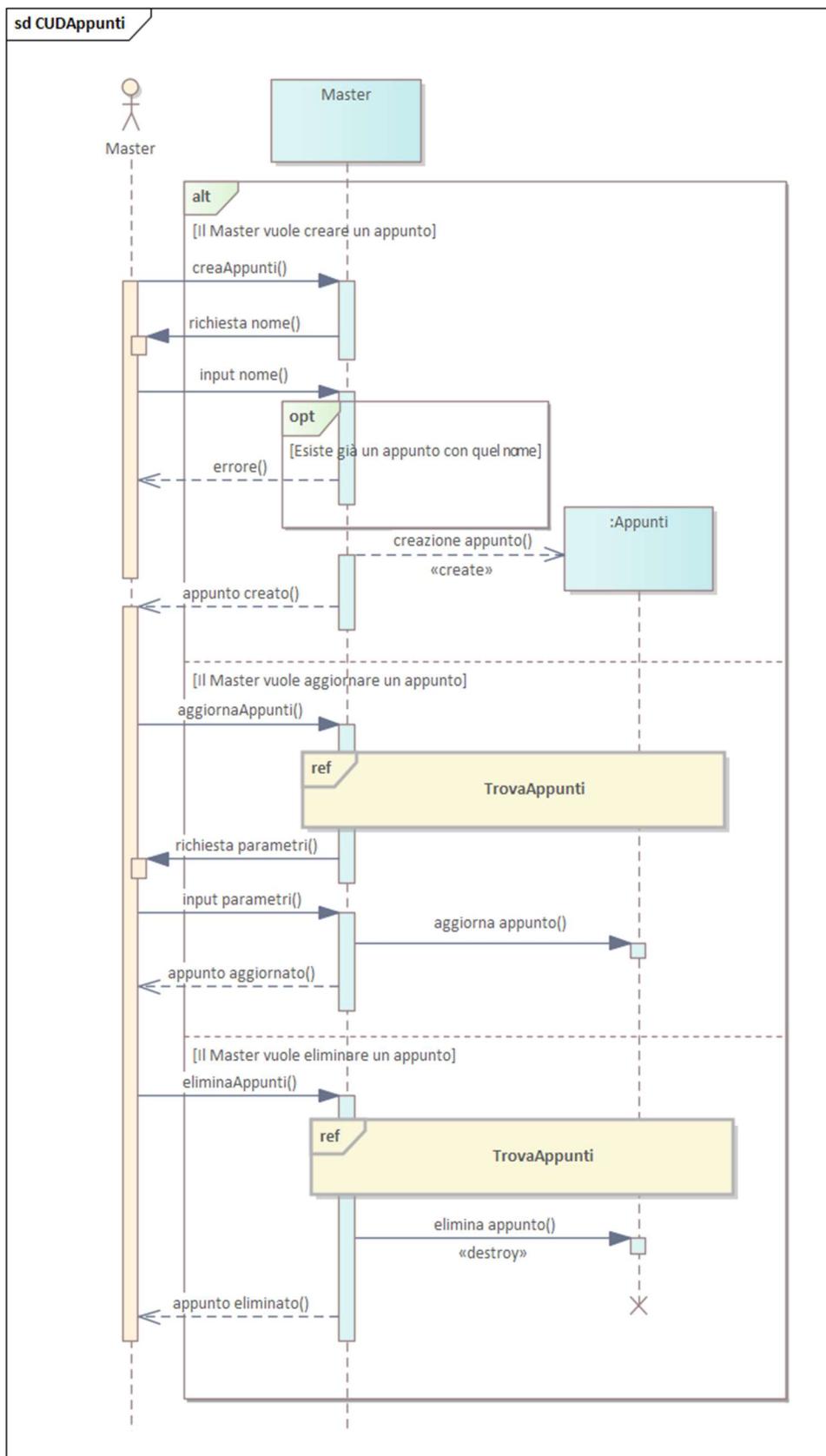
Diagrammi di sequenza



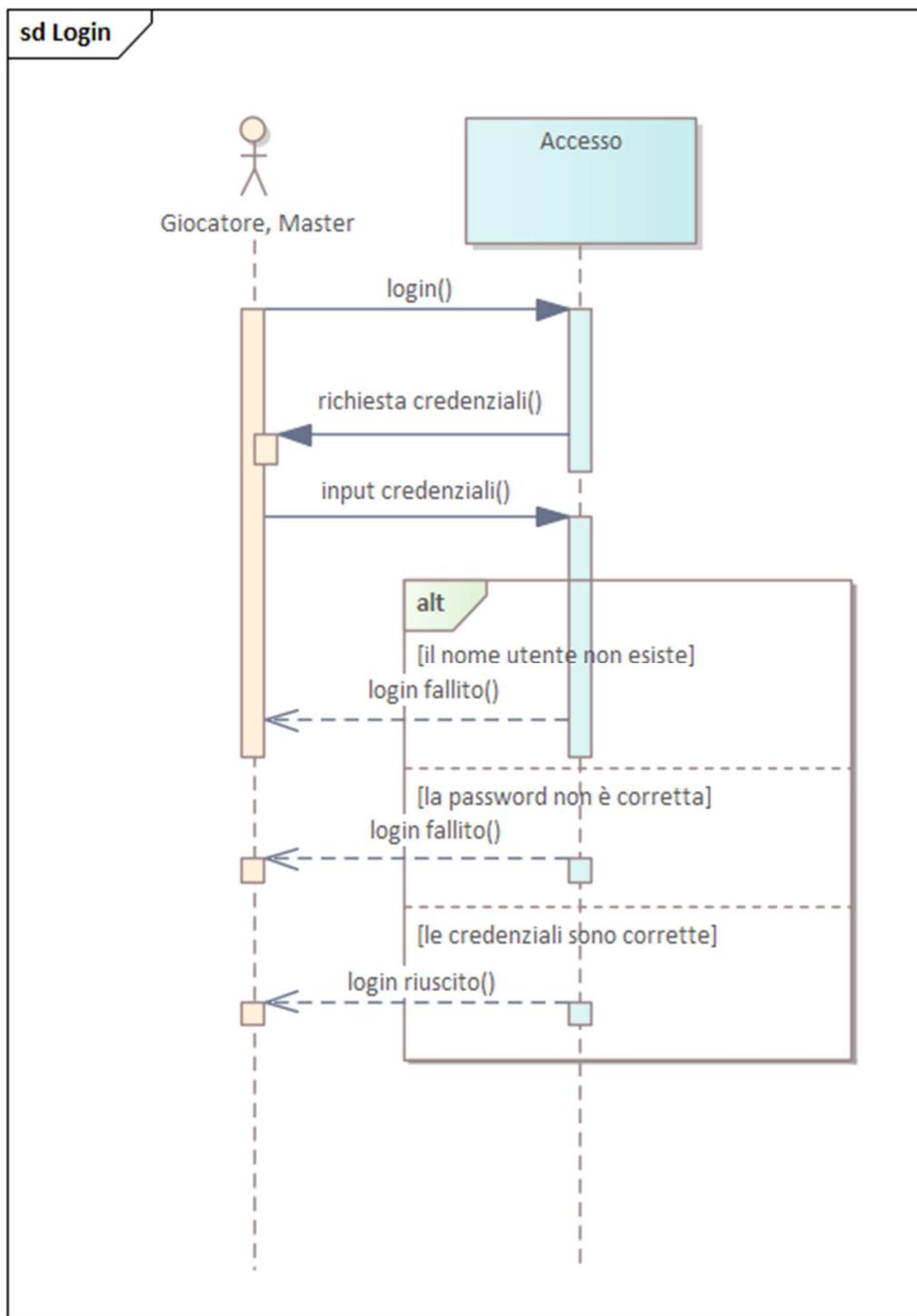
AggiornaPersonaggi



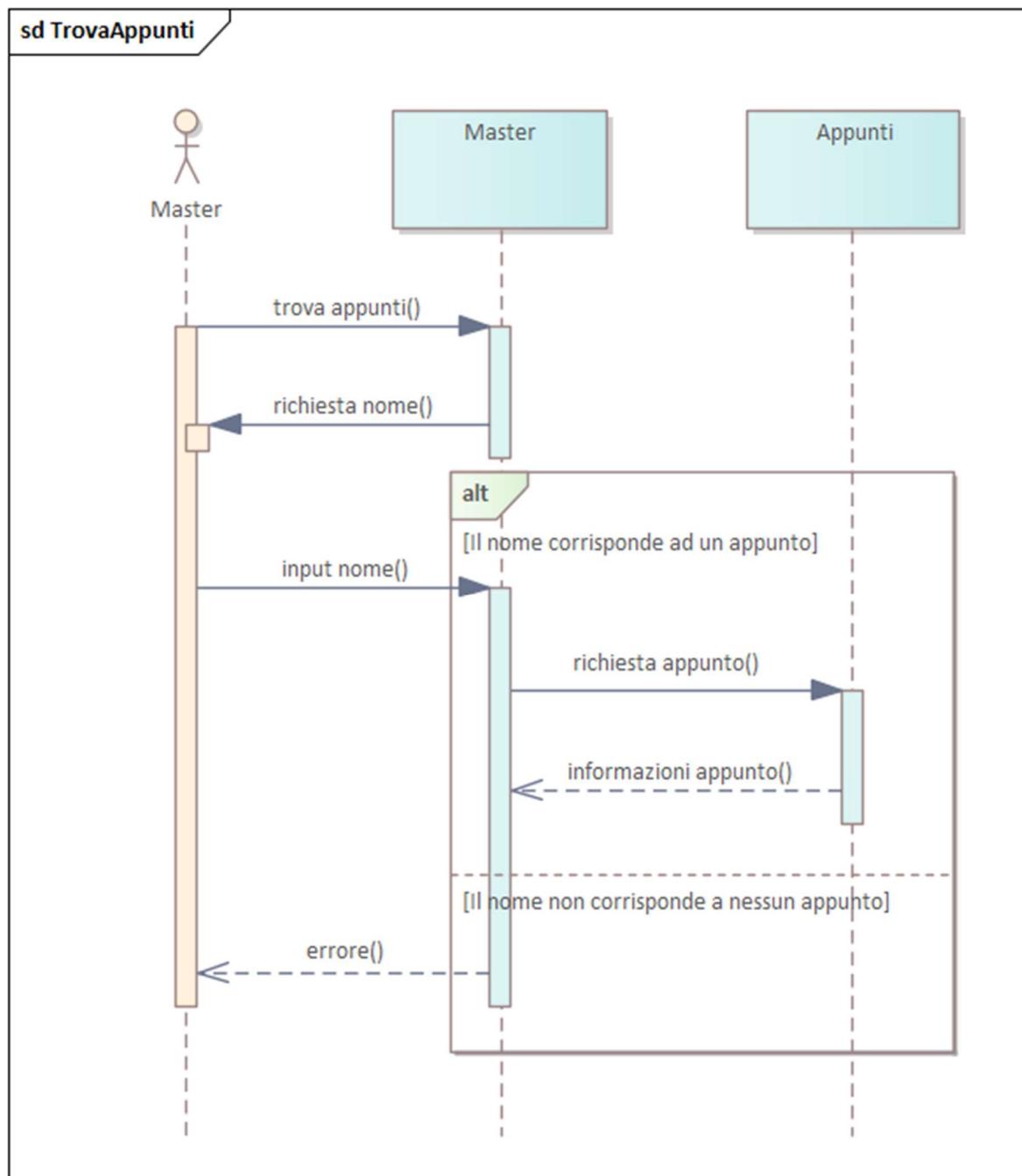
CUDAppunti



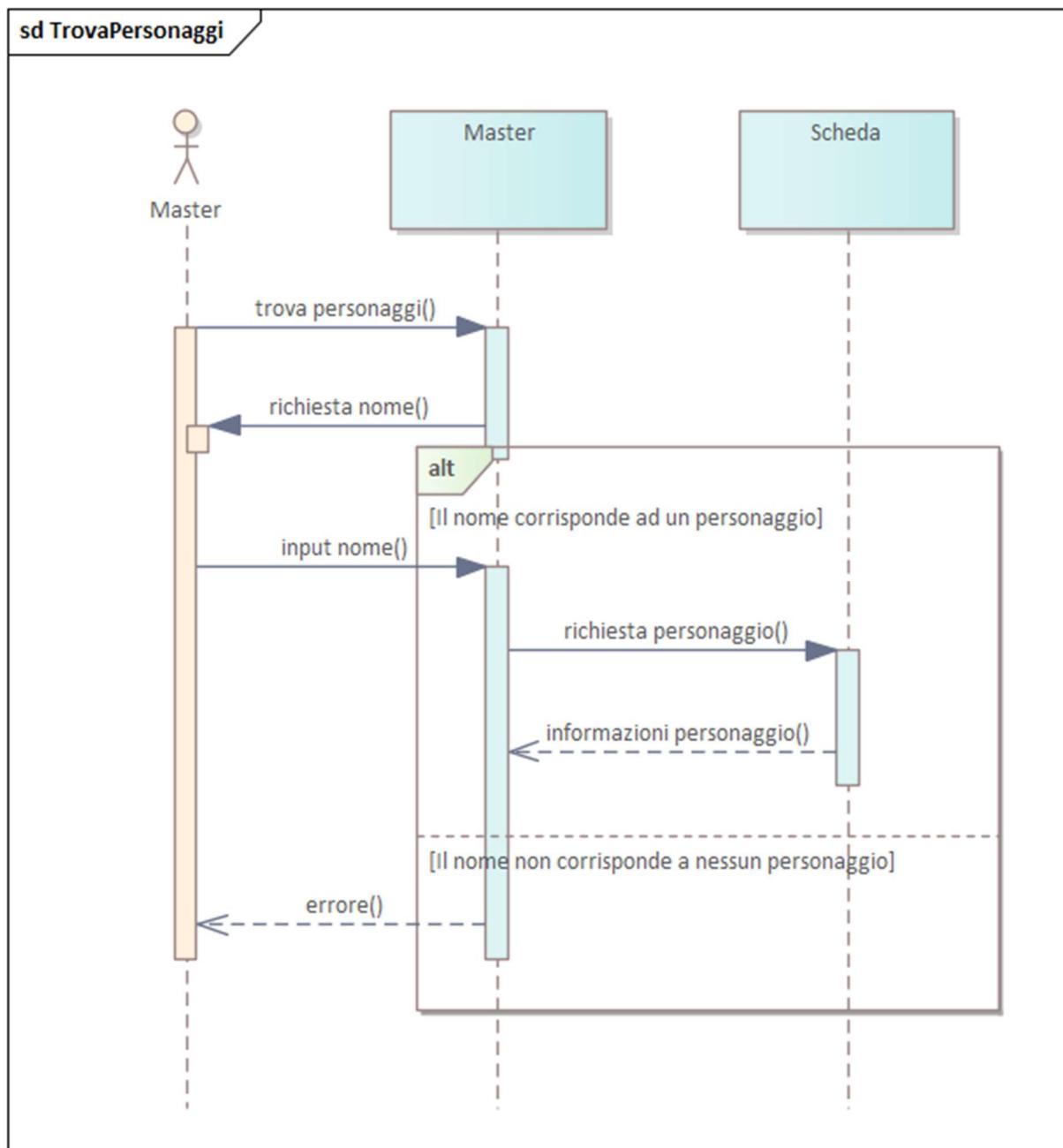
Login



TrovaAppunti



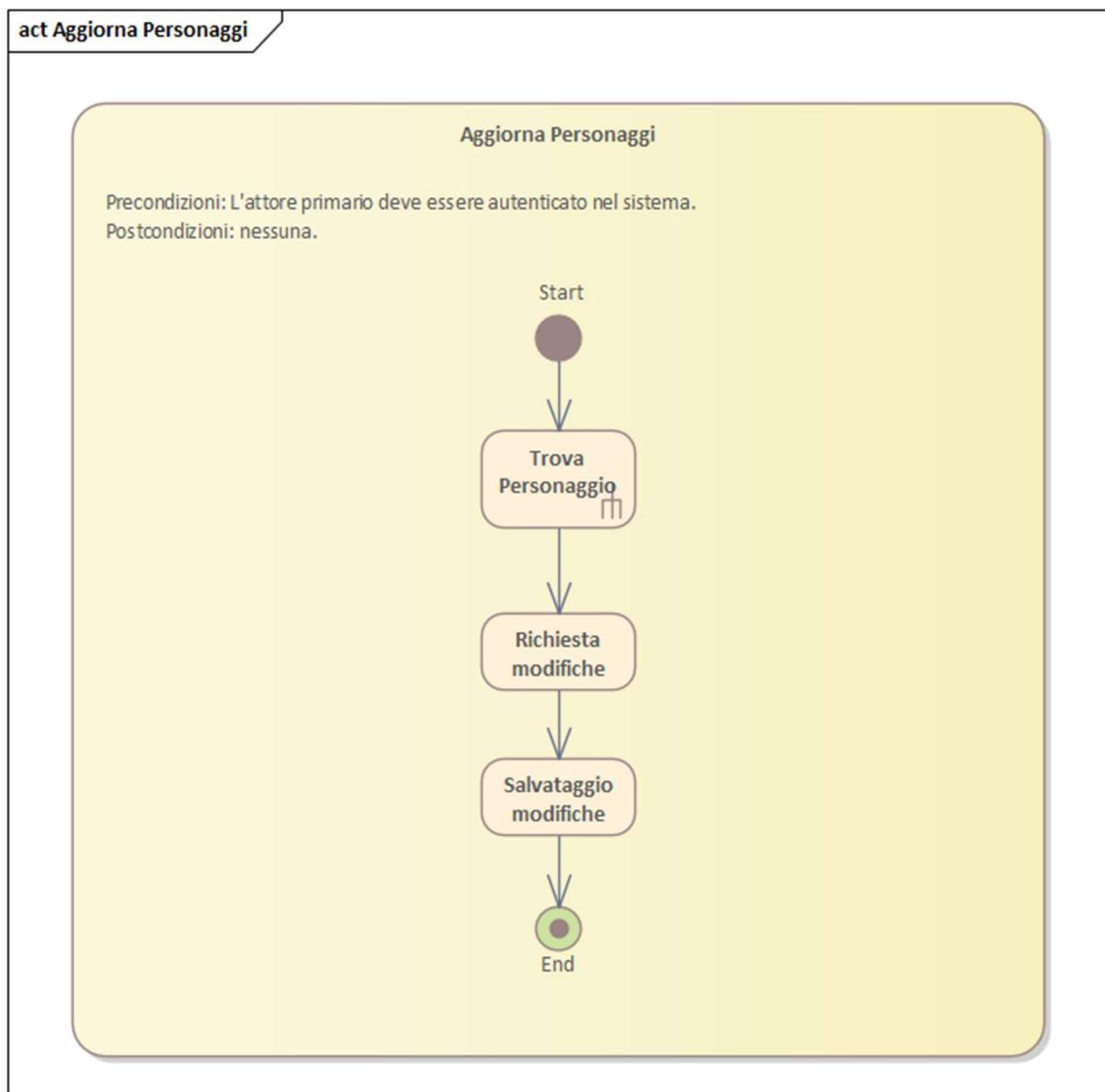
TrovaPersonaggi



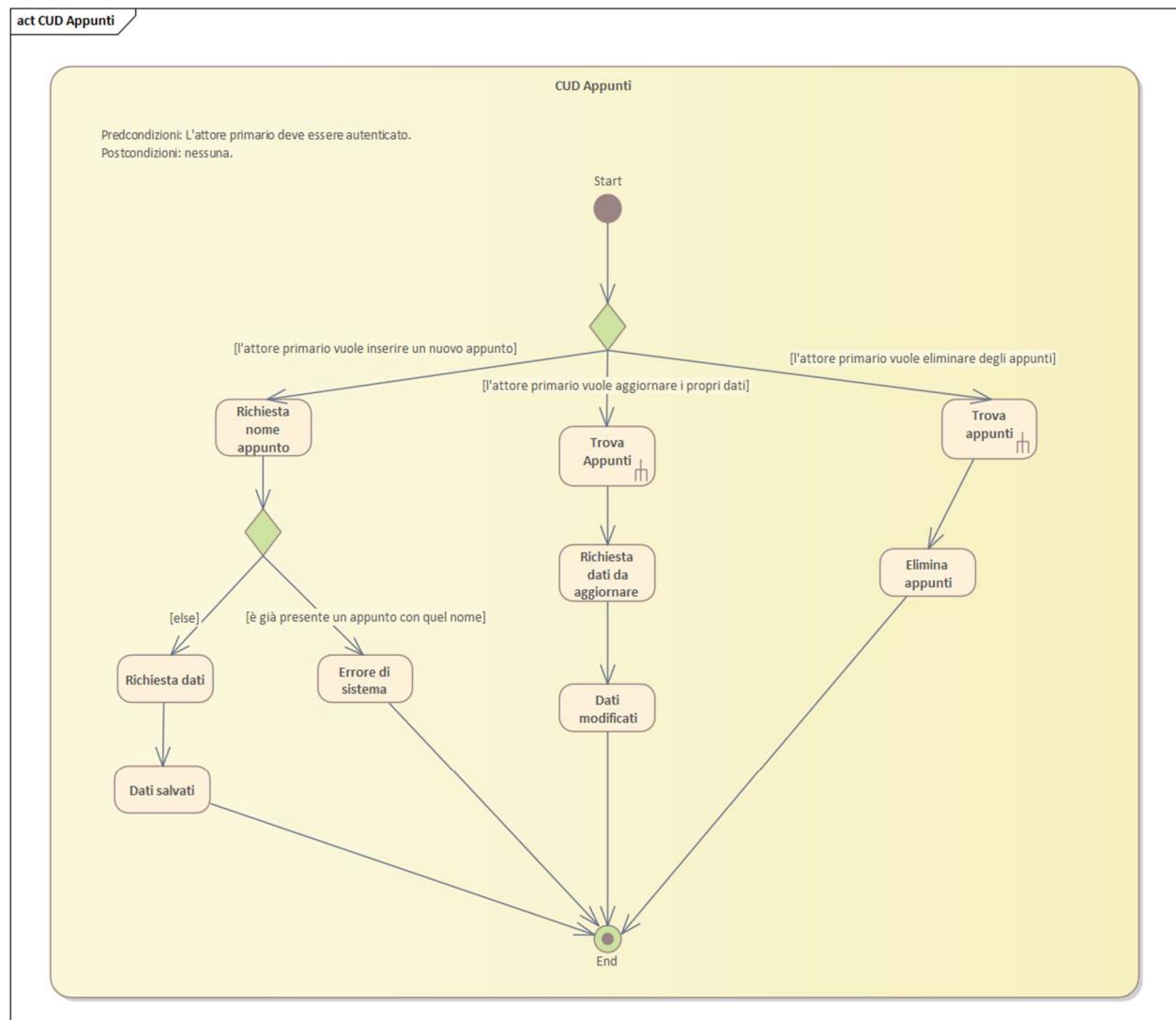
Diagrammi di attività



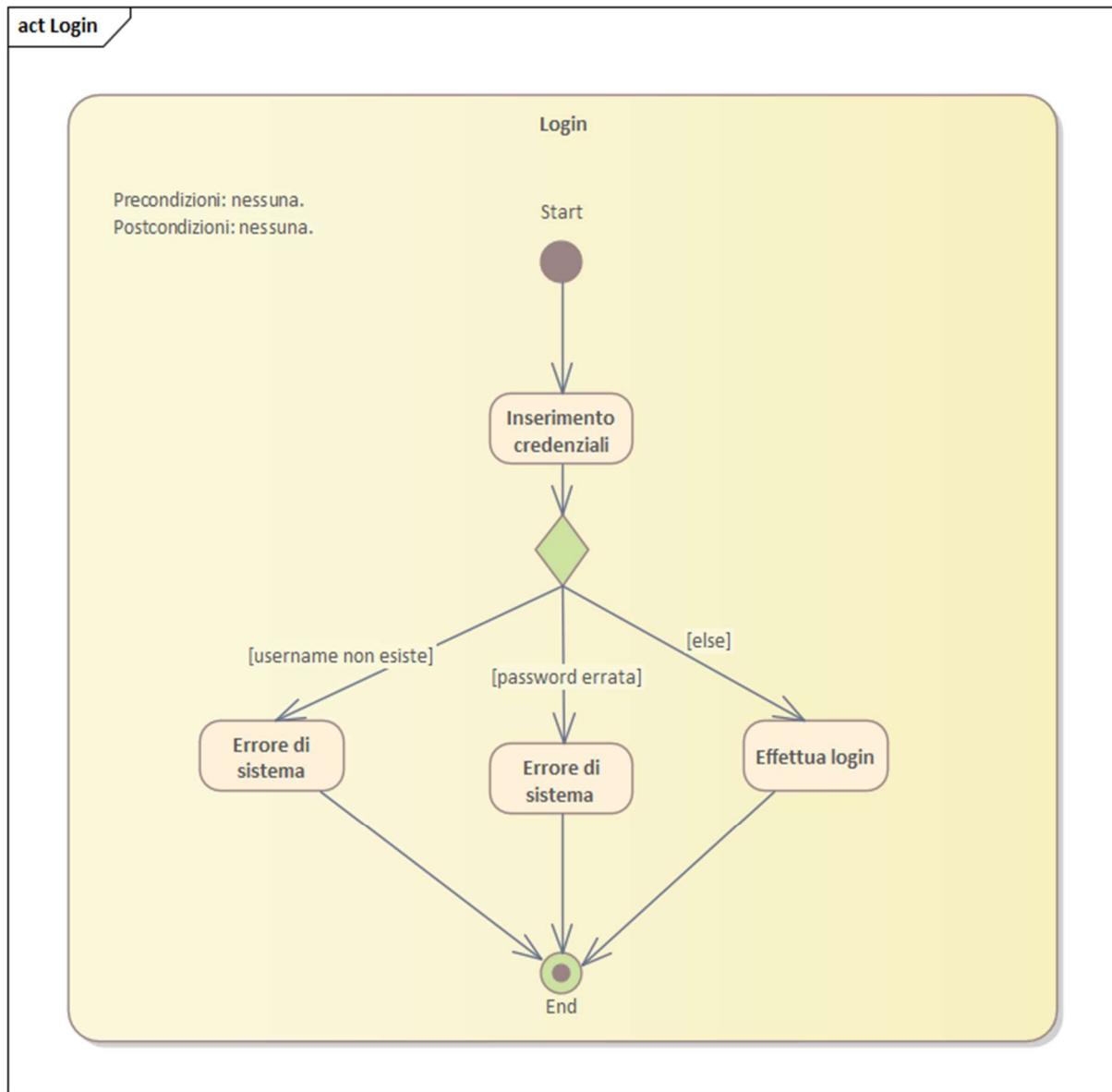
Aggiorna Personaggi



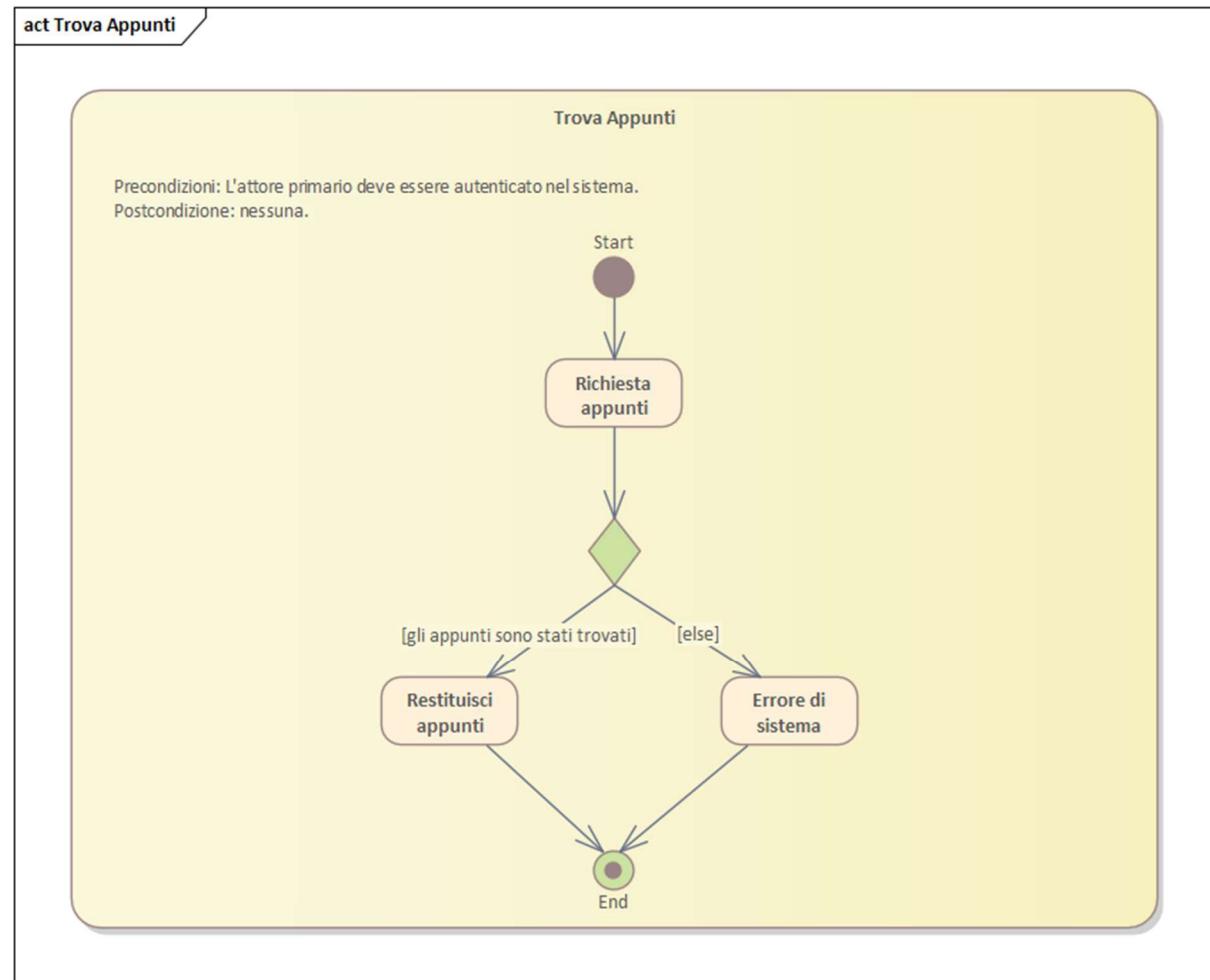
CUD Appunti



Login



Trova Appunti



Trova Personaggi

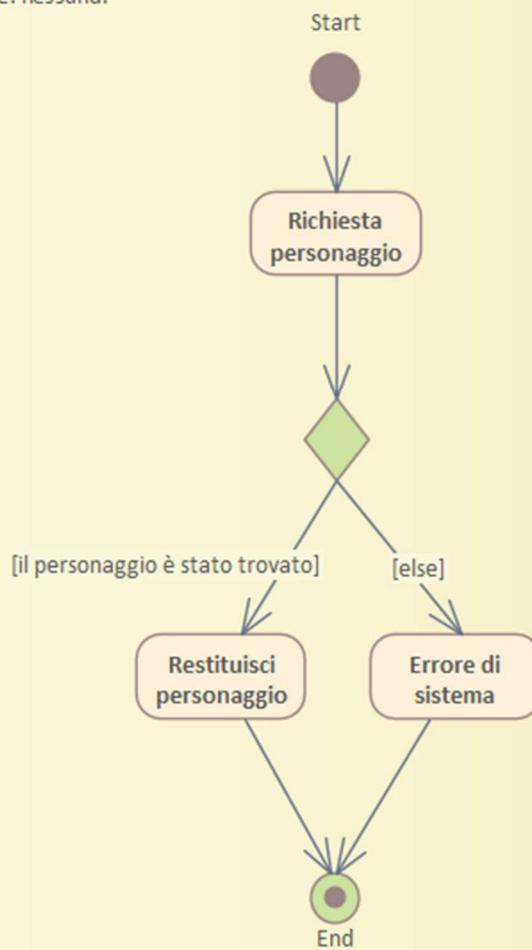


act Trova personaggi

Trova Personaggi

Precondizioni: L'attore primario deve essere autenticato nel sistema.

Postcondizione: nessuna.

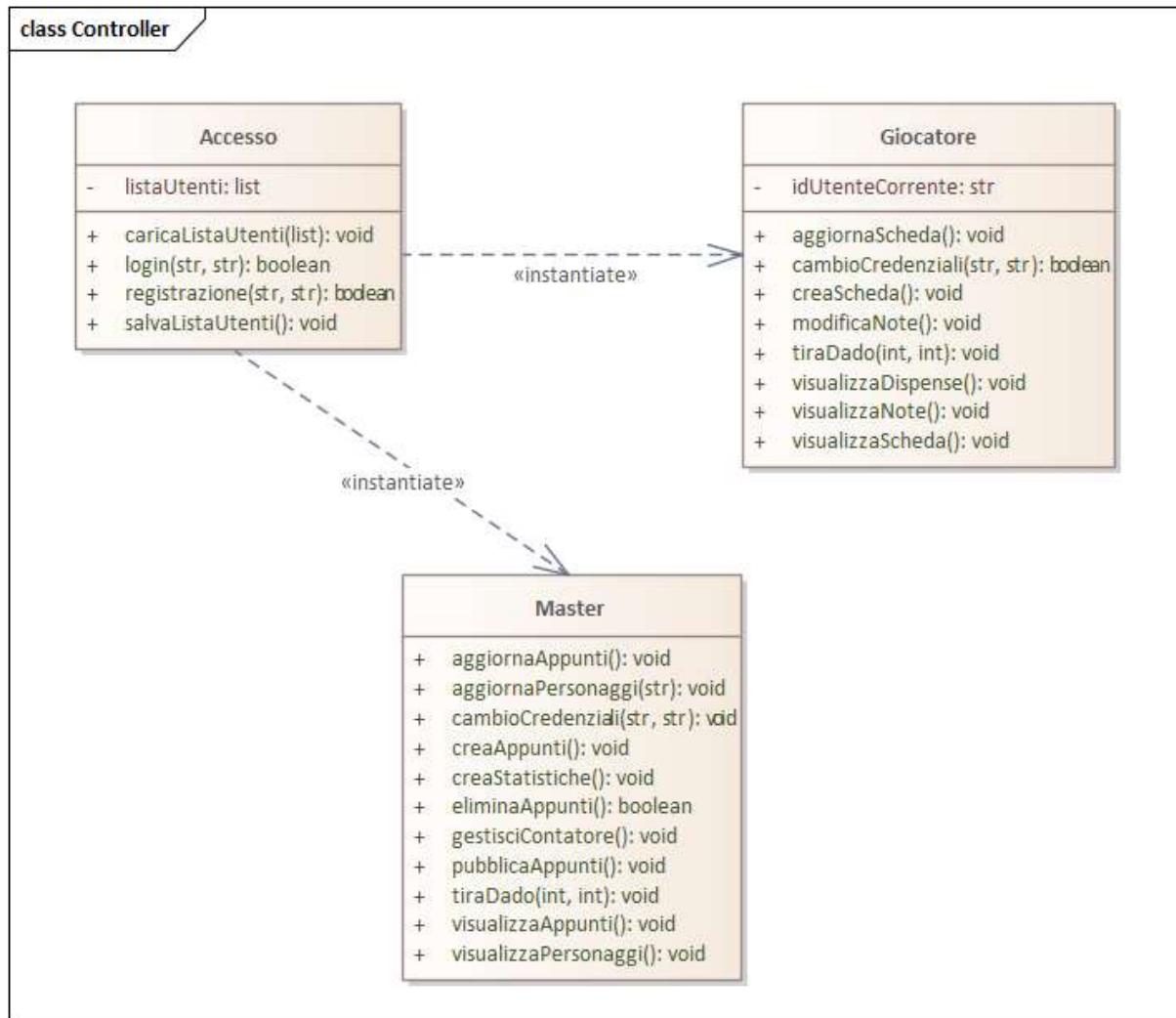


Progettazione

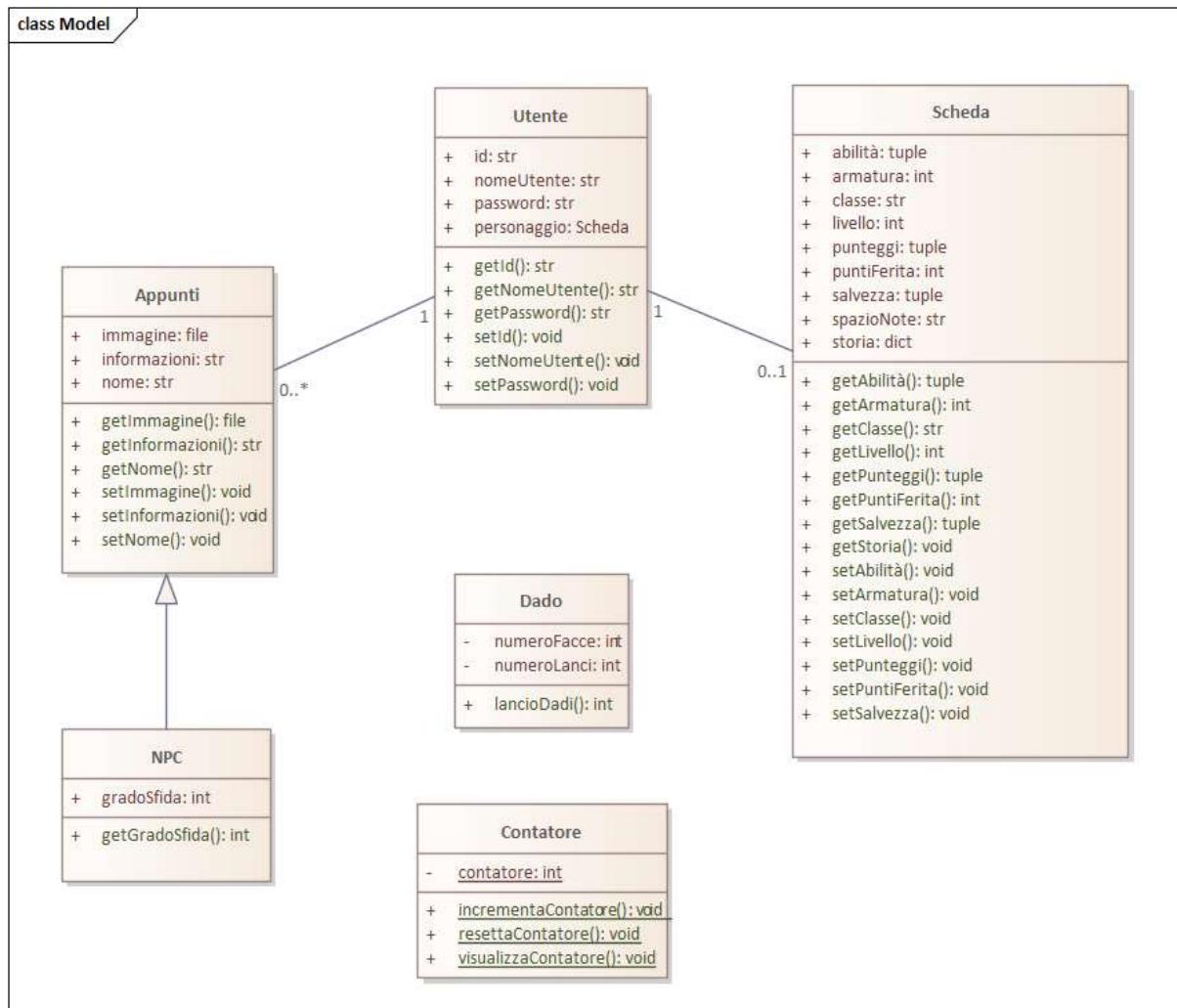


Classi di progettazione

Controller



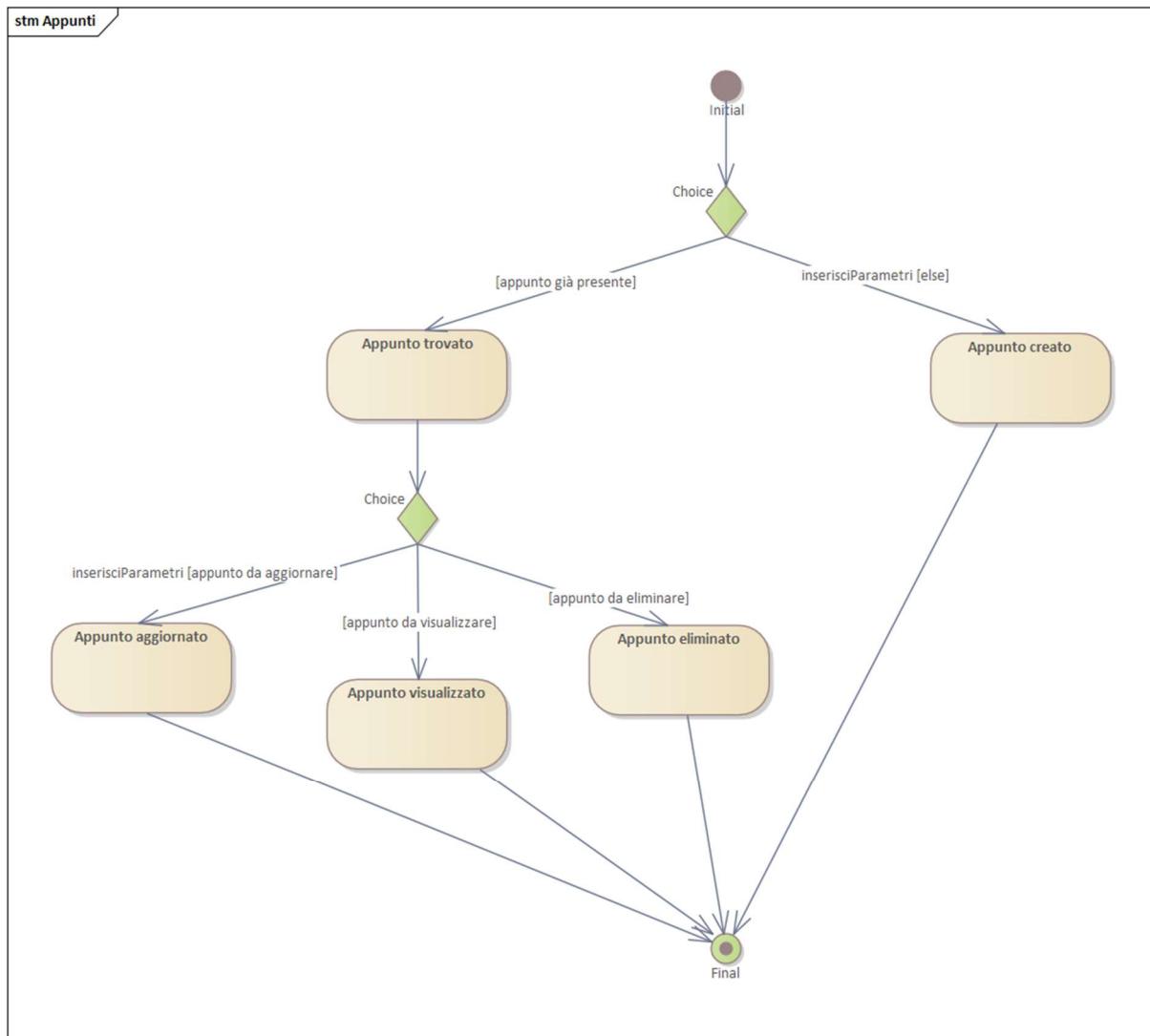
Model



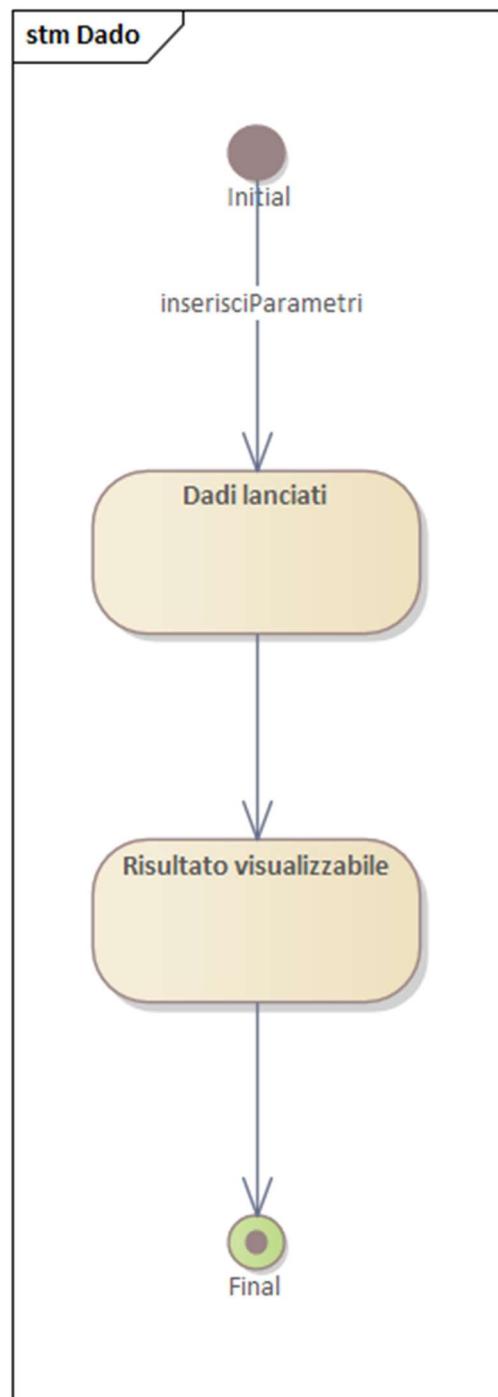
Diagrammi delle macchine a stati



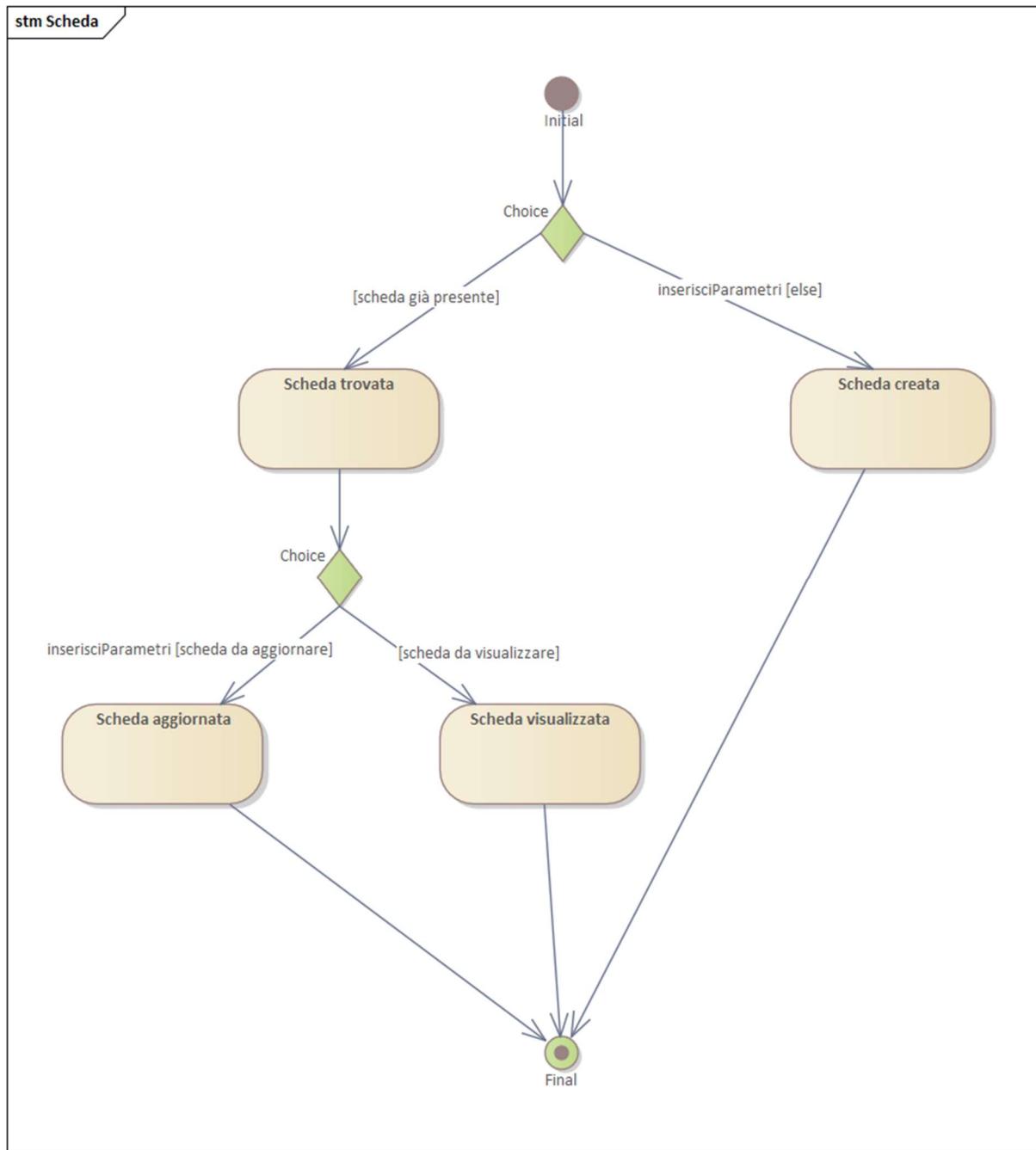
Appunti



Dado



Scheda





Utente

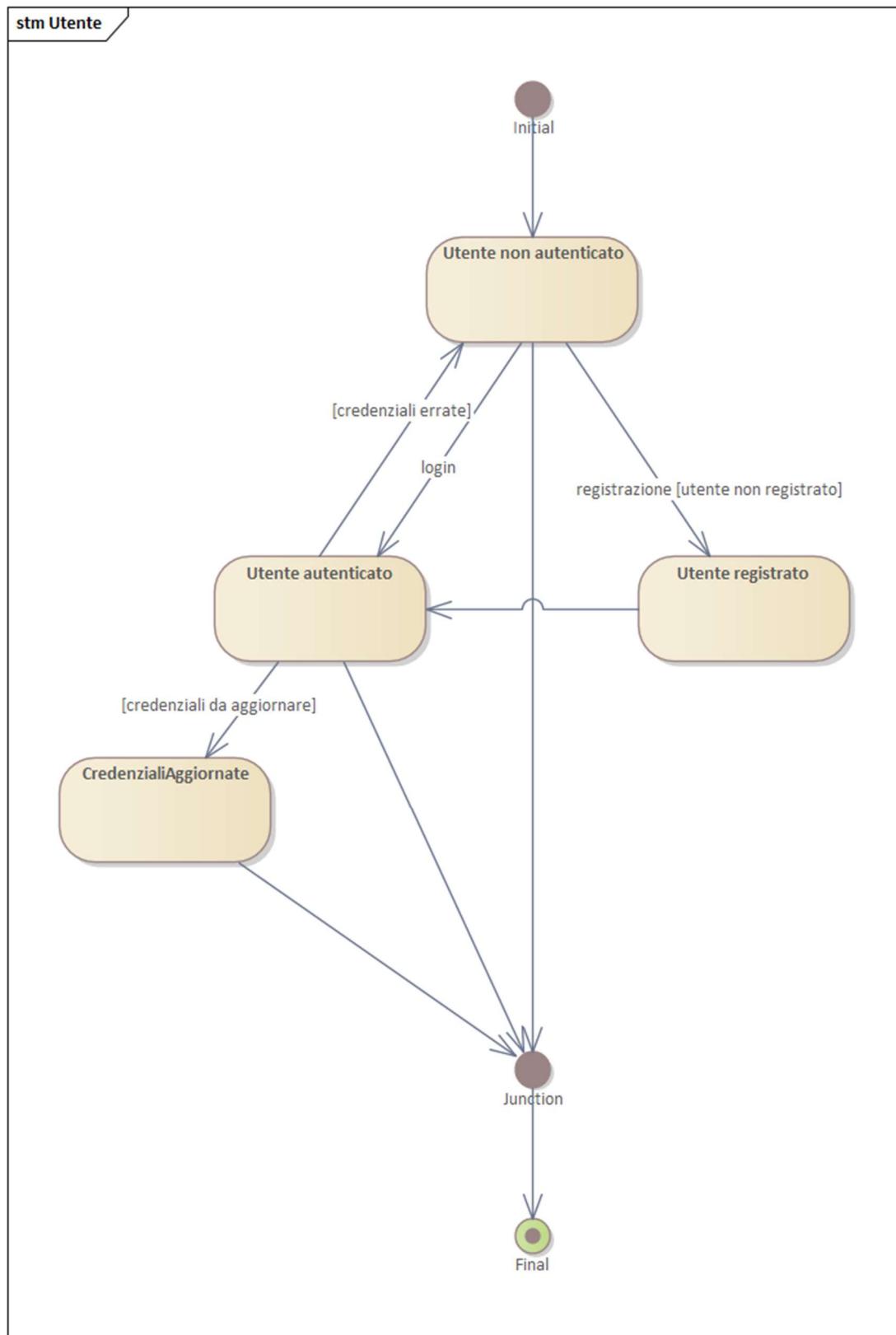
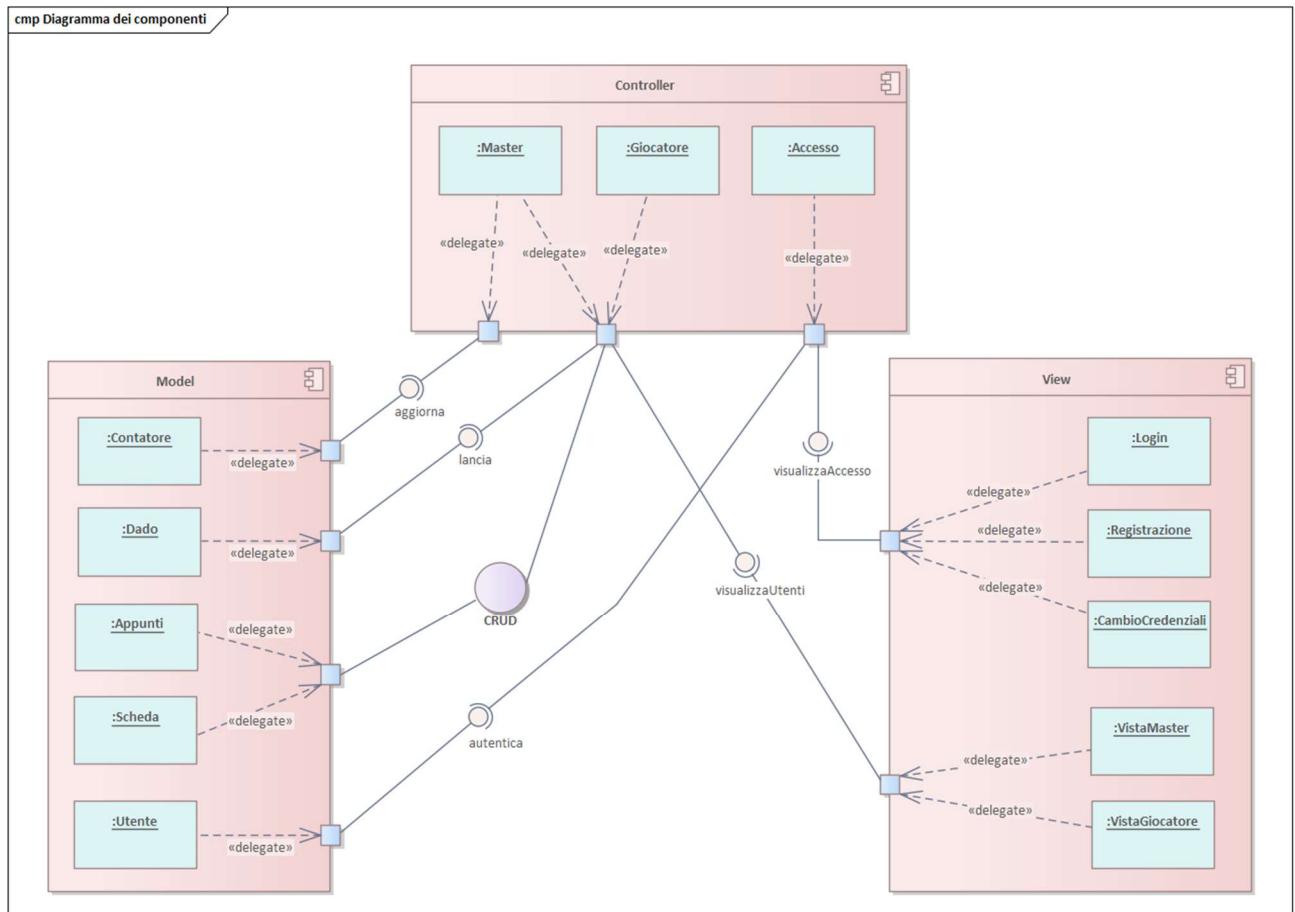


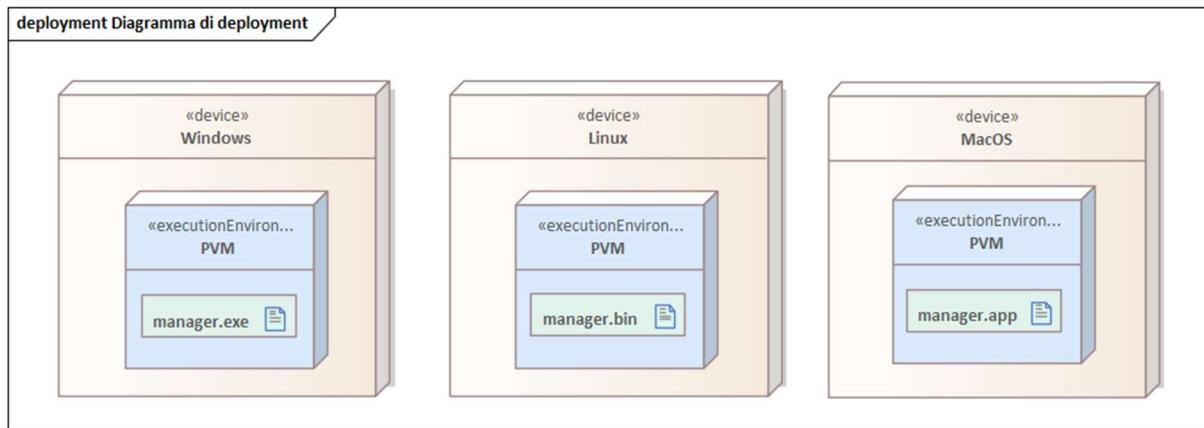


Diagramma dei componenti



Implementazione

Diagramma di deployment



Mockup

Registrazione



Registrazione

Inserire nome utente e password per accedere.

Nome Utente:

Password:

Master



DnD Manager

- □ X

Id: 7123

Ruolo: Master

Cambia Credenziali

Modifica PG

Visualizza PG

Crea Appunto

Visualizza Appunto

Modifica Appunto

Elimina Appunto

Tira i dadi

Genera Statistiche

Gestisci Contatore

Pubblica Appunti

Cambio Credenziali



?

X

Cambio Credenziali

Selezionare le nuove credenziali

Nuovo nome utente:

Nuova password:

Crea Scheda



Creazione Scheda

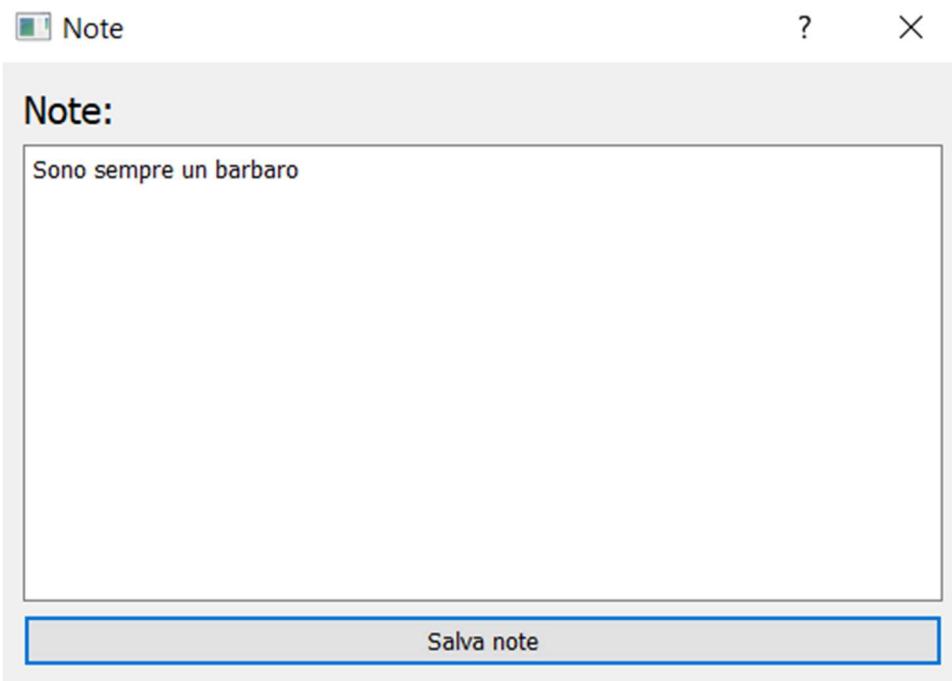
Inserisci i parametri del personaggio:

Nome	Grullo
Forza	5
Destrezza	4
Costituzione	5
Intelligenza	1
Saggezza	1
Carisma	1
Classe	Barbaro
Livello	1
Punti Ferita	77
Classe Armatura	14
Tiri Salvezza	Forza
	Costituzione
Abilità	Acrobazia
	Atletica

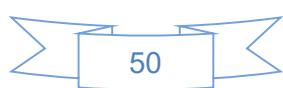
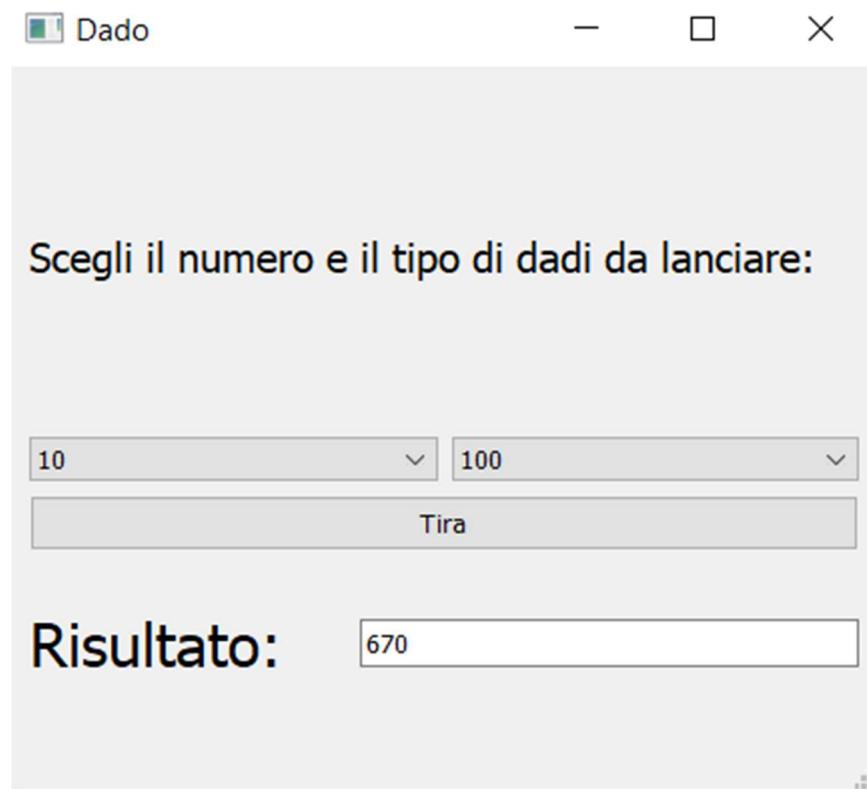
Storia personaggio

Sono un barbaro

Note



Tira dado



[Visualizza Appunto](#)

 Appunto

?

X



Lui è il boss finale

Grado sfida

99

Contatore



Contatore

?

X

Numero sessioni:

8

Incrementa

Resetta

Statistiche



Statistiche ? X

Forza media 2

Destrezza media 3

Costituzione media 3

Intelligenza media 1

Saggezza media 1

Carisma medio 1

Livello medio 1

Punti Ferita medi 47

Classe Armatura media 10

Test

PyUnit



Test Accesso

```
import unittest

from Controller import Accesso
from Model import Utente

class MyTestCase(unittest.TestCase):
    def testCaricaLista(self):
        accesso = Accesso.Accesso()

        self.assertTrue(acesso.caricaListaUtenti())

    def testLogin(self):
        accesso = Accesso.Accesso()

        self.assertIsInstance(acesso.login("nomeTest", "passwordTest"), type(Utente.Utente()))

        self.assertEqual(acesso.login("nomeTest", "passErrata"), "Password")

        self.assertEqual(acesso.login("nomeErrato", "passwordTest"), "Utente")

if __name__ == '__main__':
    unittest.main()
```

Test Giocatore



```
class TestControllerGiocatore(unittest.TestCase):
    def testCreaScheda(self):
        utente = Utente.Utente()
        controller = Giocatore.Giocatore(utente)

        controller.creaScheda("Test", utente.personaggio.punteggi, "Barbaro", 1, 12, 12, "Forza", "Destrezza",
                               "Acrobazia", "Percezione", "Storia Test")

        self.assertTrue(os.path.isfile("Schede/Test.pickle"))

    def testVisualizzaScheda(self):
        utente = Utente.Utente()
        controller = Giocatore.Giocatore(utente)

        self.assertIsInstance(controller.visualizzaScheda(), Scheda.Scheda)

    def testCaricaScheda(self):
        utente1 = Utente.Utente()
        utente2 = Utente.Utente()
        controller1 = Giocatore.Giocatore(utente1)
        controller2 = Giocatore.Giocatore(utente2)

        utente1.personaggio.setNome("Grullo")

        self.assertTrue(controller1.caricaScheda())
        self.assertFalse(controller2.caricaScheda())

if __name__ == '__main__':
    unittest.main()
```

Test Master



```
class TestControllerMaster(unittest.TestCase):
    def testCambioCredenziali(self):
        master = Utente.Utente()
        controller = Master.Master(master)

        master.setNomeUtente("Test")
        master.setPassword("Test")
        lista = [master]

        controller.accesso.listaUtenti = lista
        controller.accesso.salvaListaUtenti()

        controller.cambioCredenziali("nomeTest", "passwordTest")

        self.assertEqual(master.nomeUtente, "nomeTest")
        self.assertEqual(master.password, "passwordTest")

    def testContatore(self):
        master = Utente.Utente()
        controller = Master.Master(master)

        controller.caricaContatore()
        controller.incrementaContatore()
        controller.resettaContatore()
        controller.incrementaContatore()
        controller.incrementaContatore(0)

        self.assertEqual(controller.contatoreAttuale.contatore, 2)
```



```
def testDadi(self):
    master = Utente.Utente()
    controller = Master.Master(master)

    risultato = controller.tiraDado(4, 2)

    self.assertNotEqual(risultato, "9")

if __name__ == '__main__':
    unittest.main()
```