



Università degli Studi di Napoli “Federico II”
Scuola Politecnica e delle Scienze di Base

Corso di Laurea Magistrale in Ingegneria Informatica
Elaborato in Software Architecture Design

VIRTUAGYM



Candidati:

Antonio Borzillo

Pietro Carputo

Nicolò Discepolo

prof. Anna Rita Fasolino

Anno Accademico 2021/2022



Indice

Indice	2
Introduzione	4
Capitolo 1: Processo di sviluppo e tools adottati	5
1.1 Processo di sviluppo	5
1.2 Organizzazione del lavoro	7
1.2.1 Altre tecniche Agili	7
1.2.2 Scheduling	10
1.2.3 Tools utilizzati	11
1.2.4 Primo workshop	12
1.2.5 Secondo workshop	13
1.2.6 Terzo workshop	13
1.3 Stima dei costi	14
1.3.1 Elenco dei casi d'uso con relativa complessità stimata	14
1.3.2 Elenco degli attori con relativa complessità stimata	15
1.3.3 Unadjusted Use Case Points (UUCP)	16
1.3.4 Aggiustamenti per la Complessità Tecnica (requisiti non funzionali)	17
1.3.6 Calcolo Finale degli Use Case Points (UCP)	19
1.3.7 Ore di lavoro totali	19
1.3.8 Numero di iterazioni	20
Capitolo 2: Fase di avvio del progetto	21
2.1 Descrizione degli obiettivi	21
2.2 Analisi del testo	23
2.3 Tabella attori- obiettivi	25
2.4 Requisiti Funzionali in formato breve	26
2.5 Requisiti non Funzionali	27
2.5.1 Requisiti del prodotto	27
2.5.2 Requisiti organizzativi	27
2.5.3 Requisiti esterni	28
2.6 Vincoli generali	28
2.7 Diagramma dei casi d'uso	29
Capitolo 3: Analisi e specifica dei requisiti	31
3.1 Identificazione degli attori	31
3.2 Interfacce	32
3.2 Descrizione dettagliata dei casi d'uso	33
3.3 Diagramma di dominio	41
3.4 Diagramma di Sequenza di Analisi	42
3.4.1 Sequence Diagram – Effettua Login Socio	42

3.4.2 Sequence Diagram – Effettua Registrazione.....	43
3.4.3 Sequence Diagram – Visualizza Abbonamento.....	44
3.4.4 Sequence Diagram – Crea Abbonamento.....	45
3.4.5 Sequence Diagram – Visualizza Scheda.....	46
3.4.6 Sequence Diagram – Visualizza Info Palestra.....	47
3.4.7 Sequence Diagram – Gestione Corsi.....	48
3.4.8 Sequence Diagram – Gestione Scheda.....	49
3.4.9 Sequence Diagram – Iscrizione corso.....	50
3.5 Diagramma di attività.....	51
3.5.1 Activity Diagram – Effettua Login Socio.....	51
3.5.2 Activity Diagram – Effettua Registrazione.....	52
3.5.3 Activity Diagram – Visualizza Abbonamento.....	53
3.5.4 Activity Diagram – Crea Abbonamento.....	54
3.5.5 Activity Diagram – Visualizza Scheda.....	55
3.5.6 Activity Diagram – Visualizza Info Palestra.....	56
3.5.7 Activity Diagram – Gestione Corsi.....	57
3.5.8 Activity Diagram – Gestione Scheda.....	58
3.5.9 Activity Diagram – Iscrizione Corso.....	59
3.6 Diagramma delle classi.....	60
3.7 Diagramma di contesto.....	61
Capitolo 4: Architettura e scelte di progetto	62
4.1 Architettura logica.....	62
4.2 Design : Component Diagram.....	63
4.3 Pattern Utilizzati.....	65
4.3.1 MVC.....	65
4.3.2 repository.....	67
4.4 Scelte Progettuali.....	68
4.5 Framework utilizzati.....	69
4.6 Diagrammi Architeturali.....	73
4.6.1 Vista Architettuale – Cliente Autenticato.....	75
4.6.2 Vista Architettuale – Trainer.....	76
4.7 Diagrammi di sequenza raffinati.....	77
4.7.1 Sequence Diagram Raffinato – CreaScheda.....	77
4.7.2 Sequence Diagram Raffinato – EliminaScheda.....	78
Capitolo 5: Implementazione	80
5.1 Documentazione.....	80
5.2 Manuale per la configurazione e l’avvio.....	81
5.3 Esecuzione della web application.....	82
Capitolo 6: Testing	85
6.1 Test eseguiti.....	85
6.1 Testing iniziali.....	86
6.2 Test successivi.....	88
6.3 Test per le future iterazioni.....	88



Introduzione

Il progetto “VirtuaGym” è stato sviluppato nell’ambito del corso “Software Architecture Design” durante l’anno accademico 2021/2022.

L’idea progettuale nasce dall’esigenza di gestire i servizi offerti dal complesso sportivo di una palestra al fine di minimizzare, quanto più possibile le probabilità di contrarre il virus COVID-19 limitando le interazioni fisiche all’interno del complesso.

L’obiettivo principale del sistema è garantire la possibilità all’utente di interfacciarsi, tramite un applicativo web, per reperire informazioni generali e personali. In particolare, il socio sarà in grado di:

- a) consultare le informazioni generali della palestra;
- b) effettuare una prenotazione per l’accesso alla struttura in una determinata data in base all’eventuale disponibilità;
- c) verificare lo stato del proprio abbonamento;
- d) visualizzare la scheda assegnata per l’allenamento personalizzato.

Il sistema consente inoltre la gestione logistica e amministrativa da parte degli addetti del complesso sportivo: il segretario e il trainer. In particolare, il trainer può gestire le schede dei soci e il segretario può:

- a) gestire i corsi;
- b) gestire gli abbonamenti dei soci.



Capitolo 1: Processo di sviluppo e tools adottati

1.1 Processo di sviluppo

Il processo di sviluppo adottato per la realizzazione del sistema “VirtuaGym” è UP (Unified Process). Esso si basa sul raffinamento e perfezionamento del sistema con il susseguirsi di molteplici iterazioni, con continui feedback e adattamenti ciclici. UP può essere personalizzato a seconda dello specifico progetto. L’adozione di tale framework permette in particolare di gestire opportunamente le modifiche dovute al cambiamento dei requisiti in seguito a rilasci incrementali del software e consente, inoltre, di ridurre il rischio di fallimento, la complessità dell’implementazione e i costi di progetto.

Nel processo di sviluppo è stato impiegato il Rational Unified Process (RUP) che si configura come una estensione di UP e costituisce un modello di sviluppo del software di tipo iterativo, sviluppato da Rational Software (oggi parte di IBM). Il RUP non definisce un singolo, specifico processo, bensì un framework adattabile che può dar luogo a diversi processi in diversi contesti (per esempio in diverse organizzazioni o nel contesto di progetti con diverse caratteristiche). Il processo è costituito da 4 fasi:

- 1) **Ideazione** – identificazione dei principali casi d’uso e delle priorità relative; analisi di fattibilità e una stima di costi e tempi.
- 2) **Elaborazione** - implementazione iterativa del nucleo dell’architettura. La fase di elaborazione definisce la struttura complessiva del sistema. Questa fase comprende l’analisi di dominio e una prima fase di progettazione dell’architettura.

3) **Costruzione** - implementazione iterativa degli elementi rimanenti. In questa fase viene portato a termine la gran parte dello sviluppo e viene prodotta la prima release del sistema.

4) **Transizione** - testing e successivo rilascio del software. Il sistema passa dall'ambiente di sviluppo a quello del cliente finale.

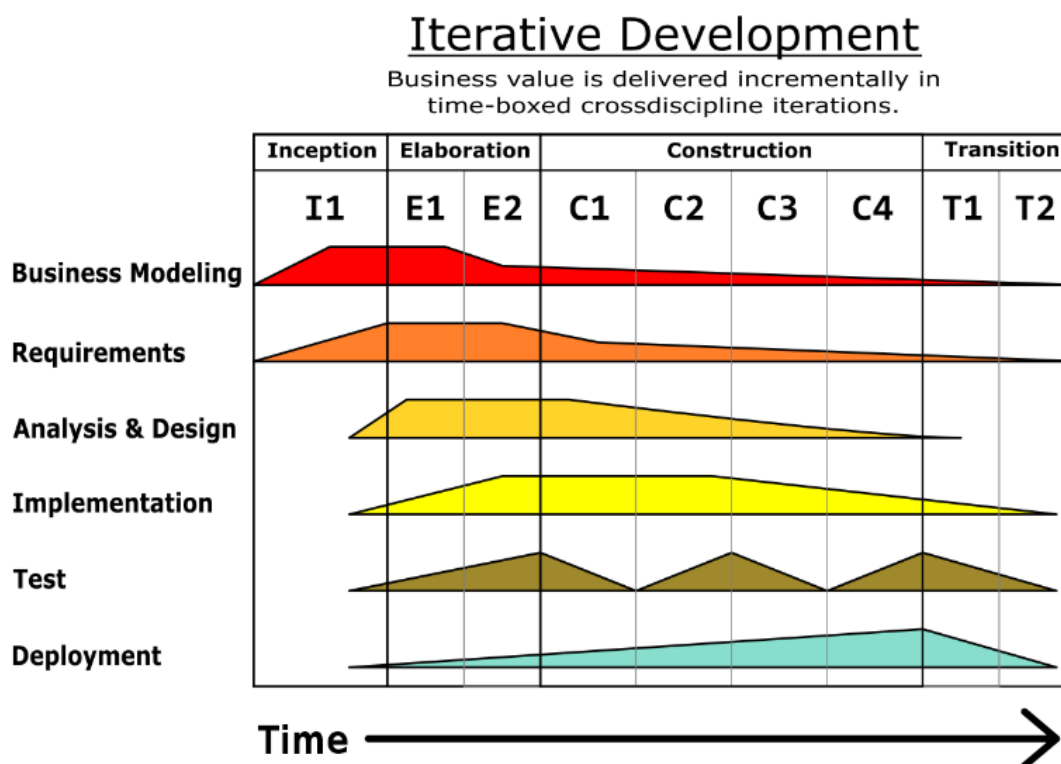


Fig. 1 Fasi del processo UP

Ogni fase è caratterizzata da un insieme di obiettivi e si conclude con la realizzazione di uno (o più) specifico *deliverable* (prodotto). Le fasi sono ulteriormente scomposte in *iterazioni*, che sono associate a periodi temporali e hanno scadenze precise.

I principali motivi che hanno indotto la scelta di RUP sono i seguenti:

- è possibile applicare UML per la modellazione agile;
- è un metodo iterativo e flessibile, che permette al team di lavorare attraverso iterazioni di durata non prestabilita e quindi di sentirsi meno vincolato a pratiche specifiche;

- prevede una serie di iterazioni time-boxed che possono essere viste come veri e propri mini-progetti che forniscono qualcosa di funzionante anche nel caso in cui non si riesca a terminare l'intero progetto entro una data di consegna predefinita;
- il codice è pensato per il cambiamento ed è quindi molto flessibile e riusabile.

Nel processo di sviluppo sono state anche adottate specifiche tecniche di configurazione/integrazione di software che hanno consentito il riuso di componenti software sviluppate per altri contesti applicativi. Tali componenti sono state gestite come “Black Box” e configurate/integrate tramite interfacce. In particolare, sono state adottati:

- 1) Spring Security come framework per la gestione della sicurezza;
- 2) JPA (Java Persistence API) come interfacce per la gestione della persistenza dei dati.

1.2 Organizzazione del lavoro

Il team di sviluppo è composto da 3 studenti che hanno lavorato al progetto per una durata complessiva di 7 settimane. Durante tale periodo di tempo sono state effettuate 3 iterazioni che hanno consentito di rilasciare la versione 1.0 del sistema a seguito di vari raffinamenti successivi.

Le sessioni di sviluppo hanno visto la partecipazione in presenza di tutti i membri del team con saltuarie occasioni di meeting online attraverso diverse piattaforme dedicate.

1.2.1 Altre tecniche Agili

- **Daily meeting** – incontri giornalieri in remoto per lo sviluppo del progetto tramite strumenti per il meeting, quali Microsoft Teams e Skype.

- **Refactoring**- tecnica strutturata per modificare la struttura interna di porzioni di codice senza modificarne il comportamento esterno; applicata per migliorare alcune caratteristiche non funzionali del software.
- **Continui workshop** – sessioni di lavoro collettive finalizzate al controllo dei requisiti già definiti ed eventualmente al loro aggiornamento/modifica.
- **Pair programming** - tecnica nella quale due programmatori lavorano insieme alla stessa postazione di lavoro. Il primo scrive il codice e l'altro svolge il ruolo di supervisore.

In media, le sessioni hanno avuto luogo 5 giorni a settimana per 6 ore giornaliere effettive che corrispondono ad un totale di 30 ore settimanali e 210 complessive per ogni membro del team.









Le diverse fasi di attività sono dettagliate nella seguente tabella.

Fase	Durata	Iterazioni	Attività	Milestones
Ideazione	1 settimana	1	Analisi di fattibilità (stima dei costi e tempi di sviluppo) con esplorazione generale dei requisiti	Specificazione dei requisiti funzionali e NON, individuazione dei principali casi d'uso, piano di iterazione.
Elaborazione	2 settimane	3	Analisi e specifica dei requisiti, implementazione, definizione della struttura complessiva del sistema e sviluppo del prototipo architetturale	Modello di dominio, Diagrammi di progettazione, documento di architettura, Prototipo funzionante.
Costruzione	2 settimane	3	Implementazione iterativa degli elementi rimanenti e raffinamento dei diagrammi.	Modello di dominio, Diagrammi di progettazione, documento di architettura, Prototipo funzionante.
Transizione	1 settimana	3	Fase di test e completamento funzionale.	Testing e rilascio della versione 1.0 del software

Per tener traccia del lavoro svolto quotidianamente è stato utilizzato Smartsheet: una web Platform, utilizzata specialmente in ambito agile, che aiuta i team nella divisione e coordinazione del lavoro.



1.2.3 Tools utilizzati

Logo	Tools	Funzione
	Microsoft Team	Gestione unificata delle comunicazioni/collaborazioni attraverso chat di lavoro, teleconferenza, condivisione di contenuti e integrazione delle applicazioni.
	Visual Paradigm	Modellazione in UML
	Google doc	Elaborazione testi, fogli elettronici, presentazioni
	Eclipse	Sviluppo di applicazioni
	Spring Tool Suite 4	Sviluppo di applicazioni aziendali basate su Spring per Eclipse
	MySQL	Gestione di database relazionali
	GitHub	Memorizzazione, aggiornamento e condivisione di documenti
	SmartSheet	Pianificazione e organizzazione del lavoro

1.2.4 Primo workshop

Durante il primo workshop dei requisiti il team si è riunito al fine di individuare e classificare i casi d'uso caratterizzanti il sistema “VirtuaGym”. Tra questi, il caso d'uso “Effettua iscrizione al Corso” è stato scelto come principale in quanto risulta essere significativo rispetto all'architettura e contraddistinto da elevato valore di business e rischio.

Il team ha stabilito gli obiettivi della prima iterazione ponendo particolare attenzione sui requisiti che avrebbero potuto garantire un funzionamento corretto del sistema nel minor tempo possibile, disaccoppiandoli da quelli implementabili in un secondo momento. Come conseguenza di ciò, il primo caso d'uso implementato è stato “Effettua Login e Registrazione”.



Fig. 4. Analisi e Specifica dei Requisiti

1.2.5 Secondo workshop

Al termine della prima iterazione, il team ha effettuato una retrospettiva al fine di valutare il lavoro svolto fino a quel momento, verificando se gli obiettivi della prima iterazione fossero stati raggiunti e se fosse necessario apportare dei miglioramenti. Inoltre, a seguito di un affinamento del lavoro del primo workshop, vi è stata la pianificazione con conseguente implementazione dei casi d'uso "Gestione Scheda" e "Login e Registrazione".

1.2.6 Terzo workshop

Nel terzo ed ultimo workshop, dopo un ulteriore affinamento, si è scelto di descrivere il caso d'uso "Effettua iscrizione al Corso", con annessa fase di implementazione e testing.



Fig. 5. Pair Programming

1.3 Stima dei costi

Lo sviluppo ha previsto una fase iniziale in cui è stata discussa e poi definita l'idea da realizzare in gruppo. Inoltre in questa fase, si è valutata l'effettiva realizzabilità del progetto, principalmente in relazione al tempo, ai costi e al lavoro necessari. Per prevedere la dimensione del software e di conseguenza stimarne i tempi e i costi, indipendentemente dal team di sviluppo, è stato utilizzato il metodo degli Use Case Points.

1.3.1 Elenco dei casi d'uso con relativa complessità stimata

Caso d'uso	Complessità
VisualizzaInfoPalestra	Semplice
EffettuaPrenotazione	Complessa
VisualizzaScheda	Semplice
IscrizioneCorso	Media
GestioneScheda	Media
GestioneCorso	Semplice
VisualizzaAbbonamento	Semplice
EliminaAbbonamento	Semplice
RinnovaAbbonamento	Semplice
CreaAbbonamento	Semplice
EffettuaPagamento	Semplice
IscrizionePalestra	Semplice
RichiediRinnovoAbbonamento	Semplice

Complessità	Peso	Numero casi d'uso	Prodotto
Semplice	5	9	45
Media	10	2	20
Complessa	15	2	30
Total (UUCW)			95

1.3.2 Elenco degli attori con relativa complessità stimata

Tipo di attore	Peso	Numero di attori	Prodotto
Semplice	1	1	1
Media	2	0	0
Complessa	3	4	12
Total (UAW)			13

Tipi di attori:

Tipo	Esempio	Peso
Semplice	altro sistema tramite API	1
Medio	altro sistema tramite protocollo oppure persona tramite interfaccia testuale	2
Complesso	persona tramite interfaccia grafica	3

Attoreo	Tipo
UtenteGenerico	3
Socio	3
Segretario	3
Trainer	3
SistGestCassa	1

1.3.3 Unadjusted Use Case Points (UUCP)

Come primo parametro è stato calcolato l'UUCP (Unadjusted Use Case Points), dato dalla somma degli UUCW (Unadjusted Use Case Weight) e UAW (Unadjusted Actor Weight):

$$UUCP = UUCW + UAW = 95 + 13 = 108$$

1.3.4 Aggiustamenti per la Complessità Tecnica (requisiti non funzionali)

A questo punto è stato determinato il fattore di complessità tecnica, meglio noto come TFactor, attribuendo ad ognuno dei tredici fattori un punteggio indicativo della rilevanza degli stessi: 0 irrilevante - 5 molto importante.

Fattore	Peso	Valutazione	Impatto
Sistema distribuito	2	2	4
Obiettivi di performance	2	2	4
Efficienza end-user	1	4	4
Complessità di elaborazione	1	2	2
Riusabilità del codice	1	3	3
Facilità di installazione	0.5	4	2
Facilità di uso	0.5	4	2
Portabilità	2	1	2
Modificabilità	1	2	2
Uso concorrente	1	3	3
Sicurezza	1	3	3
Accesso per terze parti	1	2	2
Necessità di formazione	1	1	1
Totale (TFactor)			34

Nota: gli assessment sono stati assegnati cercando sì di avere un buon software, ma anche di venire incontro al prezzo che il cliente è disposto a spendere per il software.

Ad esempio sarebbe ovviamente meglio avere un software portatile, ma questo comporterebbe una lievitazione del prezzo maggiore rispetto al far ottenere al cliente una macchina con Windows installato piuttosto che altri OS.

$$TFC = 0.6 + (0.01 \times TFactor) = 0.6 + (0.01 \times 34) = 0.94$$

1.3.5 Aggiustamento per la complessità dell'ambiente (ambiente di lavoro)

Fattore	Peso	Valutazione	Impatto
Familiarità con processo di sviluppo	1.5	1	1.5
Esperienza applicativa	0.5	1	0.5
Esperienza orientata ad oggetti	1	2	2
Capacità di condurre analisi	0.5	2	1
Motivazione	1	5	5
Requisiti stabili	2	3	6
Staff part-time	-1	0	0
Difficoltà linguaggio di programmazione	-1	0	0
Totale (EFactor)			16

Successivamente, è stato ricavato l'EFactor, meglio noto come Environmental Factor:

$$EF = 1.4 + (-0.03) \times Efactor = 1.4 + (-0.03 \times 16) = 1.4 - 0.48 = 0.92$$

1.3.6 Calcolo Finale degli Use Case Points (UCP)

Infine, una volta ricavati il TFC e l'EF possiamo individuare gli Use Case Point, moltiplicando gli UUCP per i due fattori corretti:

$$UCP = UUCP \times TFC \times EF = 108 \times 0.94 \times 0.92 \cong 93.39$$

1.3.7 Ore di lavoro totali

Supponendo che un singolo UC venga sviluppato in 20-28 ore, noi scegliamo il valore medio di 24. un rapporto di 20 ore per UCP (Metodo Karner), otteniamo un quantitativo di ore pari a:

$$Ore\ totali = UCP \times 24 = 93.39 \times 24 = 2241\ h$$

1.3.8 Numero di iterazioni

Numero iterazioni:

- 1 lavoratore = 30 ore a settimana*
- 3 lavoratori = 90 ore a settimana
- ore di lavoro totale / ore a settimana del gruppo di lavoro = $2241 / 90 = 25$

Quindi si stima che occorrano 25 settimane = 25 iterazioni per sviluppare questo progetto nella sua interezza.

Nota: si sta supponendo che uno sviluppatore lavori 30 ore a settimana sul progetto come riportato sulle slide.



Capitolo 2: Fase di avvio del progetto

2.1 Descrizione degli obiettivi

Nella fase iniziale dell'avvio del progetto sono stati stabiliti gli obiettivi principali del sistema "VirtuaGym".

Si vuole realizzare un'applicazione web per la gestione dei servizi offerti ai soci di una palestra. Il centro sportivo dispone di alcune sale attrezzate per l'allenamento individuale e sale dedicate allo svolgimento di corsi sportivi impartiti da trainer professionisti. In ciascuna sala è stabilito un numero massimo di soci (identificati tramite nome, cognome, codice fiscale, cellulare, email, data nascita) che possono essere presenti contemporaneamente (causa Covid).

Un utente generico che si collega al sito web può visualizzare la struttura del centro, i corsi offerti e i piani tariffari. Per diventare socio della palestra, un utente deve fare richiesta di iscrizione specificando i propri dati personali (dati anagrafici, email, codice fiscale, recapiti telefonici), fornendo il certificato medico e indicando la tipologia di abbonamento che intende sottoscrivere. All'atto di tale richiesta, il pagamento della quota prevista, effettuabile tramite carta di credito o altri servizi bancari, viene gestito da un sistema esterno. Al contempo, il segretario attiva la procedura per la creazione dell'abbonamento a nome dell'utente che ne ha fatto richiesta memorizzandone i dati e rilasciando al socio le credenziali di accesso.

Il socio può richiedere il rinnovo confermando/modificando la tipologia dell'abbonamento. All'atto di tale richiesta, non è più necessario specificare i propri dati personali. Il pagamento della quota prevista, effettuabile tramite carta di credito o altri servizi bancari, viene gestito dal sistema esterno. Al contempo, il segretario attiva la procedura per il rinnovo dell'abbonamento a nome del socio che ne ha fatto richiesta.

Dopo aver effettuato la procedura di autenticazione utilizzando le proprie credenziali, l'applicazione web consente al socio di visualizzare le informazioni relative all'affluenza della palestra in tempo reale, effettuare la prenotazione per l'accesso ad una (o più) sale attrezzate della palestra per svolgere il proprio allenamento individuale, consultare la scheda di allenamento, visualizzare le informazioni relative allo stato del proprio abbonamento. Il socio può inoltre iscriversi ad un corso previsto nel calendario secondo disponibilità. Il socio può anche annullare l'iscrizione ad un corso se non intende più parteciparvi.

Oltre che la creazione/rinnovo di un abbonamento, il segretario può visualizzare tutti gli abbonamenti attivi ed eliminare l'abbonamento scaduto di un socio. Inoltre, il segretario è responsabile della gestione del calendario dei corsi e può effettuare operazioni di creazione, visualizzazione, modifica ed eliminazione.

Il trainer ha la responsabilità di gestire la scheda di ciascun socio e deve poter visualizzare, modificare, creare, eliminare una scheda mantenendola aggiornata in base ai progressi sportivi del socio.

2.2 Analisi del testo

Per definire il dominio del sistema, è stata effettuata l'analisi testuale dei requisiti informali iniziali attraverso il relativo Tool disponibile in Visual Paradigm. Tale analisi ha permesso di evidenziare gli attori, i casi d'uso e le classi coinvolte nel sistema e di determinarne il numero di occorrenze nel testo.

Si vuole realizzare un'applicazione web per la gestione dei servizi offerti ai soci di una palestra. Il centro sportivo dispone di alcune sale attrezzate per l'allenamento individuale e sale dedicate allo svolgimento di corsi sportivi impartiti da **trainer** professionisti. In ciascuna sala è stabilito un numero massimo di soci (identificati tramite nome, cognome, codice fiscale, cellulare, email, data nascita) che possono essere presenti contemporaneamente (causa Covid).

Un **utente generico** che si collega al sito web può **visualizzare la struttura del centro, i corsi offerti e i piani tariffari**. Per diventare **socio** della palestra, un utente deve fare **richiesta di iscrizione** specificando i propri dati personali (dati anagrafici, email, codice fiscale, recapiti telefonici), fornendo il **certificato medico** e indicando la tipologia di **abbonamento** che intende sottoscrivere. All'atto di tale richiesta, il **pagamento** della quota prevista, effettuabile tramite carta di credito o altri servizi bancari, viene gestito da un **sistema esterno**. Al contempo, il **segretario** attiva la procedura per la **creazione dell'abbonamento** a nome dell'utente che ne ha fatto richiesta memorizzandone i dati e rilasciando al **socio** le credenziali di accesso.

Il **socio** può **richiedere il rinnovo** confermando/modificando la tipologia dell'abbonamento. All'atto di tale richiesta, non è più necessario specificare i propri dati personali. Il **pagamento** della quota prevista, effettuabile tramite carta di credito o altri servizi bancari, viene gestito dal **sistema esterno**. Al contempo, il **segretario** attiva la procedura per il **rinnovo dell'abbonamento** a nome del **socio** che ne ha fatto richiesta.

Dopo aver effettuato la procedura di **autenticazione** utilizzando le proprie credenziali, l'applicazione web consente al **socio** di **visualizzare le informazioni** relative all'affluenza della palestra in tempo reale, effettuare la **prenotazione** per l'accesso ad una (o più) sale attrezzate della palestra per svolgere il proprio allenamento individuale, **consultare la scheda di allenamento**, **visualizzare le informazioni relative allo stato del proprio abbonamento**. Il **socio** può inoltre **iscriversi ad un corso** previsto nel calendario secondo disponibilità. Il **socio** può anche **annullare l'iscrizione ad un corso** se non intende più parteciparvi.

Oltre che la creazione/rinnovo di un **abbonamento**, il **segretario** può **visualizzare tutti gli abbonamenti** attivi ed **eliminare l'abbonamento scaduto** di un **socio**. Inoltre, il **segretario** è responsabile della **gestione del calendario dei corsi** e può effettuare operazioni di creazione, visualizzazione, modifica ed eliminazione.

Il **trainer** ha la responsabilità di **gestire la scheda** di ciascun **socio** e deve poter visualizzare, modificare, creare, eliminare una scheda mantenendola aggiornata in base ai progressi sportivi del **socio**.

Fig.6 Analisi testuale mediante Visual Paradigm

No.	Candidate Class	Extracted Text	Type	Occurren...	Highlight
1	trainer	trainer	Actor	2	
2	sistema esterno	sistema esterno	Actor	2	
3	prenotazione	prenotazione	Class	1	
4	abbonamento	abbonamento	Class	2	
5	certificato medico	certificato medico	Class	1	
6	corso	corso	Class	0	
7	richiedere il rinnovo	richiedere il rinnovo	Use Case	1	
8	richiesta di iscrizione	richiesta di iscrizione	Use Case	1	
9	pagamento	pagamento	Use Case	2	
10	creazione dell'abbonamento	creazione dell'abbonamento	Use Case	1	
11	rinnovo dell'abbonamento	rinnovo dell'abbonamento	Use Case	1	
12	autenticazione	autenticazione	Use Case	1	
13	gestire la scheda	gestire la scheda	Use Case	1	
14	visualizzare tutti gli abbonamenti	visualizzare tutti gli abbonamenti	Use Case	1	
15	eliminare l'abbonamento scaduto	eliminare l'abbonamento scaduto	Use Case	1	
16	gestione del calendario dei corsi	gestione del calendario dei corsi	Use Case	1	
17	consultare la scheda di allenamento	consultare la scheda di allenamento	Use Case	1	
18	visualizzare le informazioni relative allo stato del proprio abbonamento	visualizzare le informazioni relative allo stato del proprio abbonamento	Use Case	1	
19	visualizzare la struttura del centro, i corsi offerti e i piani tariffari	visualizzare la struttura del centro, i corsi offerti e i piani tariffari	Use Case	1	
20	iscriversi ad un corso	iscriversi ad un corso	Use Case	1	
21	annullare l'iscrizione ad un corso	annullare l'iscrizione ad un corso	Use Case	1	
22	utente generico	utente generico	Actor	1	
23	segretario	segretario	Actor	4	
24	socio	socio	Actor	10	

Fig.7 Analisi testuale mediante Visual Paradigm - Tabella

2.3 Tabella attori- obiettivi

Attore	Obiettivi
Utente Generico	Visualizzare informazioni palestra Richiedere nuovo abbonamento
Socio	Visualizzare l'abbonamento Visualizzare la scheda Effettuare prenotazione Effettuare iscrizione ad un corso Annullare iscrizione ad un corso Richiedere rinnovo abbonamento
Segretario	Creare nuovo abbonamento Socio Rinnovare abbonamento Socio Eliminare abbonamento socio Visualizzare gli abbonamenti dei soci Gestire i corsi
Trainer	Gestire le schede dei soci
Sistema esterno gestione cassa	Gestire il pagamento

2.4 Requisiti Funzionali in formato breve

Attore	Caso d'uso	Descrizione
Utente generico	Effettua registrazione	Gli utenti generici possono registrarsi nel sistema fornendo le credenziali: nome, cognome, email, password.
Utente generico/ Socio	Visualizza informazioni palestra	Tutti gli utenti possono visualizzare le informazioni della palestra riguardanti i corsi offerti, le aree di allenamento e le tipologie di abbonamento con le relative tariffe offerte.
Socio	Effettua login	Il socio può accedere al sistema fornendo le proprie credenziali di email e password
	Effettua prenotazione	Il socio può prenotare una seduta di allenamento in data/orario disponibile
	Effettua iscrizione	Il socio può iscriversi ad uno dei corsi organizzati
	Annulla iscrizione	Il socio può annullare l'iscrizione effettuata
	Visualizza scheda	Il socio può visualizzare gli esercizi con relative serie e ripetizioni assegnategli dal trainer per il proprio allenamento personalizzato
	Iscrizione Palestra	Il socio può chiedere di abbonarsi
	Richiedi rinnovo abbonamento	Il socio può chiedere di rinnovare il proprio abbonamento
	Visualizza abbonamento	Il socio può visualizzare lo stato del proprio abbonamento con le relative date di inizio/fine.
Segretario	Crea nuovo abbonamento socio	Il segretario, a seguito della convalida del pagamento, crea un abbonamento per il socio inserendo i relativi dati
	Rinnova abbonamento socio	Il segretario, a seguito della convalida del pagamento, rinnova l'abbonamento di un socio recuperando i relativi dati
	Visualizza abbonamento dei soci	Il segretario può visualizzare l'abbonamento dei soci iscritti alla palestra
	Elimina abbonamento socio	Il segretario può eliminare l'abbonamento di un socio
	Gestisci i corsi	Il segretario può gestire i corsi inserendo/modificando/cancellando le relative informazioni
Trainer	Gestisci la scheda dei soci	Il trainer può gestire la scheda di ciascun socio inserendo/modificando/cancellando le relative informazioni

2.5 Requisiti non Funzionali

2.5.1 Requisiti del prodotto

- **Requisiti di usabilità** - Il sistema deve essere, nelle sue caratteristiche fondamentali, facilmente intuibile e comprensibile per l'utente finale; senza tralasciare, al tempo stesso, aspetti come la attrattività e conformità generale.
- **Requisiti di affidabilità** - Il sistema deve garantire una costante disponibilità di accesso per gli utenti risultando, inoltre, utilizzabile in concorrenza. Un aspetto importante è il tempo di risposta, ovvero la quantità totale di tempo necessaria per rispondere a una richiesta di servizio.
- **Requisiti di portabilità** - Il sistema deve essere facilmente raggiungibile via web con l'ausilio di un generico browser da molteplici dispositivi (PC) dotati di una connessione internet.
- **Requisiti di evolvibilità** - Per ogni funzionalità offerta dal sistema, si garantisce il disaccoppiamento tra applicazione e servizio. Ciò permette una facile modifica dell'applicazione qualora fosse necessario cambiare i fornitori dei servizi.
- **Requisiti di persistenza** - I dati relativi alle diverse entità, sono memorizzate in modo persistente su database, in modo da evitare la perdita degli stessi.

2.5.2 Requisiti organizzativi

- **Requisiti di consegna** - Deve essere consegnata una demo preliminare.
- **Requisiti di implementazione** - Il sistema deve essere implementato in linguaggio JAVA, sviluppato per essere una applicazione web.

2.5.3 Requisiti esterni

- **Requisiti di riservatezza** - Ogni attore coinvolto può visionare specifiche aree di competenza del sistema. Il segretario può accedere ai dati dei soci relativi a prenotazioni e abbonamenti ma non alle schede degli stessi. Può, inoltre visionare le informazioni relative ai trainers verificando la loro disponibilità ai rispettivi corsi. I trainers possono visualizzare le informazioni relative ai certificati medici degli utenti al fine di formulare la scheda personalizzata. I trainers, come i soci iscritti, possono visualizzare lo stato delle prenotazioni ai corsi organizzati.
- **Requisiti di sicurezza** - Ogni attore del sistema, escluso l'utente generico che non esegue operazioni critiche, deve autenticarsi per poter accedere alle funzionalità offerte dall'applicazione. In questo modo è possibile garantire la confidenzialità e l'integrità dei dati. Le password dei clienti devono essere cifrate.

2.6 Vincoli generali

Il sistema è una applicazione web che girerà su un server in locale (Tomcat 9.0) in un Personal Computer avente sistema operativo Windows 10.

Tutti gli attori devono necessariamente autenticarsi per usufruire delle funzionalità del sistema, ad eccezione dell'attore Utente Generico il quale può registrarsi e visualizzare le informazioni generali della palestra senza autenticarsi.

2.7 Diagramma dei casi d'uso

Il diagramma dei casi di Fig. 8 mostra i principali servizi del sistema.

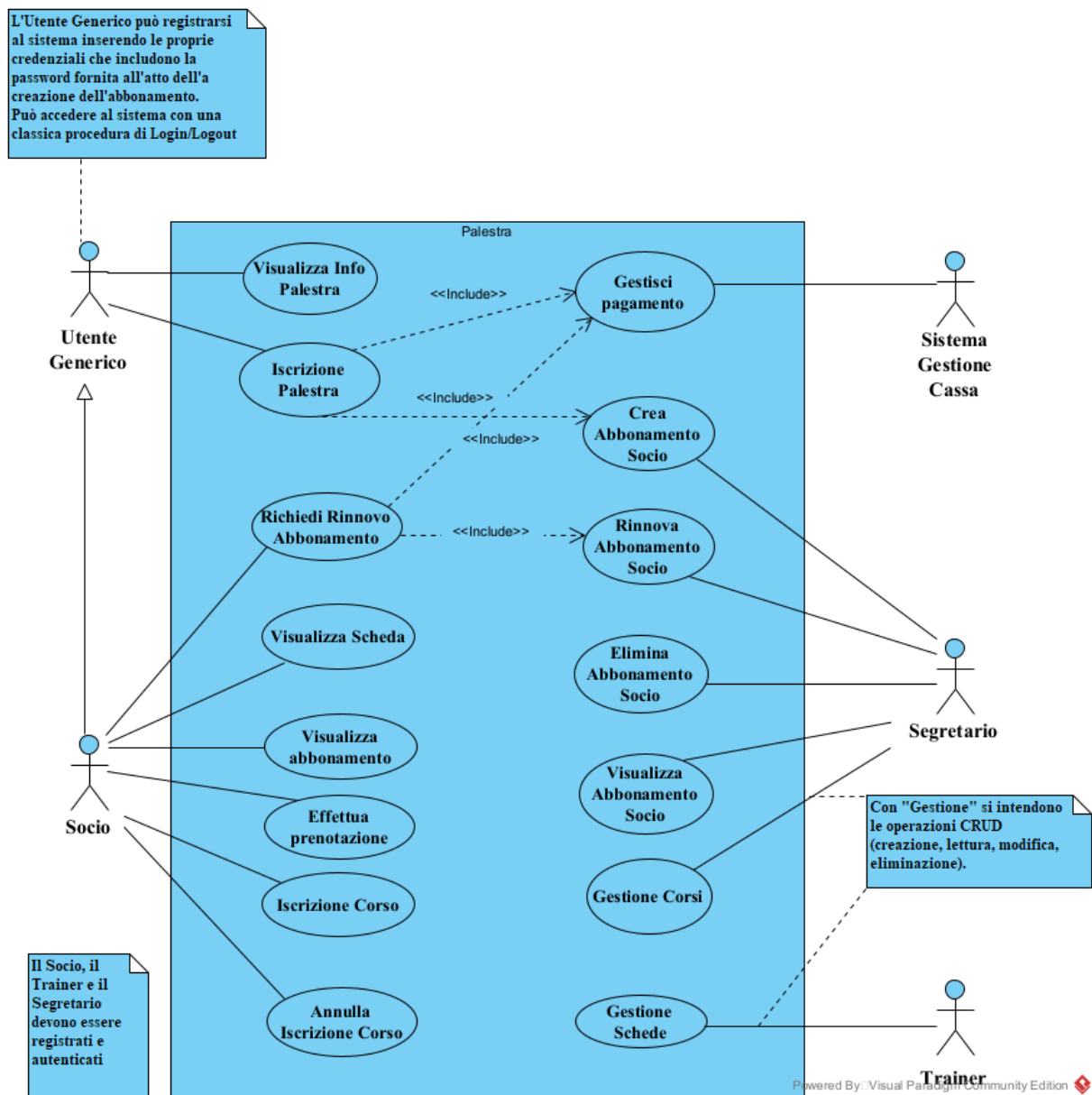


Fig.8 Diagramma dei casi d'uso

I casi d'uso “Gestione Pagamento” e Crea Abbonamento Socio” sono inclusi in “Iscrizione Palestra”. Ciò va a precisare che un Utente Generico, all’atto dell’ iscrizione ala palestra,

dovrà effettuare il pagamento della quota prevista e , al contempo, il segretario si occuperà della creazione dell'abbonamento immettendo i relativi dati.

Allo stesso modo, i casi d'uso “Gestione Pagamento” e Rinnova Abbonamento Socio” sono inclusi in “Richiedi Rinnovo Abbonamento”. Ciò va a precisare che un Socio, all'atto della richiesta di rinnovo del proprio abbonamento, dovrà effettuare il pagamento della quota prevista e , al contempo, il segretario si occuperà della estensione della durata del preesistente abbonamento senza il bisogno di immettere nuovamente i dati già in possesso.



Capitolo 3: Analisi e specifica dei requisiti

3.1 Identificazione degli attori

- **UTENTE GENERICO** - può visualizzare le informazioni generali della specifica palestra (corsi, piani tariffari e aree disponibili) oppure accedere al sistema se possiede le credenziali di accesso.



- **SOCIO** - utilizza il sistema per effettuare le prenotazioni in palestra oltre che visualizzare le informazioni personali relative al proprio abbonamento ed alla scheda personalizzata. Può, inoltre, iscriversi ad uno dei corsi organizzati in caso di disponibilità.



- **TRAINER** - interagisce con il sistema. Egli può infatti creare una scheda associata ad un socio, in base ai suoi dati personali, modificandola opportunamente in base ad esplicita richiesta del cliente.



- **SEGRETARIO** - gestisce il sistema in quanto ha l'onere di creare o modificare abbonamenti dei soci, oltre che calendarizzare la attività relative ai corsi organizzati.



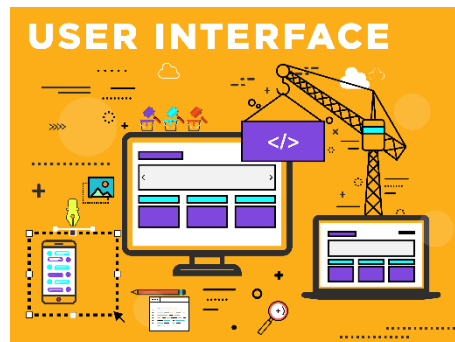
Il sistema prevede come attore di supporto:

SISTEMA GESTIONE CASSA -

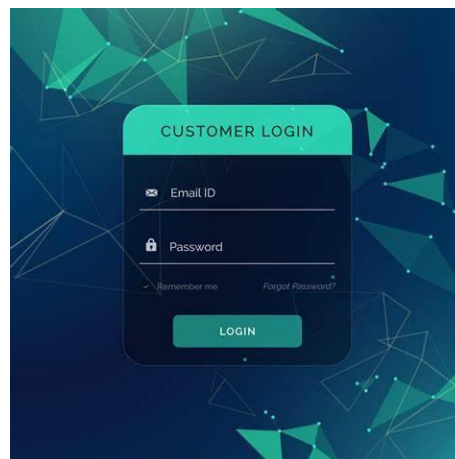
3.2 Interfacce



- Interfaccia per accedere e visualizzare le informazioni base della palestra



- Interfaccia che mostra le informazioni personali.
- Interfaccia che consente di effettuare una prenotazione.
- Interfaccia che consente di iscriversi ai corsi.



- Interfaccia per gestire le schede di allenamento

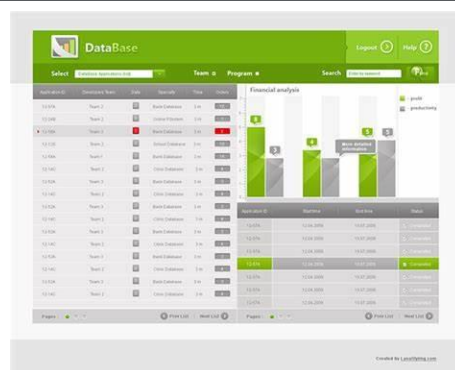
Schede List

[Add Scheda](#)

Scheda nome	Scheda cognome	Scheda obiettivo	Scheda gruppi_muscolari	Scheda serie	Scheda ripetizioni	Actions
John	Smith	Aumento massa	petto,bicipiti,gambe, dorsali	4	8	Update Delete
Pietro	Carputo	Tonificazione	tutto	4	8	Update Delete



- Interfaccia per gestire gli abbonamenti
- Interfaccia per gestire i corsi



3.2 Descrizione dettagliata dei casi d'uso

Caso d'uso	
Titolo	Login – Specializzazione: Login Socio
ID	1.1
Descrizione	Utente non autenticato accedere al sistema inserendo le credenziali fornite per accedere alla propria area personale
Attori primari	Utente non autenticato
Attori secondari	Nessuno
Precondizioni	Il socio deve avere completato l'iscrizione e deve aver avuto le credenziali dal segretario
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Utente non autenticato inserisce username e password 2. Il sistema trova una corrispondenza, il sistema autenticcherà l'Utente 3. Il sistema mostrerà l'area personale del socio.
Postcondizioni	Nessuna
Sequenze degli eventi alternative	Credenziali non corrette non consentono accesso all'area personale e si ritorna al punto 1

Caso d'uso	
Titolo	Login – Specializzazione: Login Segretario o Trainer
ID	1.2
Descrizione	Utente non autenticato inserisce le credenziali fornite per accedere alla propria area personale
Attori primari	Utente non autenticato
Attori secondari	Nessuno
Precondizioni	
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Utente non autenticato inserisce username e password 2. Il sistema trova una corrispondenza, il sistema 3. autenticcherà il segretario oppure il trainer ad accedere alla propria interfaccia 4. Il sistema mostrerà l'area personale specifica
Postcondizioni	Utente è riconosciuto come segretario o trainer e accede al sistema
Sequenze degli eventi alternative	Credenziali non corrette non consentono accesso all'area personale e si ritorna al punto 1

Caso d'uso	
Titolo	Visualizza Abbonamento
ID	5
Descrizione	Il socio accede all'homepage del sito e accede alla propria area personale visualizza il proprio abbonamento.
Attori primari	Socio
Attori secondari	Nessuno
Precondizioni	Il socio deve avere effettuato l'accesso
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. L'utente Generico visualizza la schermata principale del sito. 2. Il socio accede alla propria area personale attraverso le credenziali fornitigli. 3. Il sistema mostra le informazioni relative al proprio abbonamento.
Postcondizioni	Nessuna
Sequenze degli eventi alternative	Credenziali NON valide. Ritorno al punto 1.

Caso d'uso	
Titolo	Elimina Abbonamento
ID	2
Descrizione	Il segretario avvia la procedura di eliminazione di un abbonamento associato ad un socio.
Attori primari	Segretario
Attori secondari	Nessuno
Precondizioni	Il socio deve essere iscritto alla palestra.
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Il segretario inserisce i dati del socio nel sistema per recuperare le informazioni relative all'abbonamento. 2. Il segretario procede all'eliminazione dell'abbonamento. 3. Il segretario conferma l'eliminazione. 4. Il sistema aggiorna gli abbonati.
Postcondizioni	Un abbonamento viene eliminato dal sistema.
Sequenze degli eventi alternative	Il segretario sbaglia ad inserire i dati anagrafici del socio e il sistema lo riporta al punto 1.

Caso d'uso	
Titolo	Crea Abbonamento
ID	2
Descrizione	Il segretario avvia la creazione di un abbonamento associato ad un socio
Attori primari	Segretario
Attori secondari	Nessuno
Precondizioni	L'Utente Generico ha effettuato richiesta di iscrizione alla palestra
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Il segretario crea un abbonamento inserendo tutti i dati anagrafici dell'utente generico. 2. Il sistema salva i dati forniti sul database e restituisce l'esito positivo della creazione. 3. Il segretario fornisce le credenziali di accesso uniche al socio.
Postcondizioni	Un abbonamento viene creato permettendo al socio di visualizzarlo.
Sequenze degli eventi alternative	Il segretario sbaglia ad inserire i dati anagrafici del socio e il sistema lo riporta al punto 1.

Caso d'uso	
Titolo	Rinnova Abbonamento
ID	5
Descrizione	Il segretario può prolungare la data di scadenza di un abbonamento.
Attori primari	Segretario
Attori secondari	Nessuno
Precondizioni	Il socio deve già possedere un abbonamento alla palestra.
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Il segretario visualizza la schermata principale del sito. 2. Il segretario inserisce i dati del socio per ritrovare le informazioni ad esso associate. 3. Il sistema fornisce le informazioni del socio. 4. Il segretario prolunga la data di scadenza dell'abbonamento. 5. Il sistema registra l'avvenuta modifica.
Postcondizioni	Nessuna
Sequenze degli eventi alternative	Socio NON trovato nel sistema.

Caso d'uso	
Titolo	Effettua Prenotazione
ID	3
Descrizione	Il socio dopo aver visualizzato l'affluenza della palestra decide di prenotarsi per una determinata data
Attori primari	Socio
Attori secondari	Nessuno
Precondizioni	Il socio ha effettuato l'accesso al sistema alla propria area personale
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Il socio clicca su "Effettua Prenotazione" 2. Il sistema restituisce lo stato di affluenza nelle tre fasce prestabilite 3. Il socio seleziona il giorno e la fascia oraria nella quale allenarsi 4. Il socio conferma la prenotazione 5. Il sistema mostra al socio la conferma della prenotazione
Postcondizioni	Il sistema aggiorna la disponibilità della palestra
Sequenze degli eventi alternative	Il socio decide di non prenotarsi più per la troppa affluenza o per disponibilità non residua e viene riportato all'homepage

Caso d'uso	
Titolo	Visualizza Scheda
ID	5
Descrizione	Il socio accede all'homepage del sito, accede alla propria area personale e visualizza la propria scheda di allenamento personalizzato.
Attori primari	Socio
Attori secondari	Nessuno
Precondizioni	Il socio ha effettuato l'accesso al sistema alla propria area personale
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. L'utente Generico visualizza la schermata principale del sito. 2. Il socio accede alla propria area personale attraverso le credenziali fornitegli. 3. Il sistema mostra le informazioni relative alla scheda di allenamento personalizzato.
Postcondizioni	Nessuna
Sequenze degli eventi alternative	Credenziali NON valide. Ritorno al punto 1.

Caso d'uso	
Titolo	Visualizza Info Palestra
ID	5
Descrizione	L'utente Generico accede all'homepage del sito e visualizza la lista di servizi offerti dalla palestra.
Attori primari	Utente generico
Attori secondari	Nessuno
Precondizioni	Nessuna
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. L'utente Generico visualizza la schermata principale del sito. 2. Il sistema mostra le informazioni della palestra, i corsi previsti e le aree dedicate.
Postcondizioni	Nessuna
Sequenze degli eventi alternative	Nessuna

Caso d'uso	
Titolo	Effettua Pagamento
ID	5.1
Descrizione	Il Sistema esterno di gestione cassa salda il conto necessario all'abbonamento
Attori primari	Sistema gestione cassa
Attori secondari	
Precondizioni	L'Utente Generico ha effettuato richiesta di iscrizione alla palestra
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. Il sistema esterno vuole completare il pagamento di un socio 2. Il socio fornisce il denaro 3. Il sistema esterno verifica che il denaro sia sufficiente a saldare il conto 4. Il sistema valida il pagamento
Postcondizioni	Il sistema registra l'esito del pagamento e il segretario fornisce le credenziali di accesso al socio
Sequenze degli eventi alternative	Il pagamento può essere effettuato anche tramite iban e ad avvenuta transazione il sistema valida il pagamento

Caso d'uso	
Titolo	Gestione Corsi
ID	6
Descrizione	Il segretario crea o modifica i corsi organizzati
Attori primari	Segretario
Attori secondari	Nessuno
Precondizioni	Il segretario ha effettuato l'accesso al sistema
Sequenza degli eventi principali	<p>CREATE</p> <ol style="list-style-type: none"> 1. Il segretario inserisce il nome del corso, tipologia, attività previste e gli orari in cui si terrà 2. Il sistema crea il corso sul database e fornisce un resoconto dell'esito <p>READ</p> <ol style="list-style-type: none"> 1. Il segretario richiede di vedere tutti i corsi disponibili 2. Il sistema restituisce i corsi <p>UPDATE</p> <ol style="list-style-type: none"> 1. Il segretario inserisce l'id del corso che desidera modificare 2. Il sistema recupera tutte le informazioni associate a tale corso 3. Il segretario effettua le modifiche di suo interesse 4. Il sistema salva le modifiche effettuate dal segretario. <p>DELETE</p> <ol style="list-style-type: none"> 1. Il segretario inserisce il numero del corso che intende cancellare 2. Il sistema cancella dal database il corso e restituisce l'esito dell'operazione
Postcondizioni	Il corso viene sottoposto alle operazioni CRUD sul database
Sequenze degli eventi alternative	Gestione degli eventuali errori per le varie operazioni CRUD

NOTA: Per il caso d'uso Gestione Corsi, si è deciso di utilizzare lo standard proposto da Övergaard and Palmkvist ed è stata inserita una sequenza separata per ognuna delle operazioni CRUD (Create, Read, Update, Delete)

Caso d'uso	
Titolo	Gestione Scheda
ID	7
Descrizione	Il trainer crea o modifica le schede
Attori primari	Trainer
Attori secondari	Nessuno
Precondizioni	Il trainer ha effettuato l'accesso al sistema
Sequenza degli eventi principali	<p>CREATE</p> <ol style="list-style-type: none"> 1. Il trainer inserisce i dati del socio 2. Il sistema crea una scheda sul database e fornisce un resoconto dell'esito <p>READ</p> <ol style="list-style-type: none"> 1. Il trainer richiede di vedere tutte le schede create 2. Il sistema restituisce le schede <p>UPDATE</p> <ol style="list-style-type: none"> 1. Il trainer inserisce id della scheda da modificare 2. Il sistema recupera la scheda desiderata 3. Il trainer effettua le modifiche di suo interesse 4. Il sistema salva le modifiche effettuate dal trainer. <p>DELETE</p> <ol style="list-style-type: none"> 1. Il trainer inserisce id della scheda che intende cancellare 2. Il sistema cancella dal database la scheda e restituisce l'esito dell'operazione
Postcondizioni	La scheda viene sottoposta alle operazioni CRUD sul database
Sequenze degli eventi alternative	Gestione degli eventuali errori per le varie operazioni CRUD

NOTA: Per il caso d'uso Gestione Scheda, si è deciso di utilizzare lo standard proposto da Övergaard and Palmkvist ed è stata inserita una sequenza separata per ognuna delle operazioni CRUD (Create, Read, Update, Delete)

Caso d'uso	
Titolo	Iscrizione Corso
ID	8
Descrizione	Il socio può effettuare una iscrizione ad un corso organizzato dalla palestra
Attori primari	Socio
Attori secondari	Nessuno
Precondizioni	
Sequenza degli eventi principali	<ol style="list-style-type: none"> 1. L'utente generico accede al sito web. 2. Il socio accede al sistema inserendo le apposite credenziali. 3. Il sistema fornisce una vista di tutti i corsi disponibili all'iscrizione. 4. Il socio sceglie il corso al quale iscriversi. 5. Il sistema convalida l'iscrizione al corso selezionato.
Postcondizioni	Il socio risulterà iscritto al corso selezionato.
Sequenze degli eventi alternative	Credenziali NON valide. Ritorno al punto 1.

3.3 Diagramma di dominio

Nel diagramma di dominio in Fig. 9 sono mostrate le entità in gioco e le associazioni tra esse. Tale diagramma descrive le varie entità che fanno parte o hanno rilevanza nel sistema e le loro relazioni. Inoltre, è utile per mettere a fuoco i concetti fondamentali di un sistema e definire un vocabolario specifico per il sistema. Vengono messi in evidenza gli aspetti essenziali del dominio di interesse tralasciando momentaneamente le funzionalità di sistema che verranno successivamente implementate.

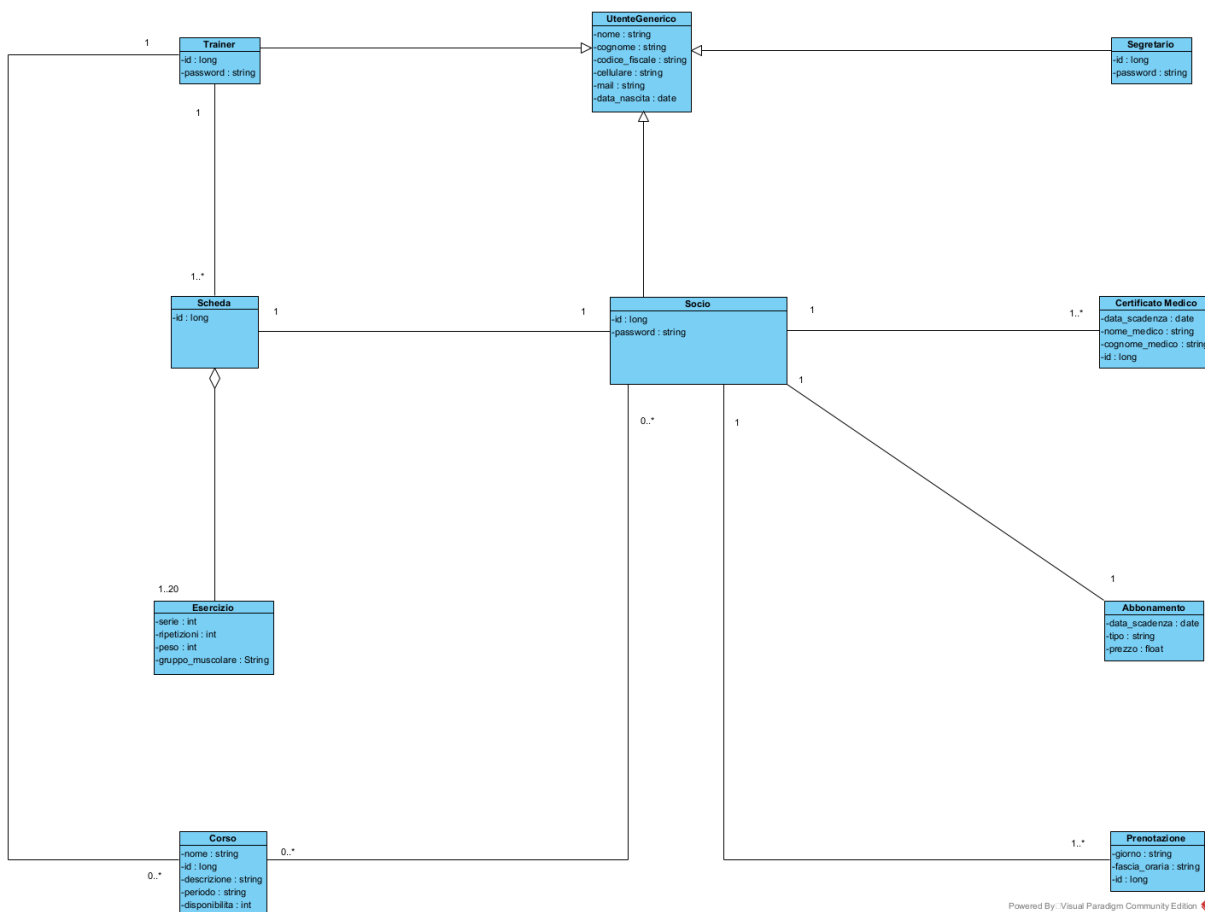


Fig.9 Diagramma di dominio

3.4 Diagramma di Sequenza di Analisi

In ciascun diagramma di Sequenza di Analisi in Fig. 10-18 è mostrata la sequenza degli eventi generati dall'interazione dell'utente con il sistema.

3.4.1 Sequence Diagram – Effettua Login Socio

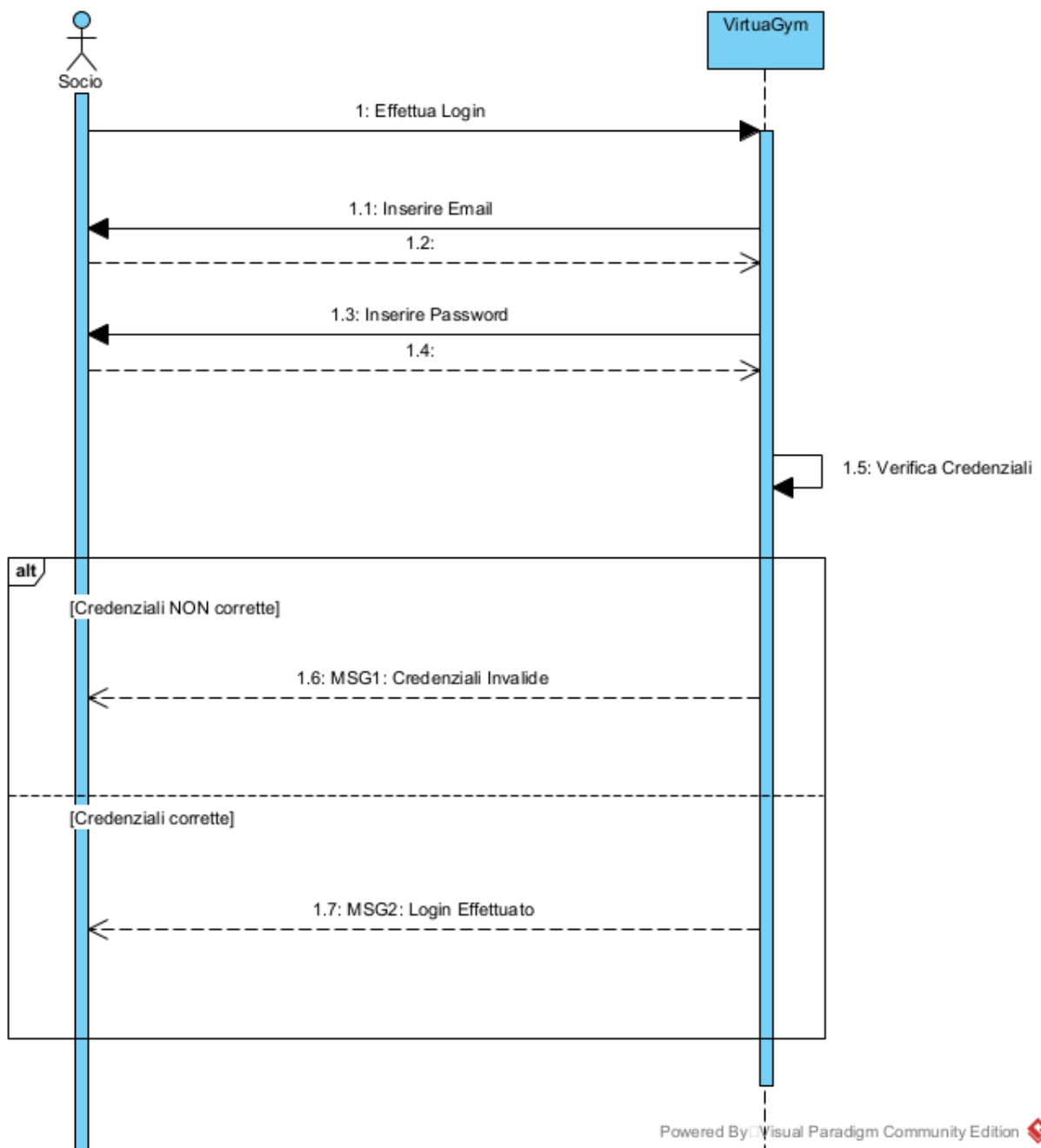


Fig.10 Sequence Diagram – Effettua Login Socio

3.4.2 Sequence Diagram – Effettua Registrazione

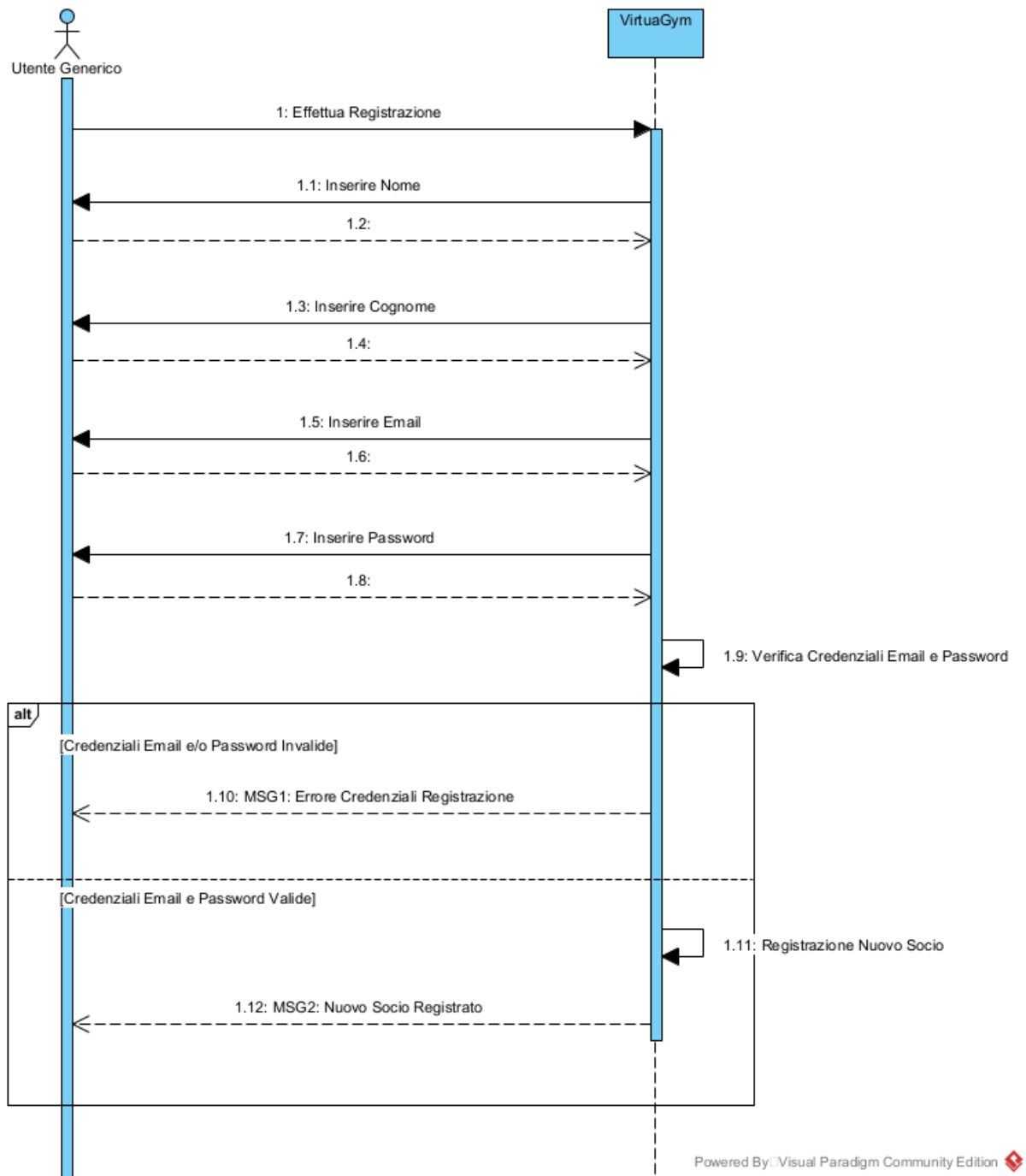


Fig.11 Sequence Diagram – Effettua Registrazione

3.4.3 Sequence Diagram – Visualizza Abbonamento

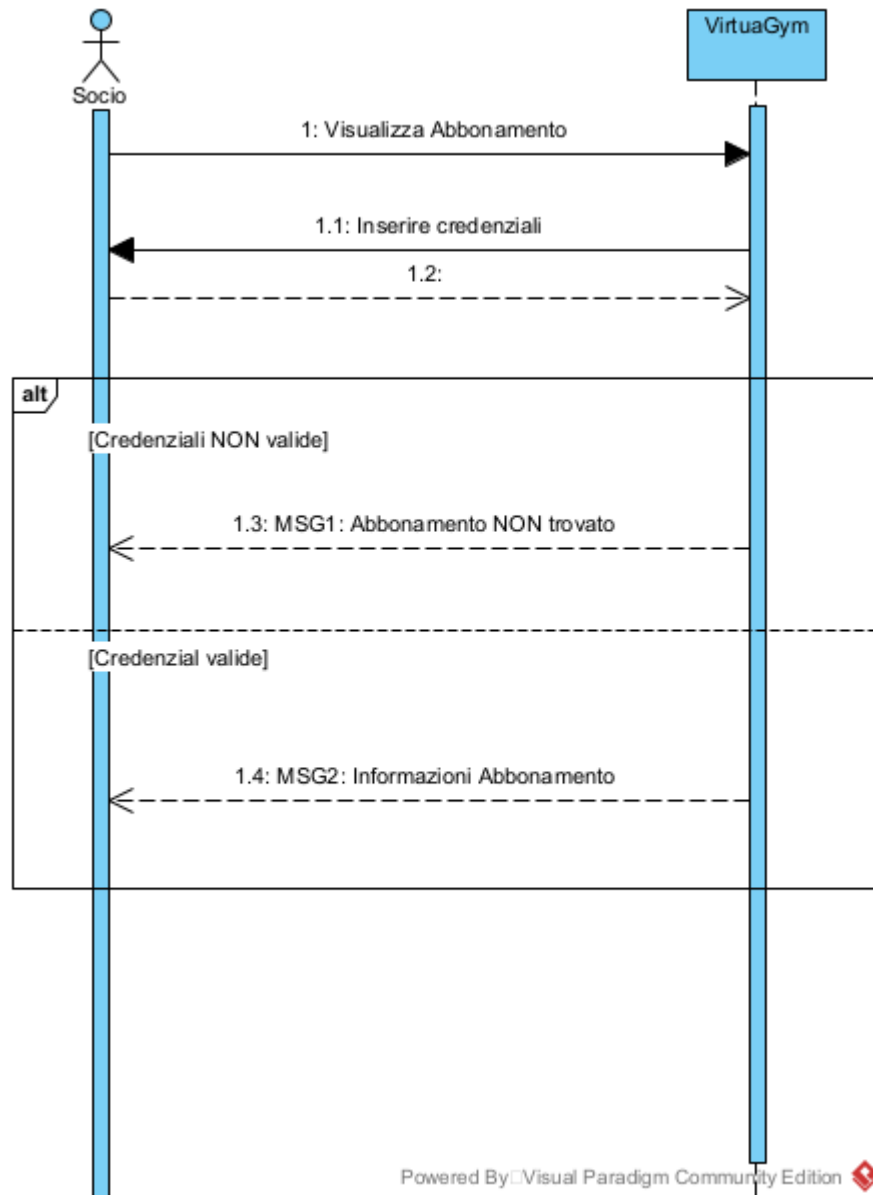


Fig.12 Sequence Diagram – Visualizza Abbonamento

3.4.4 Sequence Diagram – Crea Abbonamento

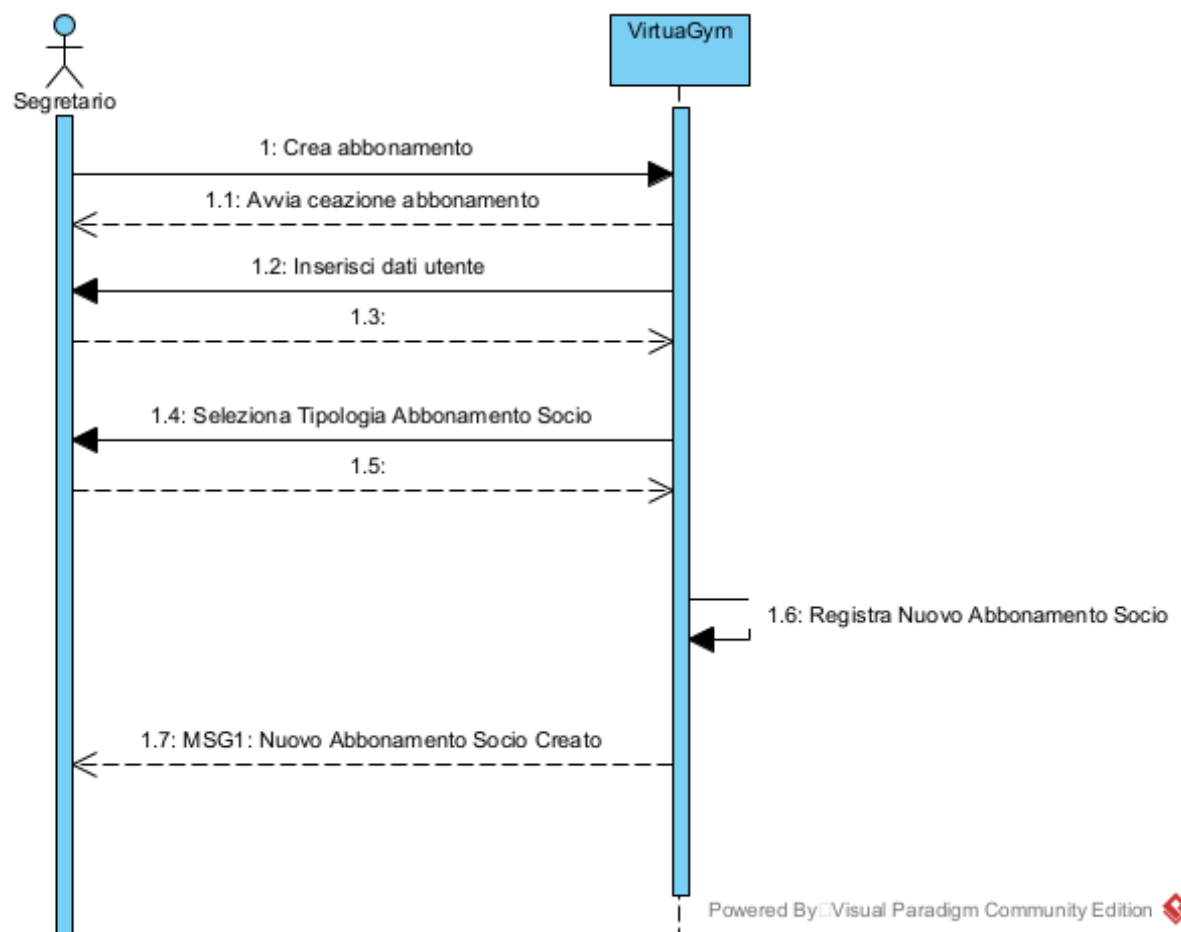


Fig.13 Sequence Diagram – Crea Abbonamento

3.4.5 Sequence Diagram – Visualizza Scheda

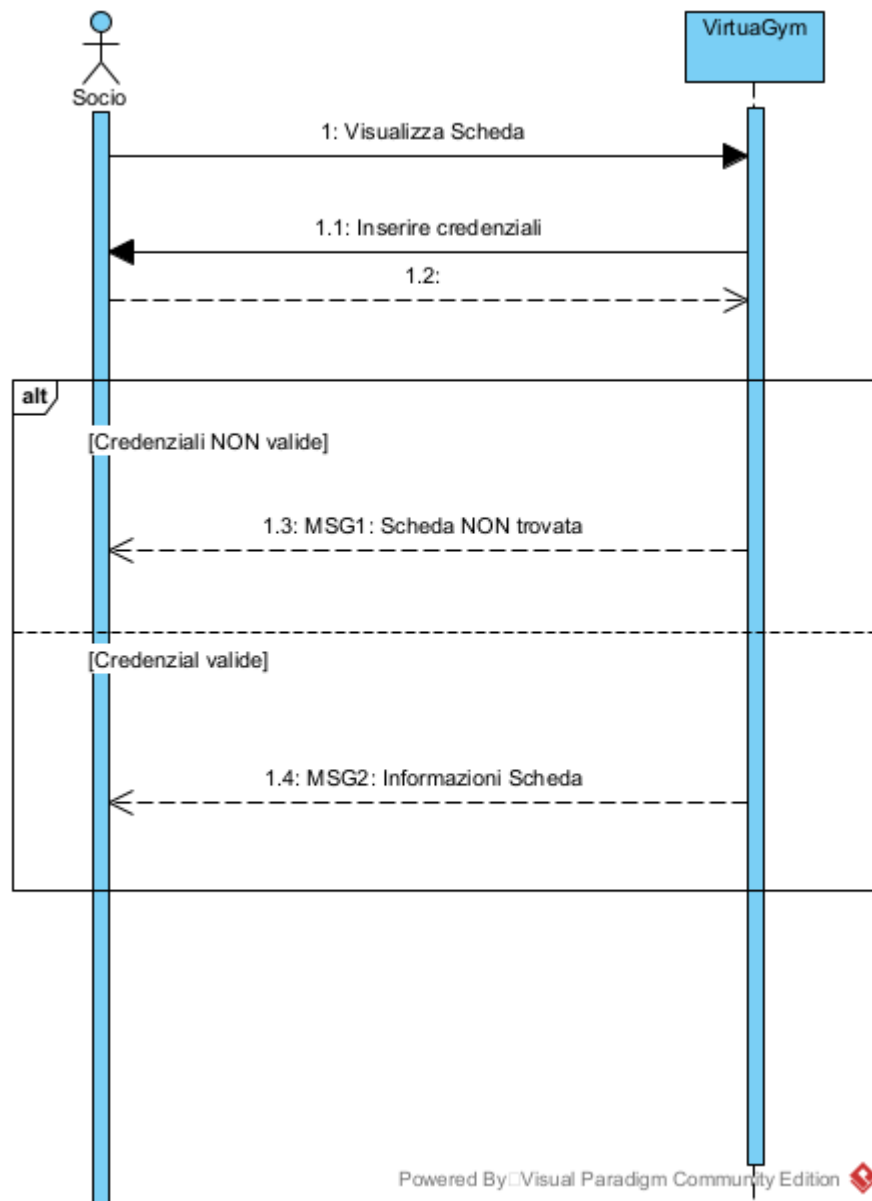


Fig.14 Sequence Diagram – Visualizza Scheda

3.4.6 Sequence Diagram – Visualizza Info Palestra

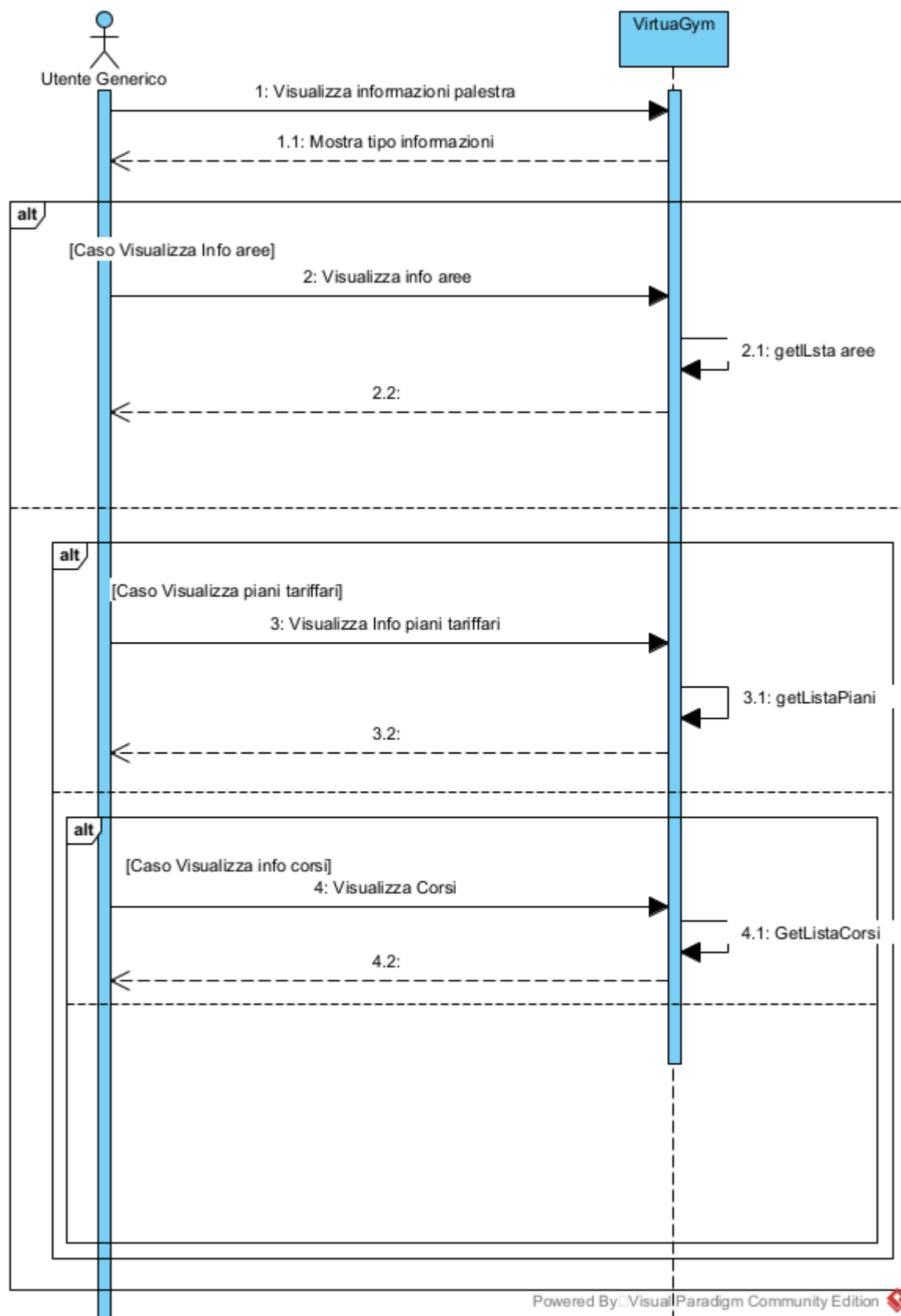


Fig.15 Sequence Diagram – Visualizza Info Palestra

3.4.7 Sequence Diagram – Gestione Corsi

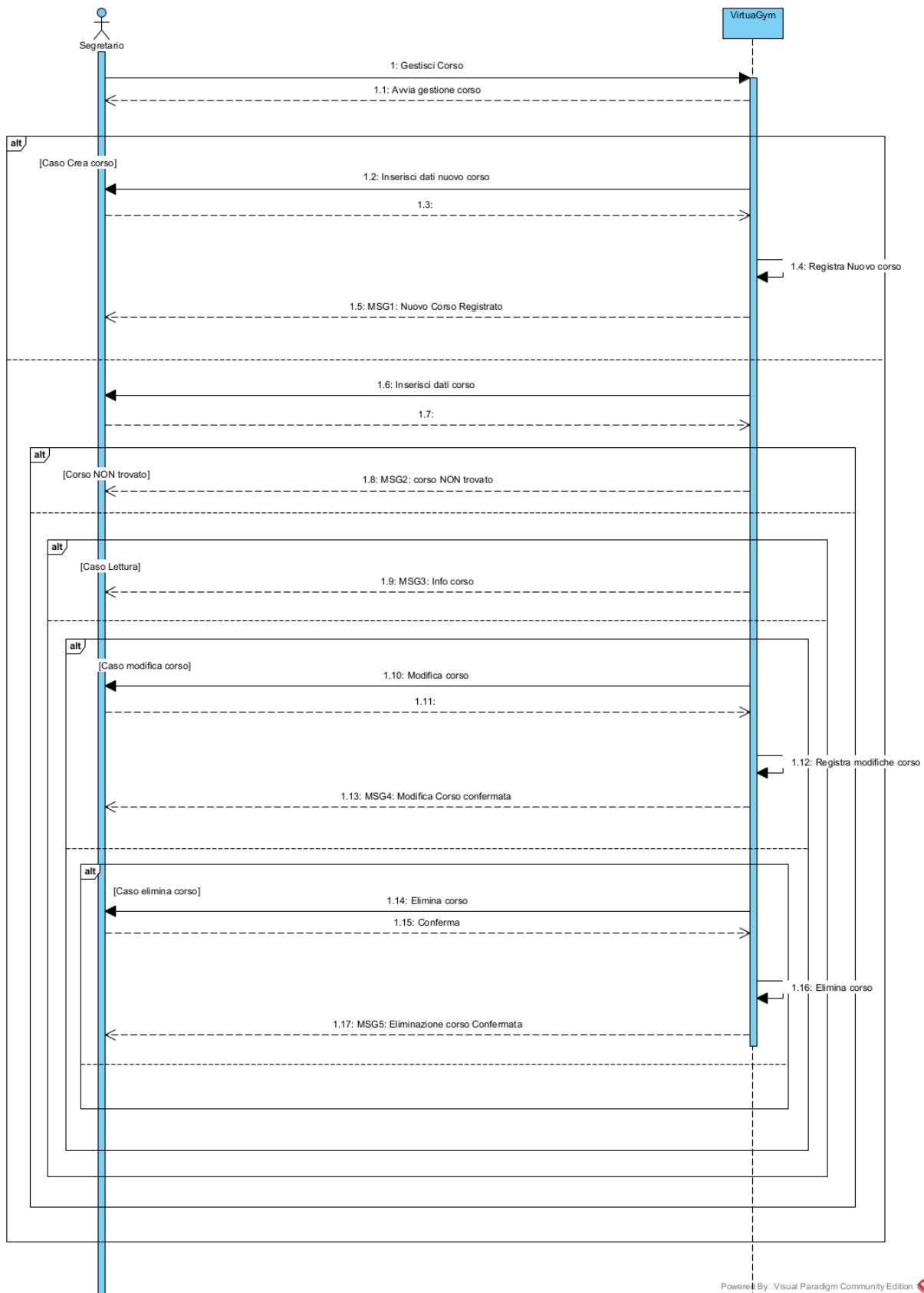


Fig.16 Sequence Diagram – Gestione Corsi

3.4.8 Sequence Diagram – Gestione Scheda

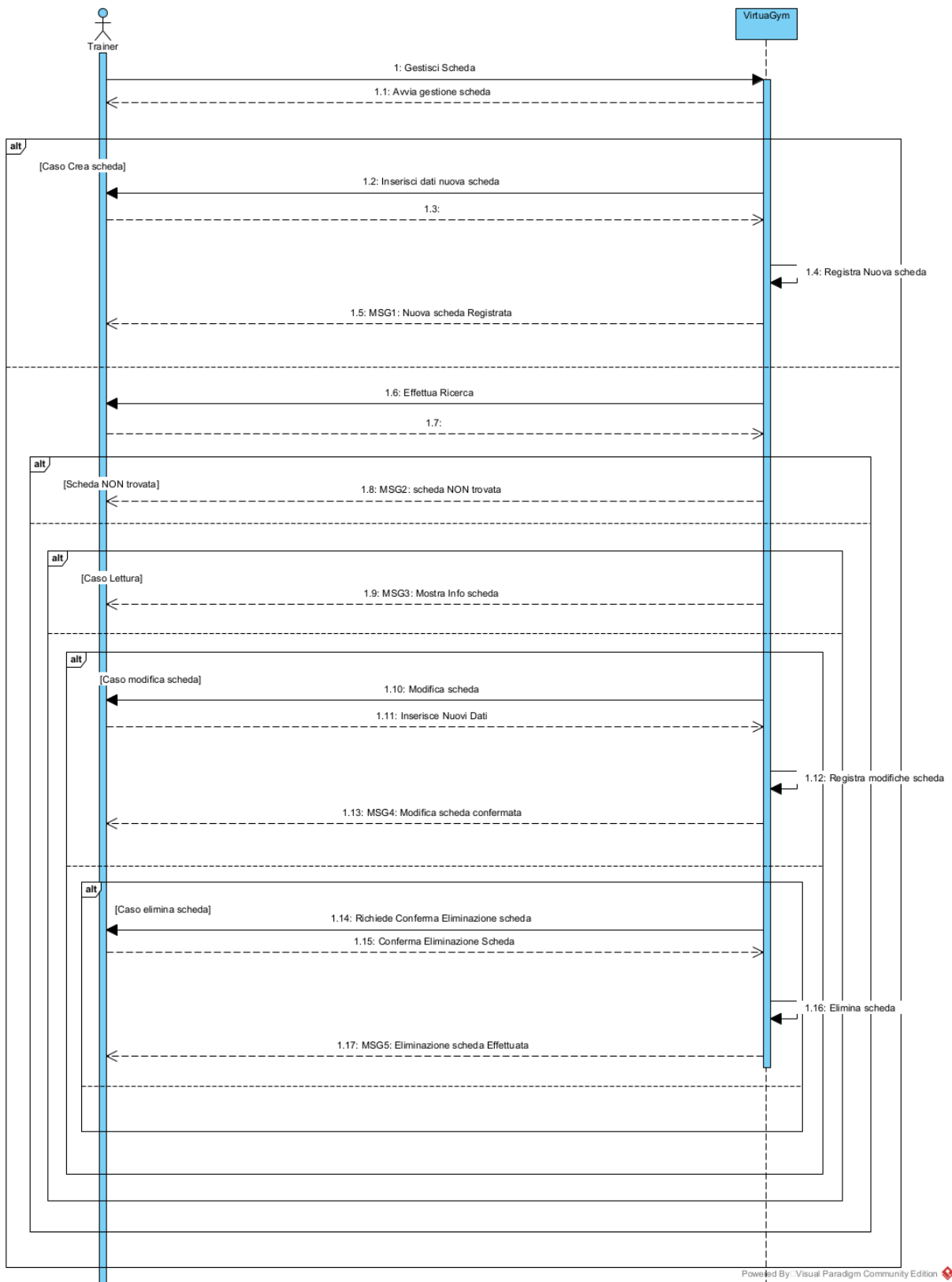


Fig. 17 Sequence Diagram – Gestione Scheda

3.4.9 Sequence Diagram – Iscrizione corso

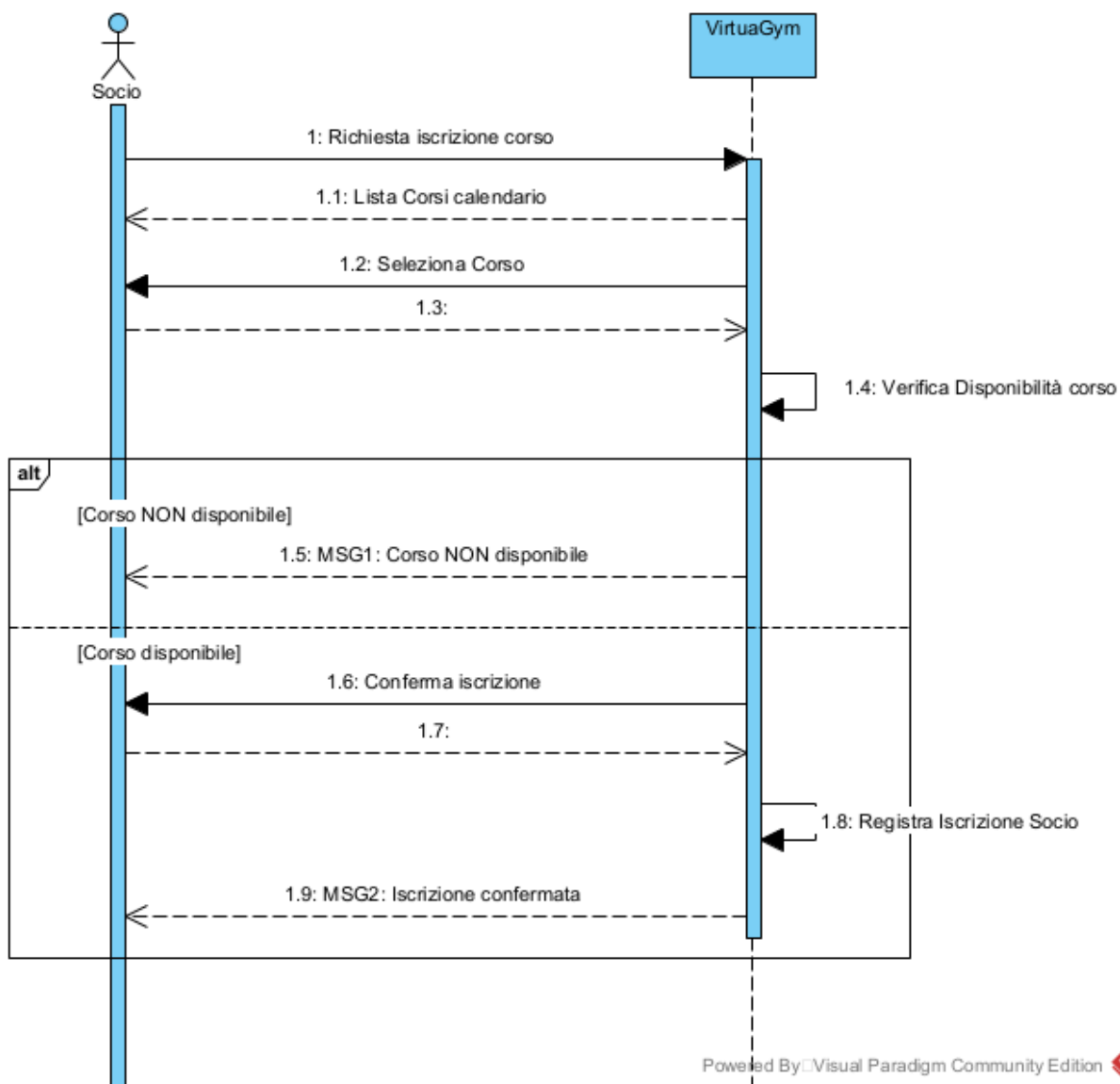


Fig. 18 Sequence Diagram – Iscrizione Corso

3.5 Diagramma di attività

In ciascun diagramma di attività in Fig. 19-27 è mostrata la rappresentazione grafica degli scenari principali dei casi d'uso.

3.5.1 Activity Diagram – Effettua Login Socio

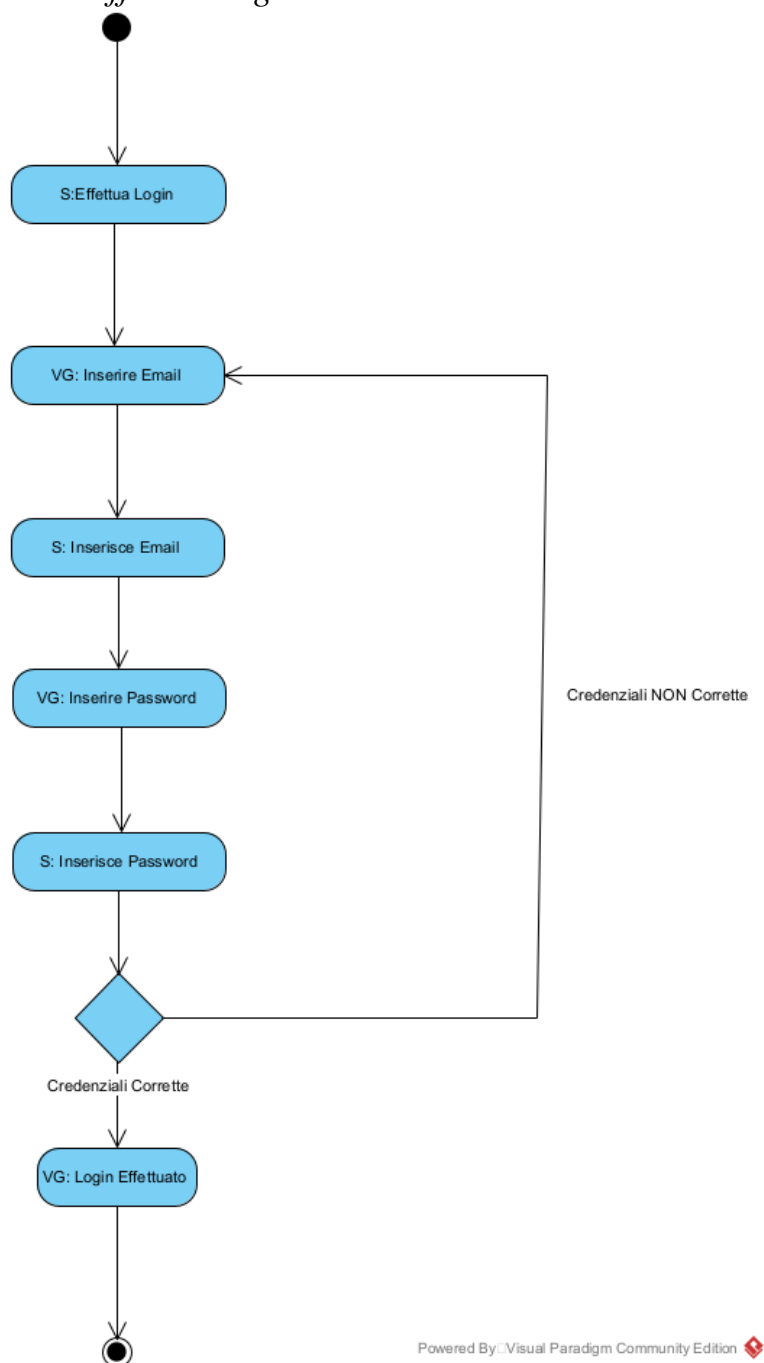


Fig. 19 Activity Diagram – Effettua Login Socio

3.5.2 Activity Diagram – Effettua Registrazione

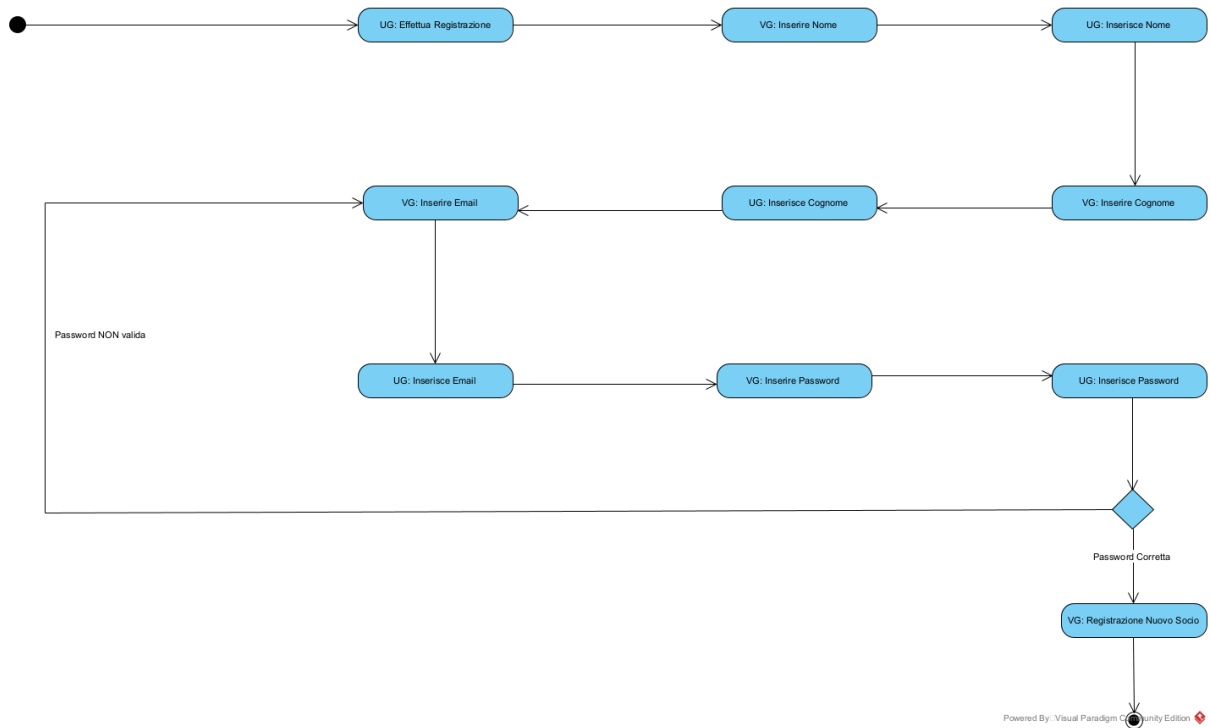


Fig. 20 Activity Diagram – Effettua Registrazione

3.5.3 Activity Diagram – Visualizza Abbonamento

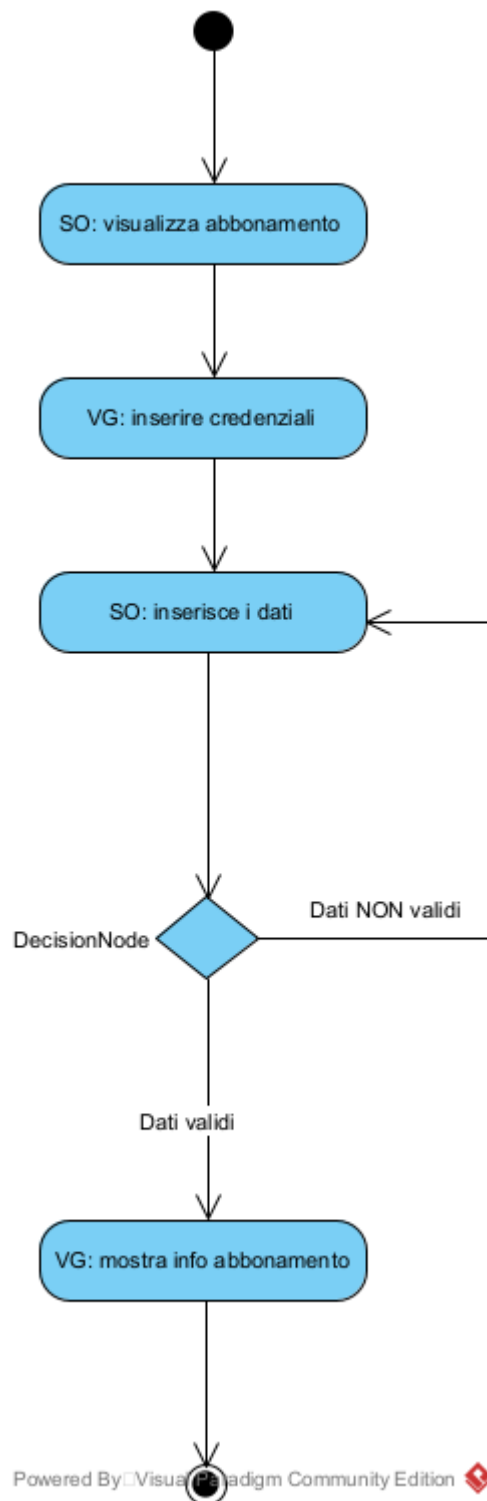


Fig. 21 Activity Diagram – Visualizza Abbonamento

3.5.4 Activity Diagram – Crea Abbonamento

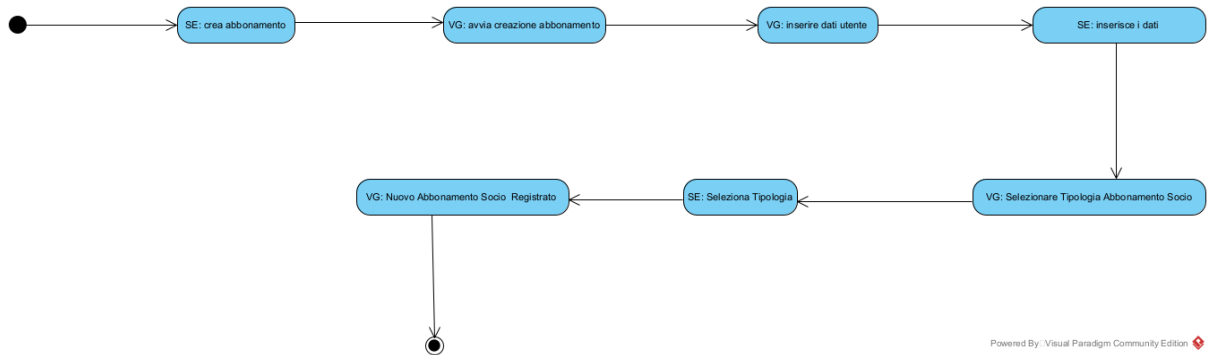


Fig. 22 Activity Diagram – Crea Abbonamento

3.5.5 Activity Diagram – Visualizza Scheda

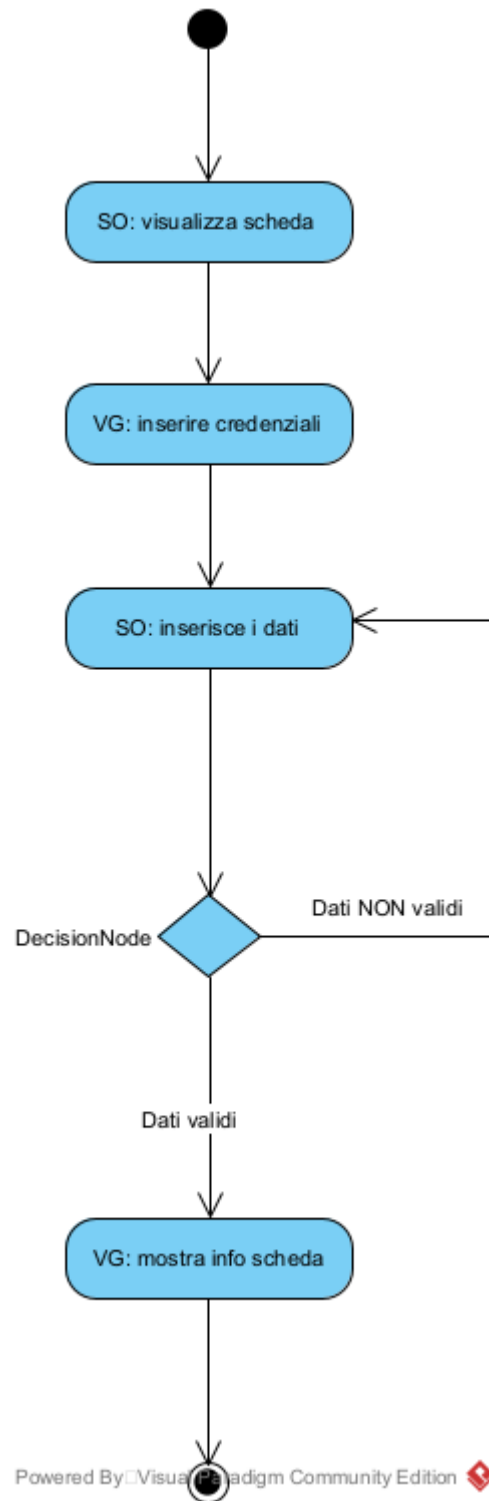


Fig. 23 Activity Diagram – Visualizza Scheda

3.5.6 Activity Diagram – Visualizza Info Palestra

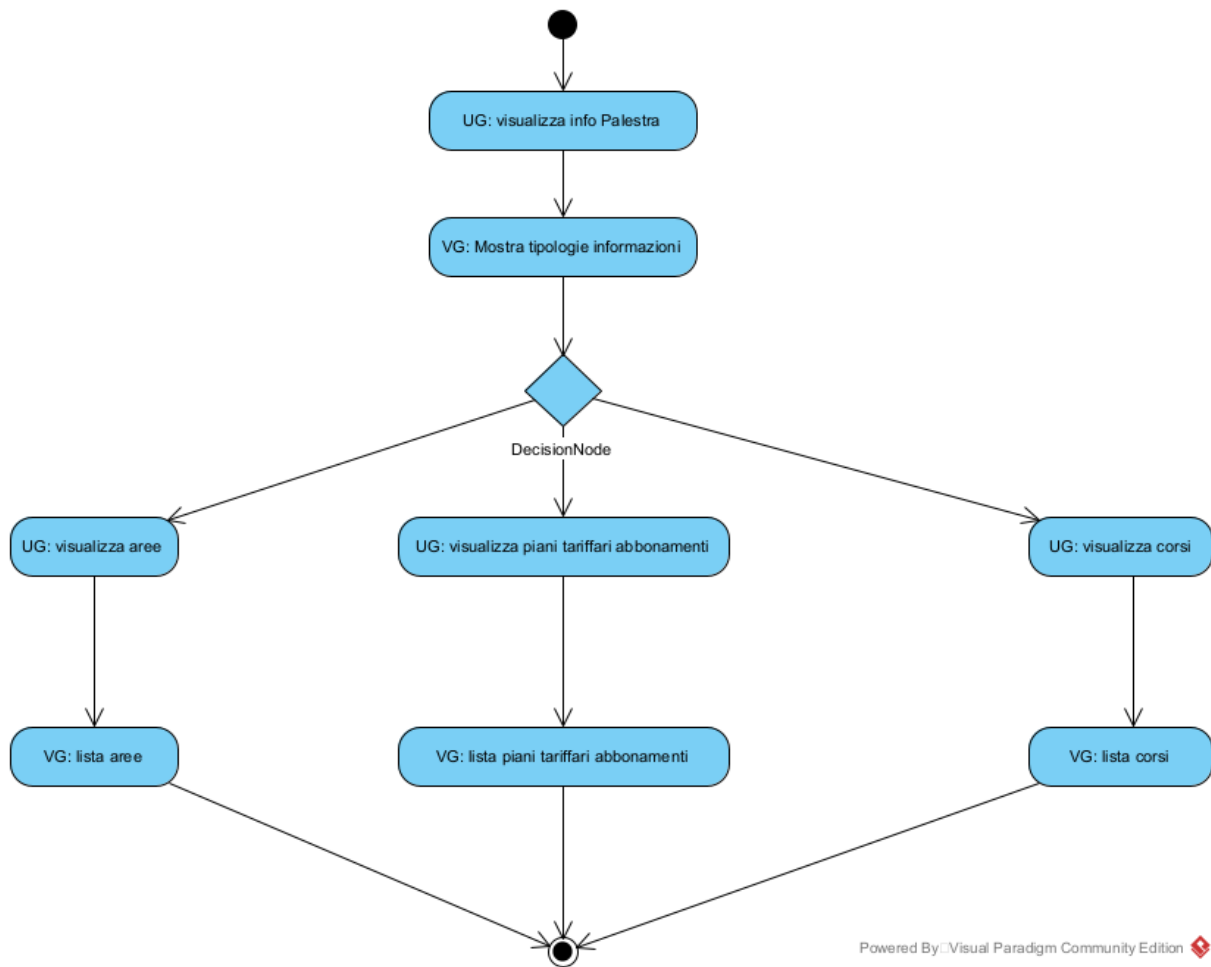


Fig. 24 Activity Diagram – Visualizza Info Palestra

3.5.7 Activity Diagram – Gestione Corsi

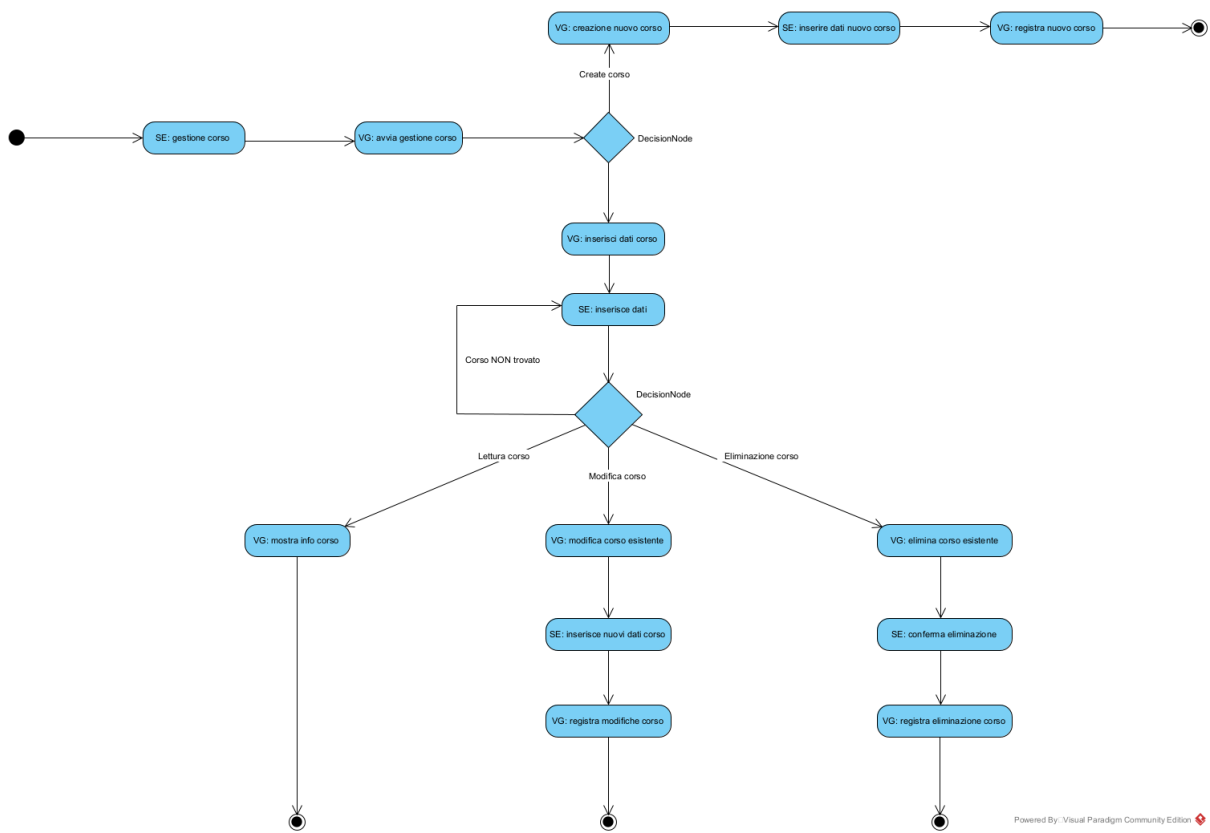


Fig. 25 Activity Diagram – Gestione Corsi

3.5.8 Activity Diagram – Gestione Scheda

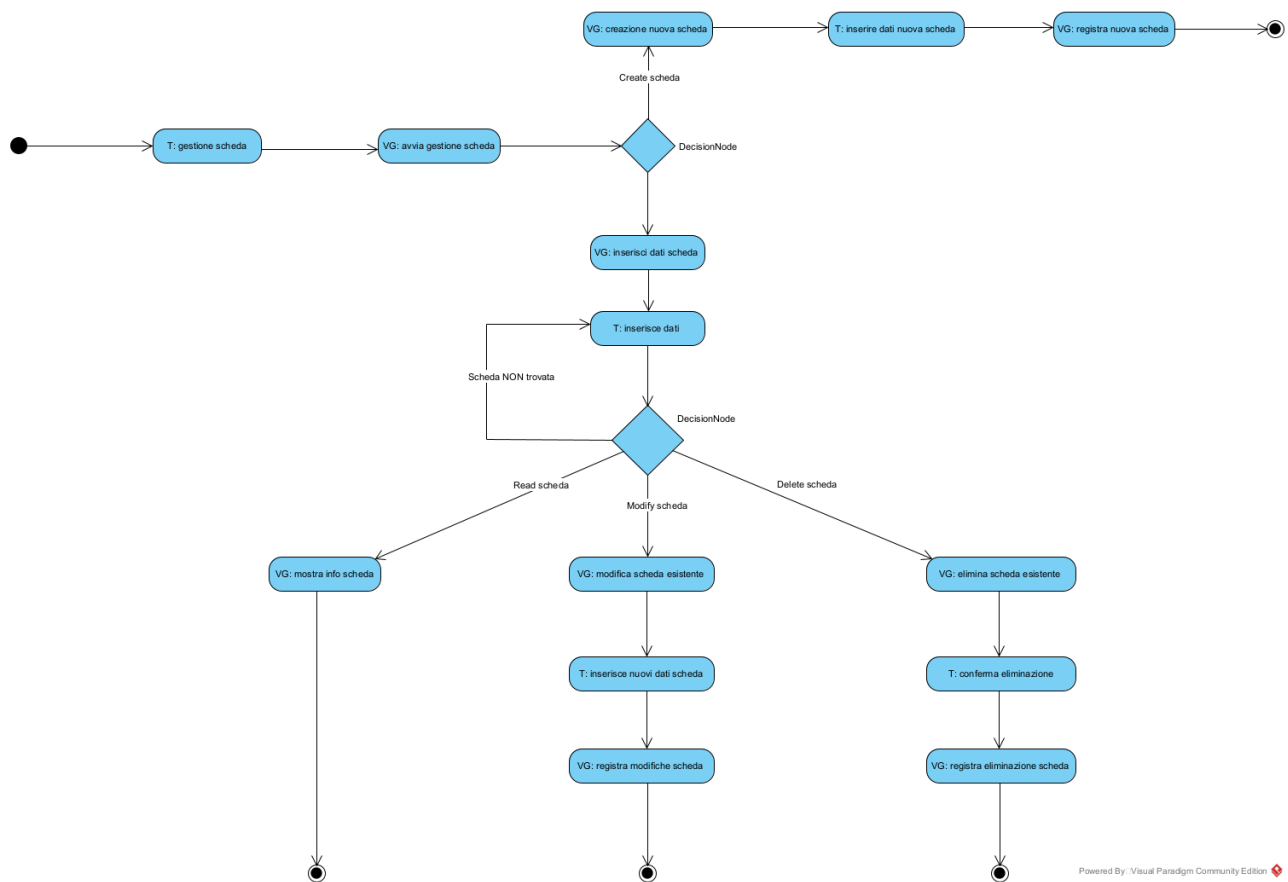


Fig. 26 Activity Diagram – Gestione Scheda

3.5.9 Activity Diagram – Iscrizione Corso

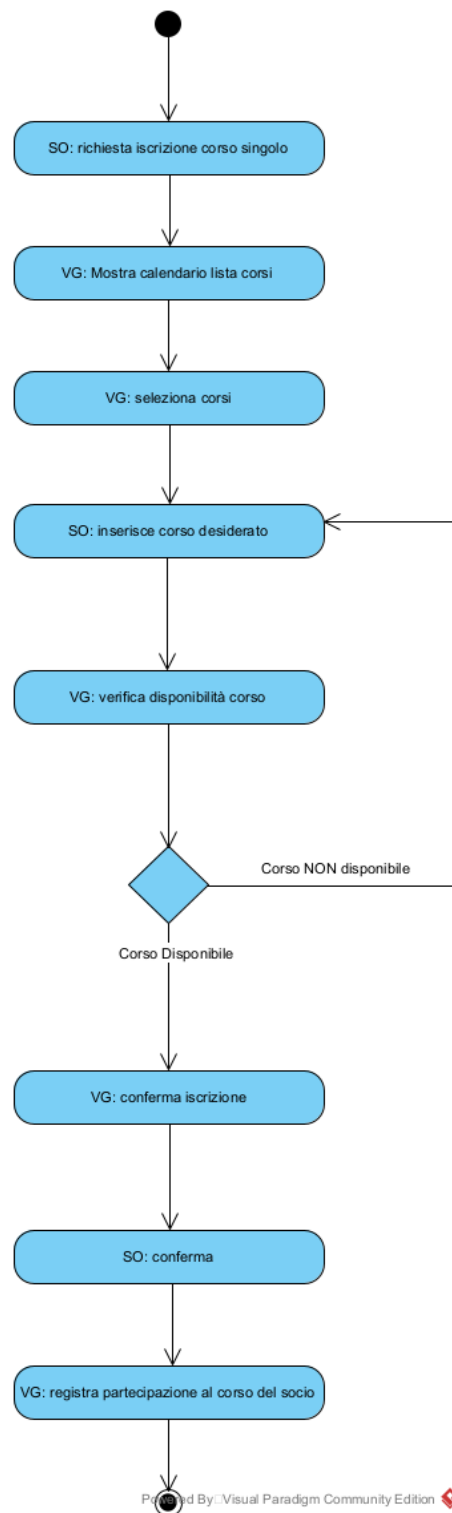


Fig. 27 Activity Diagram – Iscrizione Corso

3.6 Diagramma delle classi

Nel diagramma delle classi in Fig. 28 vengono mostrate le operazioni caratterizzanti ciascuna classe del sistema.

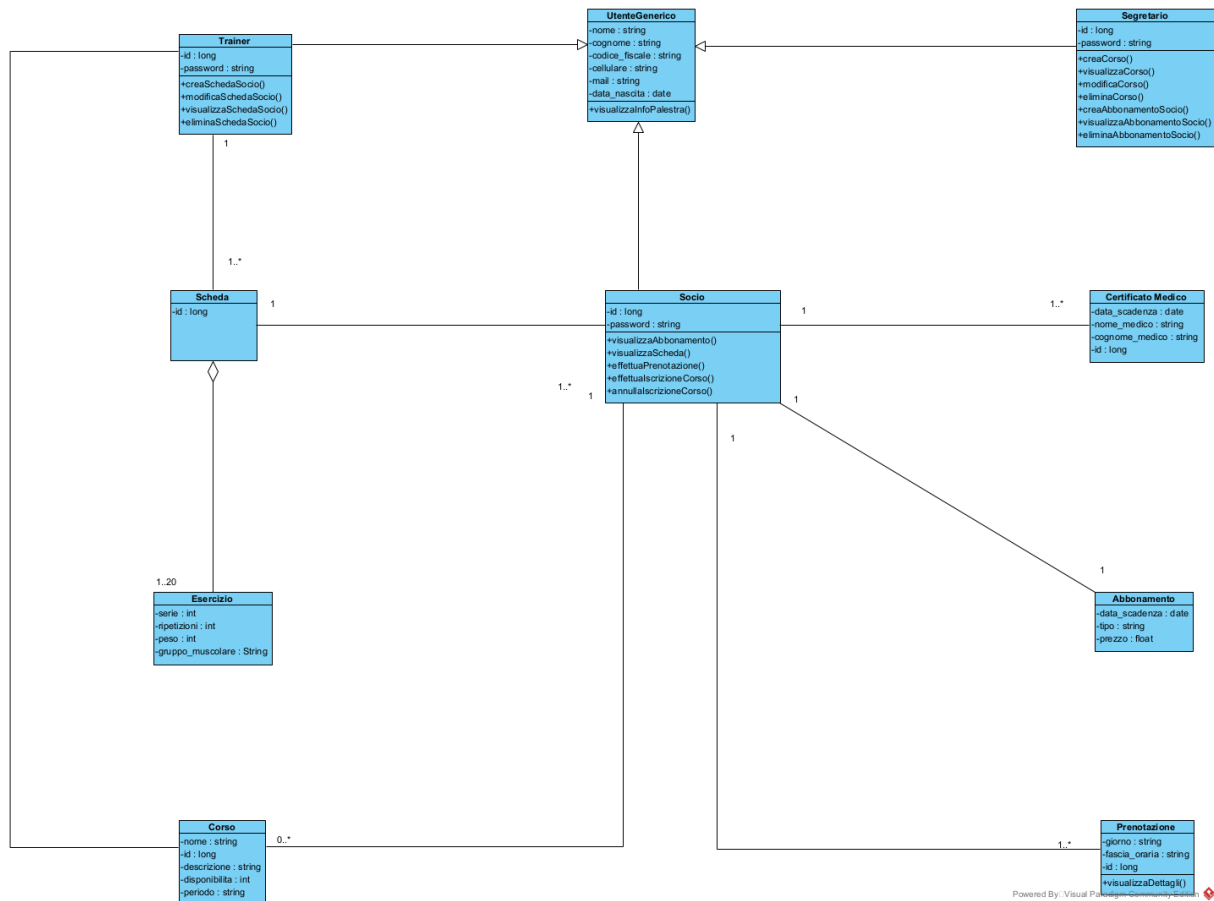


Fig. 28 Class Diagram

Il criterio adottato   basato sulle seguenti considerazioni:

- assegnare le operazioni alla classe che   in grado di soddisfare la responsabilit  ad essa assegnata.
- assegnare le operazioni alla classe che invoca il metodo

Si   inoltre cercato di favorire il “Low Coupling” evitando di assegnare i metodi a classi molto “distanti” che richiederebbero pi  passaggi per recuperare i dati utili necessari ad eseguire l’operazione.

3.7 Diagramma di contesto

Nel diagramma di contesto in Fig. 29 sono evidenziate le interazioni tra gli attori esterni e il sistema “VIRTUAGYM”.

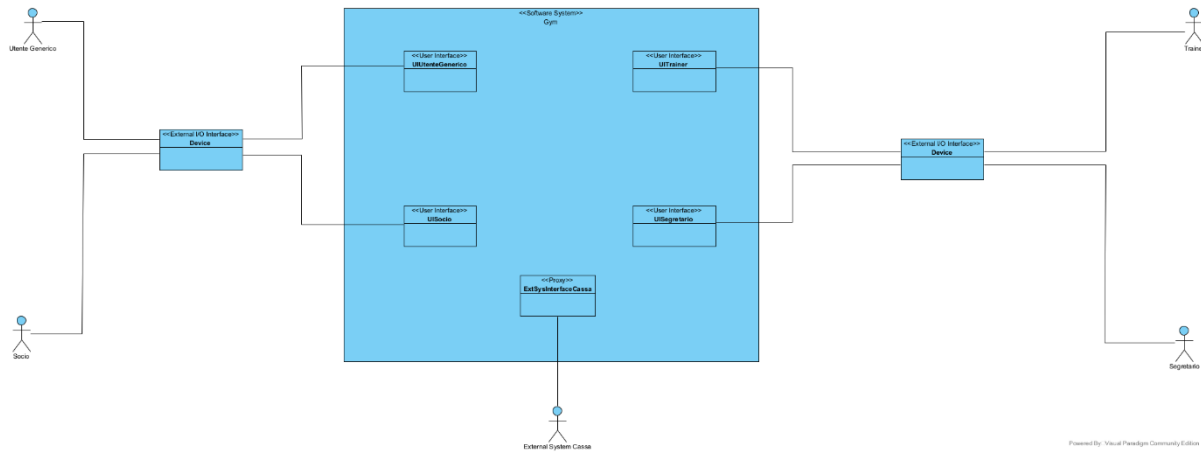


Fig. 29 Context Diagram



Capitolo 4: Architettura e scelte di progetto

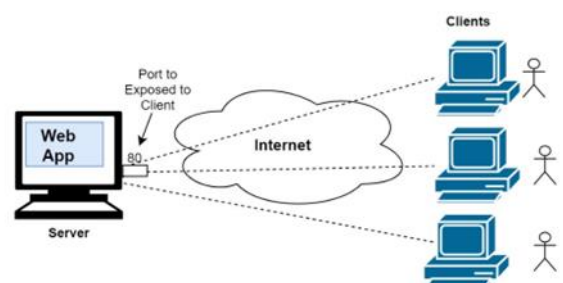
Architettura Software e scelte di progetto

Questa fase ha come input la documentazione di analisi e ha il compito di aggiungere dettagli di progettazione e di sviluppo delle diverse soluzioni implementative.

4.1 Architettura logica

Lo stile architetturale prescelto per l'applicazione è quello Client-Server. Il paradigma client-server è un modello di iterazione tra processi software, ove processi interagenti si suddividono tra client, che richiedono servizi, e server, che offrono servizi. Il processo client è tipicamente dedicato ad interagire con l'utente finale; esso svolge un ruolo attivo, in quanto genera autonomamente richieste di servizi. Invece, il processo server è reattivo: esso svolge una computazione solo a seguito di una richiesta da parte di un qualunque client.

La maggior parte del codice, logica di business e accesso ai dati, è tenuta propria nel Server, che quindi è definito come fat.



Vantaggi: **Scalability, Integrity, Easy Maintenance and Security.**

Svantaggi (se paragonata al P2P):

- **Overloaded server:** Quando sono frequenti le richieste simultanee dei client, i server vengono sovraccaricati gravemente, formando una congestione del traffico.
Ma in una rete P2P l'aggiunta di più nodi aumenterà la sua larghezza di banda poiché è calcolata come la somma delle larghezze di banda di ciascun nodo nella rete.
- **Centralized architecture:** Impatto dell'architettura centralizzata: poiché è centralizzata se un server critico si guasta, le richieste dei client non vengono soddisfatte. Pertanto il client/server manca della robustezza di una buona rete P2P.

4.2 Design : Component Diagram

Per illustrare l'architettura del sistema, si è fatto ricorso al Component Diagram, il quale è un diagramma che ha lo scopo di rappresentare la struttura interna del sistema software modellato in termini dei suoi componenti principali e delle relazioni fra di essi. Per componente si intende una unità software dotata di una precisa identità, nonché responsabilità e interfacce ben definite

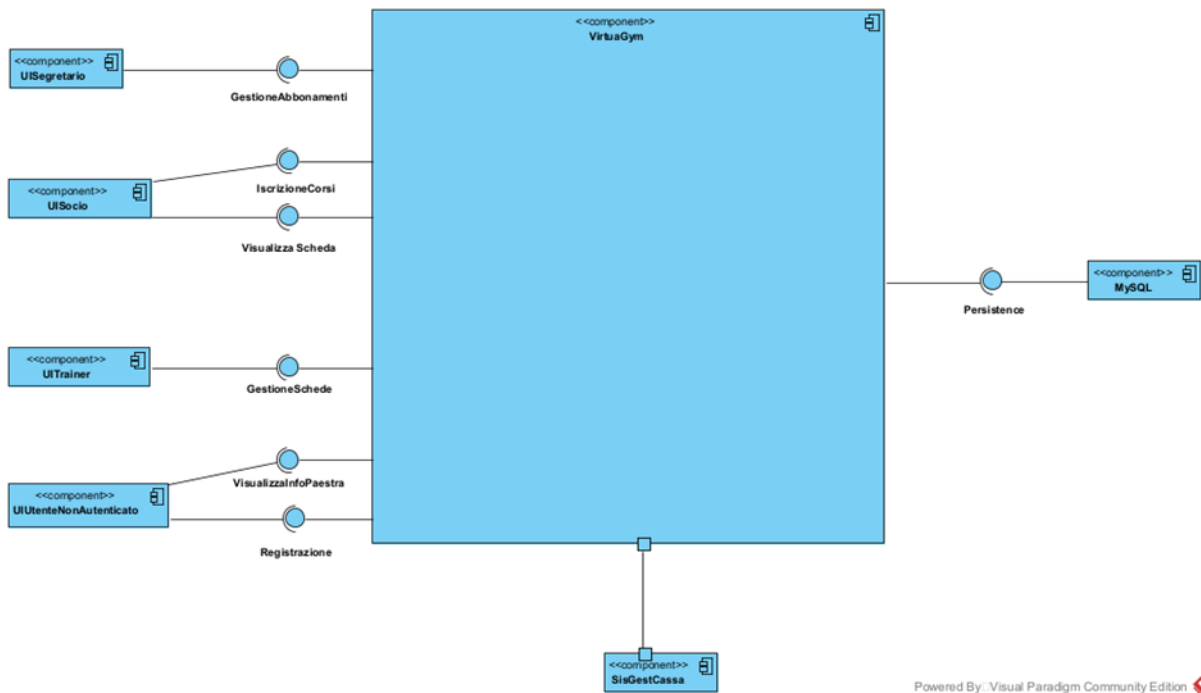


Figura: Component Diagram

Come evidenziato dal diagramma in Figura, il sistema espone una serie di risorse, ognuna delle quali può essere consumata da un tipo di utente diverso (Trainer, Cliente, Segretario). Ciascuna risorsa è rappresentata da una URL e permette al client di eseguire determinate operazioni.

Alla fine le componenti sono User Interfaces di tipo html che sono inviate al Client specifico, tramite un browser engine, e eseguite dallo stesso.

Riassumendo si hanno 3 componenti principali:

- **Client:** web browser in grado di inoltrare richieste HTTP al server. I diversi client possono essere visti come thin-client, in quanto la logica applicativa è concentrata quasi interamente sul server.
- **Server:** web server che espone risorse ai client. Il server richiede un'interfaccia fornita dal sistema di notifiche esterno.

- **Database:** database relazionale che permette al server di gestire la persistenza dei dati.

4.3 Pattern Utilizzati

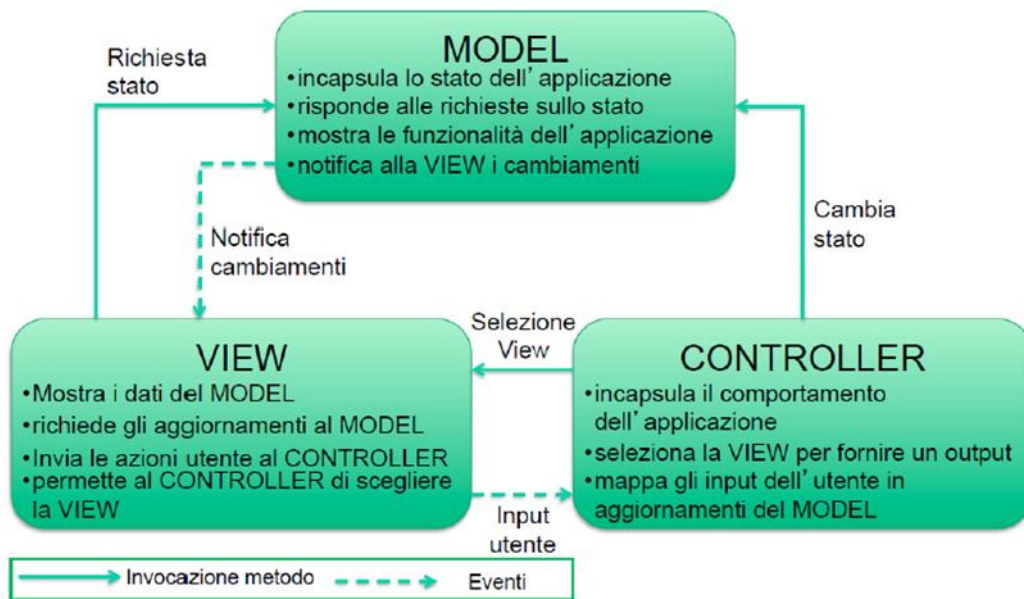
Di seguito verranno discussi i pattern (architetturali e di design) usati per l'applicazione.

4.3.1 MVC

Per la realizzazione del server è stato scelto il pattern architetturale Model-View-Controller. La scelta è ricaduta su di esso vista la necessità di mantenere separata la logica di visualizzazione dalle logiche di business e di accesso ai dati.

In particolare, ricordiamo i principali ruoli di questi componenti:

- Il Model gestisce i dati e le regole di business per poter interagire con essi.
Fornisce alla view ed al controller le funzionalità per l'accesso e l'aggiornamento.
- La View gestisce la logica di presentazione dei dati e permette di implementare diverse viste su di essi
- Il Controller trasforma gli input utente della view in azioni da eseguire sul model e seleziona le schermate da mostrare tramite la view. I dati riportati dall'utente attraverso la view verranno automaticamente riportati nel view model.

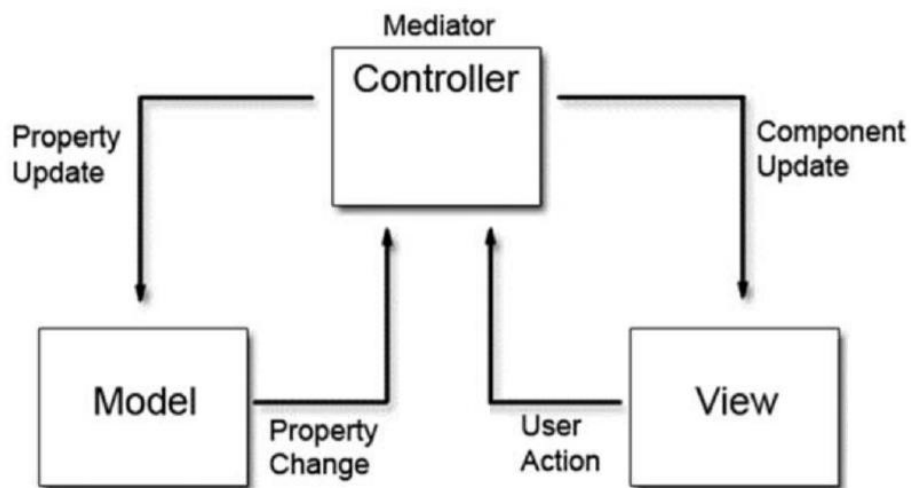


A questa tipologia di MVC detta di tipo push, si affianca una seconda tipologia di tipo pull tipicamente utilizzata nelle soluzioni distribuite.

Nel Model View Controller di tipo pull la sequenza di azioni tra i tre componenti è leggermente diversa.

In particolare:

- La view riceve un input utente e lo invia al controller
- Il controller seleziona l'azione corrispondente all'input, che dovrà essere eseguita sul model
- Il model esegue l'azione, si aggiorna e restituisce al controller il nuovo stato
- Il controller è a sua volta delegato di restituire alla view il nuovo stato del model. dell'UI.



4.3.2 repository

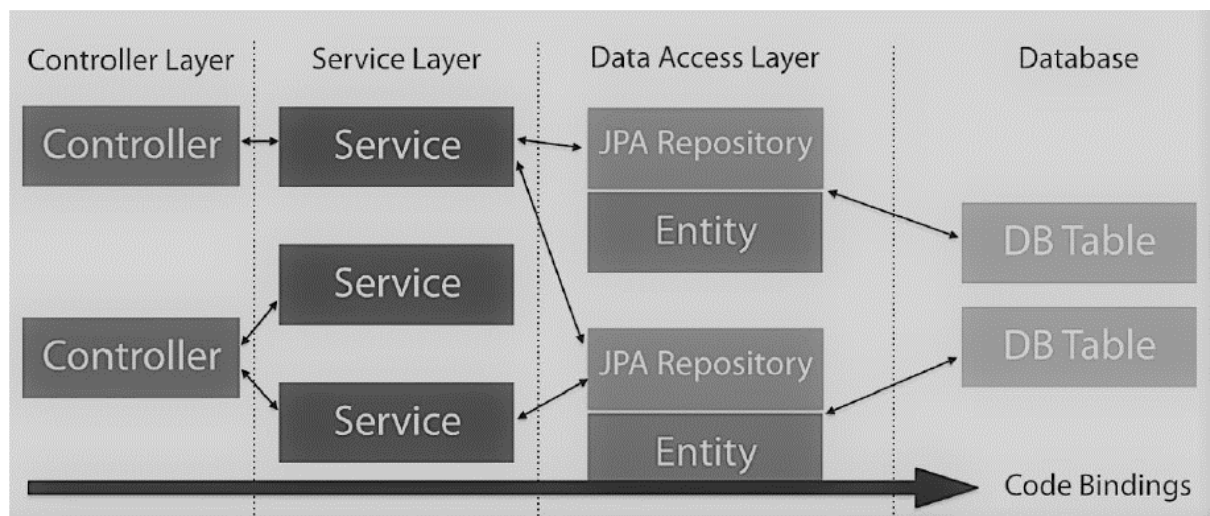
Il pattern Repository è un design pattern per gestire la persistenza dei dati.

Esso nasce per garantire una separazione tra la logica di business e la logica di persistenza. Infatti, il package Repository contiene tutto il codice relativo alla persistenza, ma nessuna logica di business.

Rende più facile la scrittura e la leggibilità del codice, favorisce il riuso, e permette di concentrarsi sull'implementazione dei requisiti anziché sull'interazione con il database.

Precisamente fornisce metodi per inserire, aggiornare e rimuovere le entità nei database ed anche metodi che istanziano ed eseguono query specifiche sul database.

L'architettura del pattern Repository è la seguente:



- Il Controller Layer contiene la logica di business tramite cui coordina l'esecuzione di una certa funzionalità o servizio del sistema;
- il Service Layer mette a disposizione operazioni di base sulle entità del sistema, come ad esempio quelle di creazione, aggiornamento, cancellazione o ricerca;
- il Data Access Layer, composto dalle interfacce JPA più le classi di entity con annotazioni Hibernate. Questo layer definisce le entità del sistema e ne permette il "mapping" con il database relazionale e le sue tabelle.

4.4 Scelte Progettuali

Dovendo sviluppare un'applicazione web, gli aspetti da definire sono stati

essenzialmente tre: frontend, backend e database.

4.1.1 Frontend

Al fine di migliorare la presentazione delle viste offerte dalla view si è optato per l'uso di Thymeleaf. Esso è un elaboratore di template JAVA XML/HTML, il quale unisce JSP ad HTML con una migliore integrazione per il framework Spring MVC.

4.1.2 Backend

È stata effettuata un'analisi relativa allo stato dell'arte dello sviluppo di applicazioni web. Tale analisi ha portato a scegliere per lo sviluppo del backend uno dei framework attualmente più usati in ambito enterprise, ovvero Spring.

4.1.3 Database

Si è scelto di utilizzare il database relazionale open source MySQL.

4.4 Framework utilizzati

4.5.1 Spring Framework

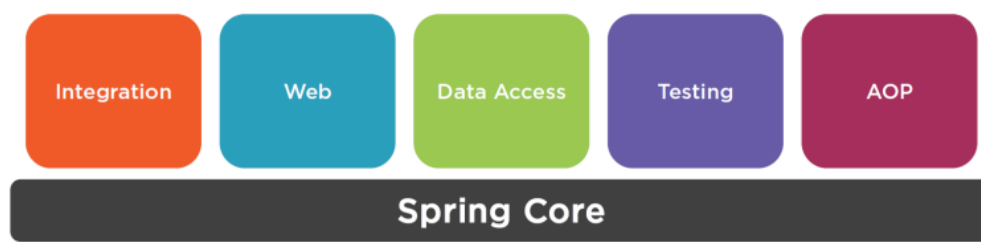
Spring Framework è un framework per lo sviluppo di moderne applicazioni enterprise Java-based. Fornisce un supporto infrastrutturale all'applicazione, sollevando gli sviluppatori da tale onere e permettendo loro di concentrarsi sulla logica di business. Un aspetto chiave del framework è sicuramente la sua modularità.

Tra i principali moduli riportati in Figura , è stato utilizzato:

- Web: offre diverse funzionalità per lo sviluppo di applicazioni web.
- Spring Web MVC: framework web costruito al di sopra della Servlet API offerta da Java. Fornisce un API di livello più alto, più semplice da usare.

In generale, un Servlet è un oggetto Java, definito nel contesto di un web server, che

riceve una richiesta e genera una risposta sulla base della richiesta ricevuta. Spring MVC è progettato secondo il pattern Front Controller, il quale prevede un servlet centrale, detto Dispatcher Servlet, che fornisce un unico punto di accesso all'applicazione. Si fa carico della ricezione di tutte le richieste, inoltrandole ai Servlet a cui sono indirizzate.



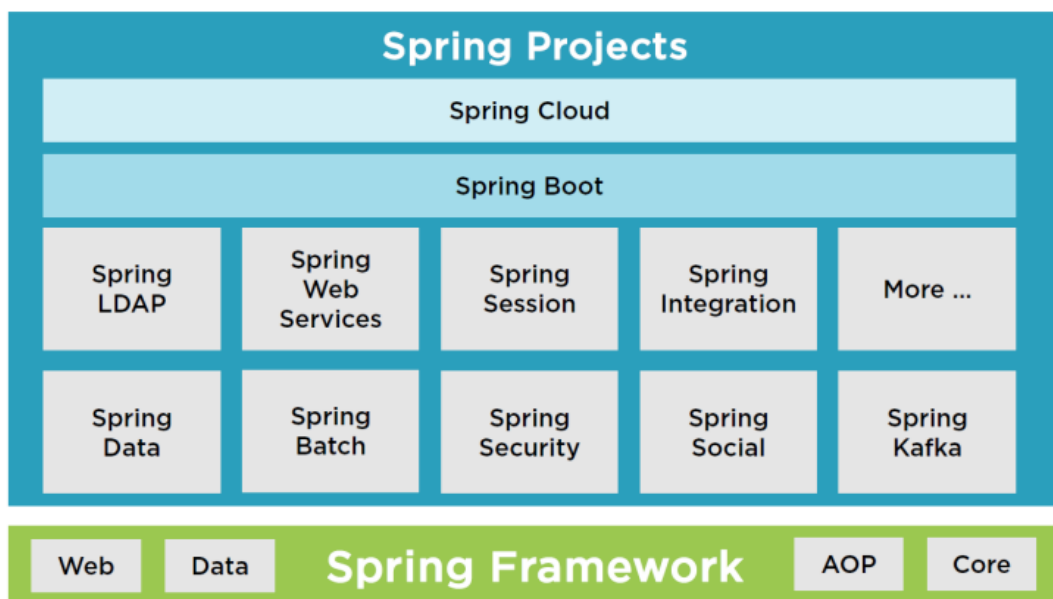
Per utilizzare le diverse funzionalità offerte da Spring Framework è sufficiente definire alcune configurazioni. A tal proposito è possibile seguire due approcci distinti: un primo approccio prevede di effettuare la configurazione tramite file XML, un secondo approccio si basa invece sul meccanismo delle annotazioni.

In conclusione, si riportano i punti di forza del framework:

- **Solidità:** è un framework sicuro e testato la cui solidità è garantita da 18 anni di sviluppo ed utilizzo; il primo rilascio risale infatti al 2002.
- **Completezza:** nato come alternativa a Java EE, negli ultimi anni Spring è cresciuto enormemente componendosi di progetti dedicati alla risoluzione di tutta una serie di problematiche tipiche del mondo enterprise.
- **Modularità:** la struttura modulare del framework, che ad oggi si compone di oltre venti sotto-progetti, ne costituisce un altro punto di forza.

- **Flessibilità:** lascia ampia libertà allo sviluppatore riguardo il percorso da seguire per arrivare al risultato desiderato, specificando pochi vincoli da rispettare.
- **Produttività:** da un lato mette a disposizione utilità per qualunque esigenza, dall'altro fornisce integrazioni con la maggior parte delle librerie e dei framework più diffusi in ambito enterprise.
- **Testabilità:** permettendo di eliminare dal codice dipendenze dirette tra le classi, consente una facile testabilità dei singoli componenti dell'applicazione.

Al di sopra di Spring Framework sono stati sviluppati diversi progetti, riportati in Figura, ognuno dei quali è orientato a fornire specifiche funzionalità. Tra i diversi progetti si riportano solo quelli utilizzati nello sviluppo dell'applicazione.



4.5.2 Spring Security

Spring Security è un progetto di Spring Framework che consente di gestire tutti gli aspetti relativi alla sicurezza di un'applicazione web Spring-based.

Le diverse funzionalità:

1) Supporto ad autenticazione e autorizzazione, entrambi aspetti chiave della sicurezza di un'applicazione.

- L'autenticazione consiste nel verificare se l'utente è davvero chi dice di essere. Tipicamente viene effettuata verificandone credenziali come username e password.
- L'autorizzazione consiste invece nel verificare a quali risorse l'utente autenticato può avere accesso. Tipicamente viene effettuata associando a ciascun utente un ruolo appropriato.

2) Protezione per le tipiche vulnerabilità a cui è esposta un'applicazione web, tra cui session fixation, clickjacking e cross site request forgery.

3) Integrazione con la Servlet API e Spring Web. Spring Security è infatti integrato con il Servlet Container, attraverso un Servlet Filter posto a monte del Dispatcher Servlet, il cui compito è quello di intercettare e filtrare le richieste dirette all'applicazione. Attraverso una vera e propria catena di filtri sono implementate tutte le funzionalità descritte nei punti precedenti.

4) Cifratura delle password degli utenti.

4.5.3 Spring Bot

Spring Boot è uno dei più recenti progetti Spring sviluppati, nato con l'obiettivo di semplificare lo sviluppo di applicazioni Spring-based, la cui configurazione può risultare complessa. Di fatto fornisce una visione categorica (opinionated) di Spring Framework ed è proprio questo ciò che riduce la complessità. In altre parole, Spring Boot definisce una configurazione di base per l'utilizzo di Spring Framework e di tutte le librerie di terze parti incluse in un progetto, semplificandone notevolmente l'avvio. In termini pratici, l'utilizzo di Spring Boot si riflette in un minor numero di annotazioni da specificare all'interno del codice.

Di seguito sono elencate alcune delle caratteristiche principali di Spring Boot:

- Configurazione automatica di Spring Framework e di librerie di terze parti, quando possibile.
- Nessuna necessità di file XML di configurazione.
- Possibilità di includere un web server/container embedded, come Tomcat o Jetty.

Di conseguenza non è necessario il deployment di file WAR (Web Application Archive) all'interno di un container esterno all'applicazione.

4.6 Diagrammi Architetture

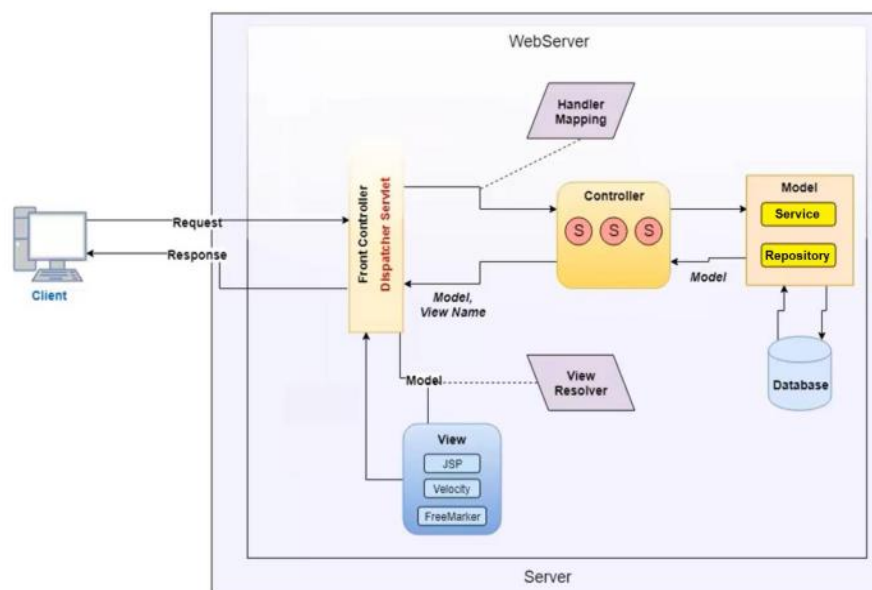
Durante lo sviluppo, e vista la natura distribuita dell'applicazione, si è utilizzato il Framework Spring MVC che implementa un MVC di tipo pull.

In VirtuaGym gli input utente corrispondono a richieste HTTP verso il server.

Queste richieste sono catturate dal Front Controller, il quale le distribuisce ai vari controller del sistema tramite la configurazione del modulo di Handler Mapping. In questo modo il Front Controller svolge una funzione di Dispatcher Servlet.

L'applicazione VirtuaGym presenta un diagramma architetturale in parte simile a quello di base, poiché sviluppato con il Framework Spring MVC, ma con delle variazioni (omissioni).

Di Seguito si è deciso di scomporre il diagramma del server in base alle viste dei vari attori del sistema, per rendere il tutto più comprensibile.



4.6.1 Vista Architettuale – Cliente Autenticato

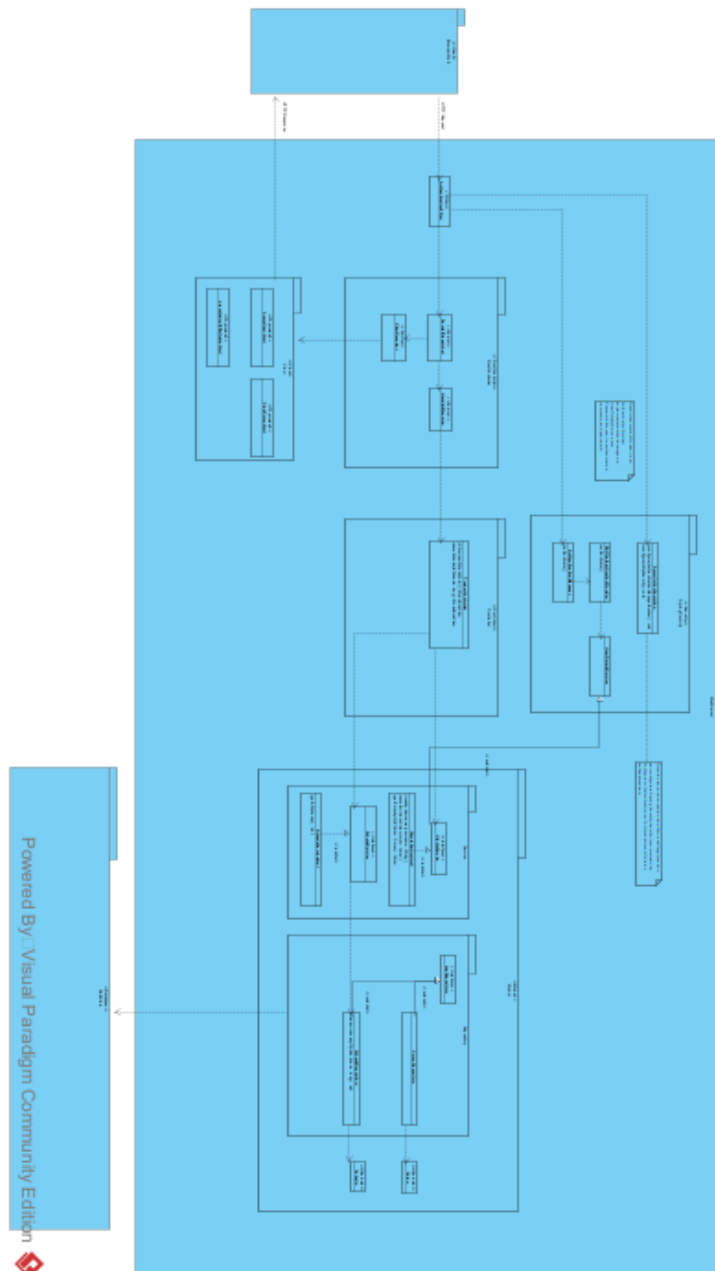


Figura: Vista Architettuale Cliente Autenticato

4.6.2 Vista Architettuale – Trainer

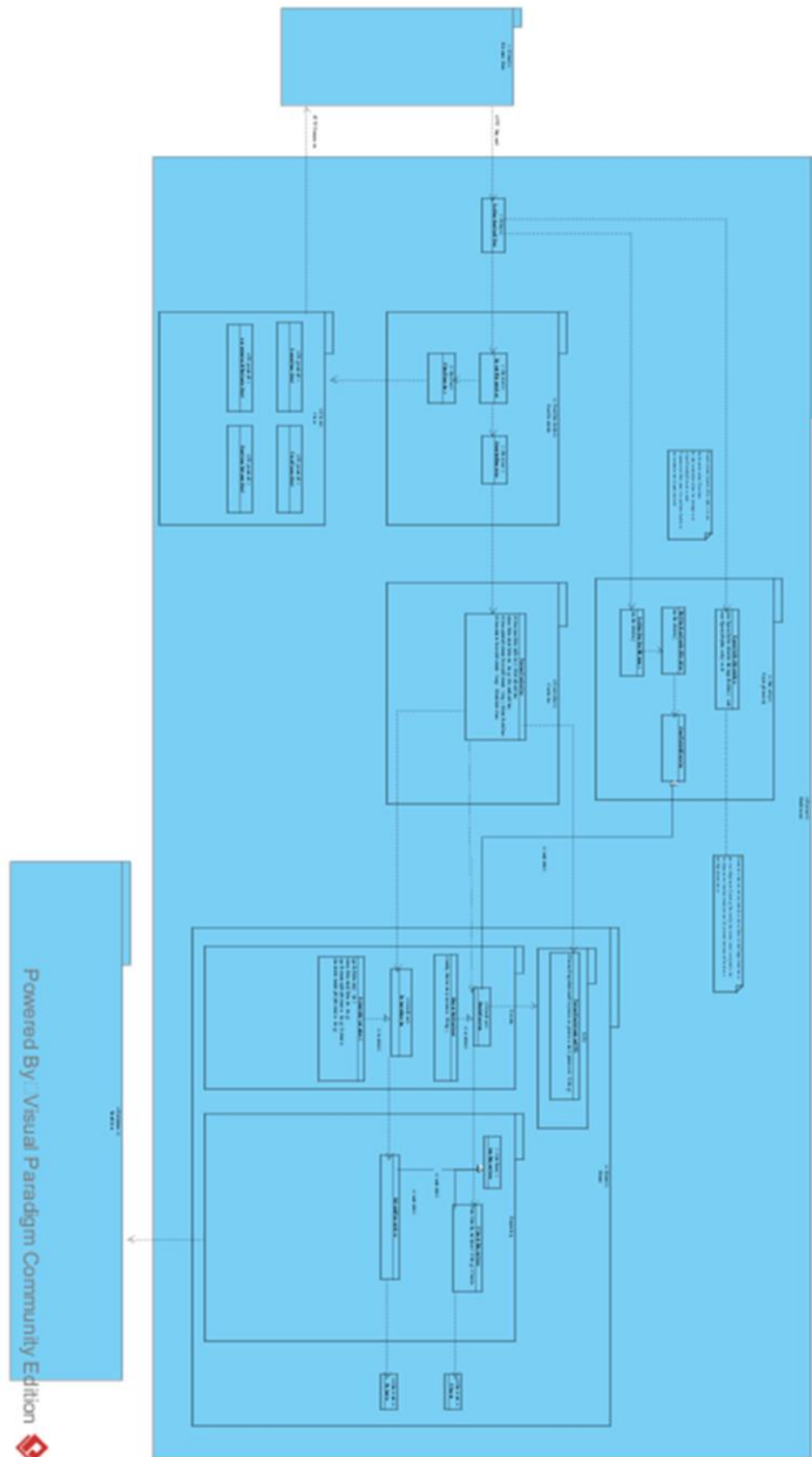


Figura : Vista Architettuale Trainer

4.7 Diagrammi di sequenza raffinati

4.7.1 Sequence Diagram Raffinato – CreaScheda

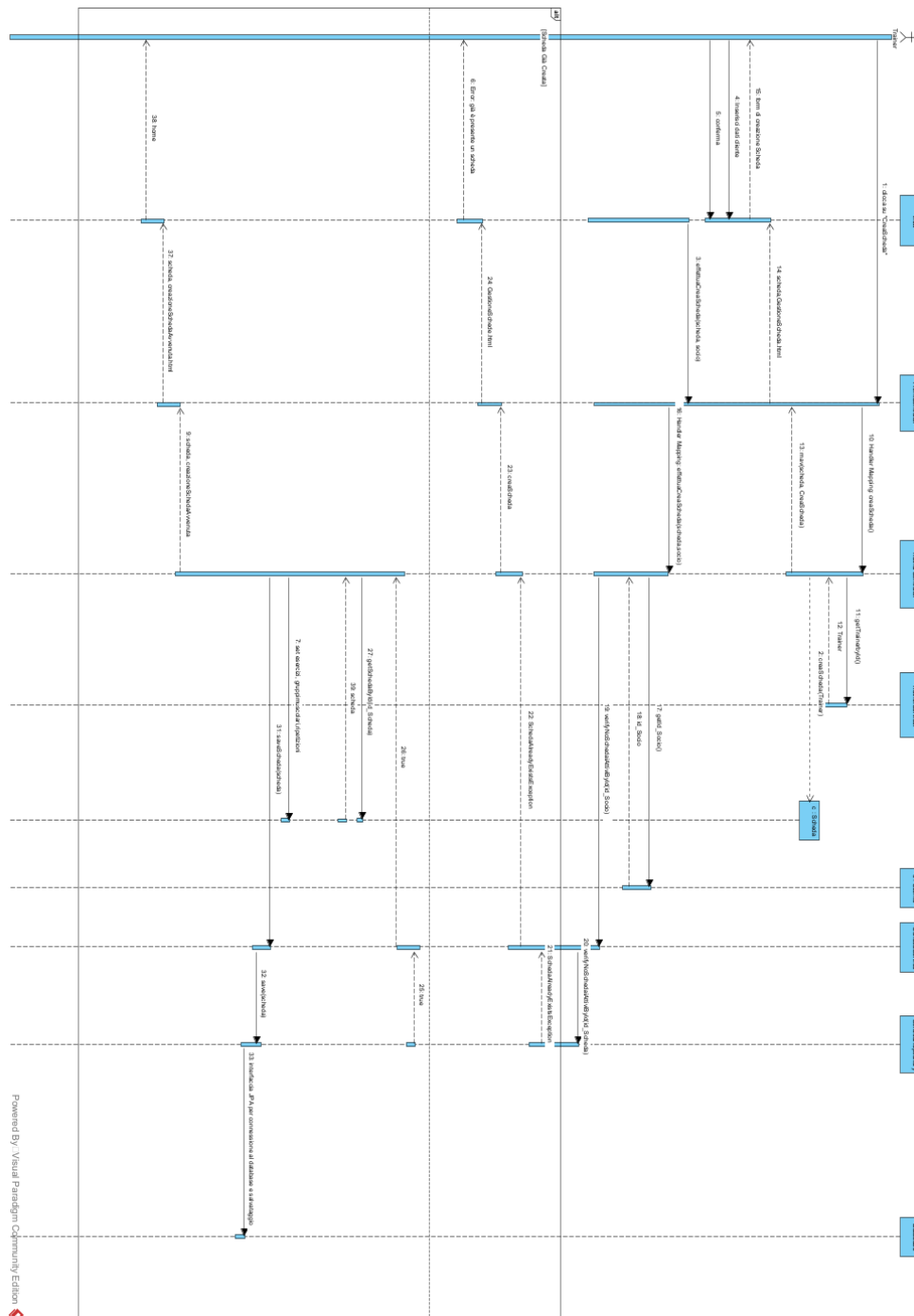


Figura: Sequence Diagram Raffinato – Crea Scheda

4.7.2 Sequence Diagram Raffinato – EliminaScheda

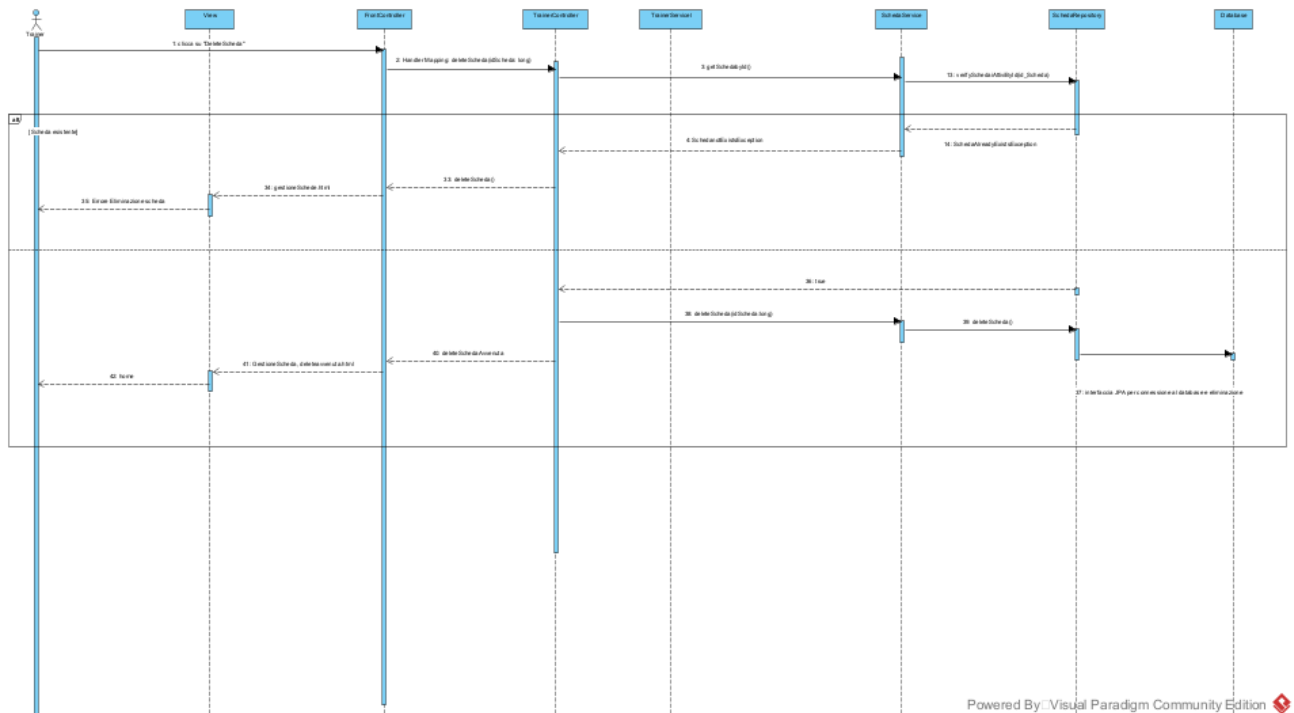


Figura: Sequence Diagram Raffinato – Elimina Scheda

4.7.3 Sequence Diagram Raffinato – Iscrizione Corso

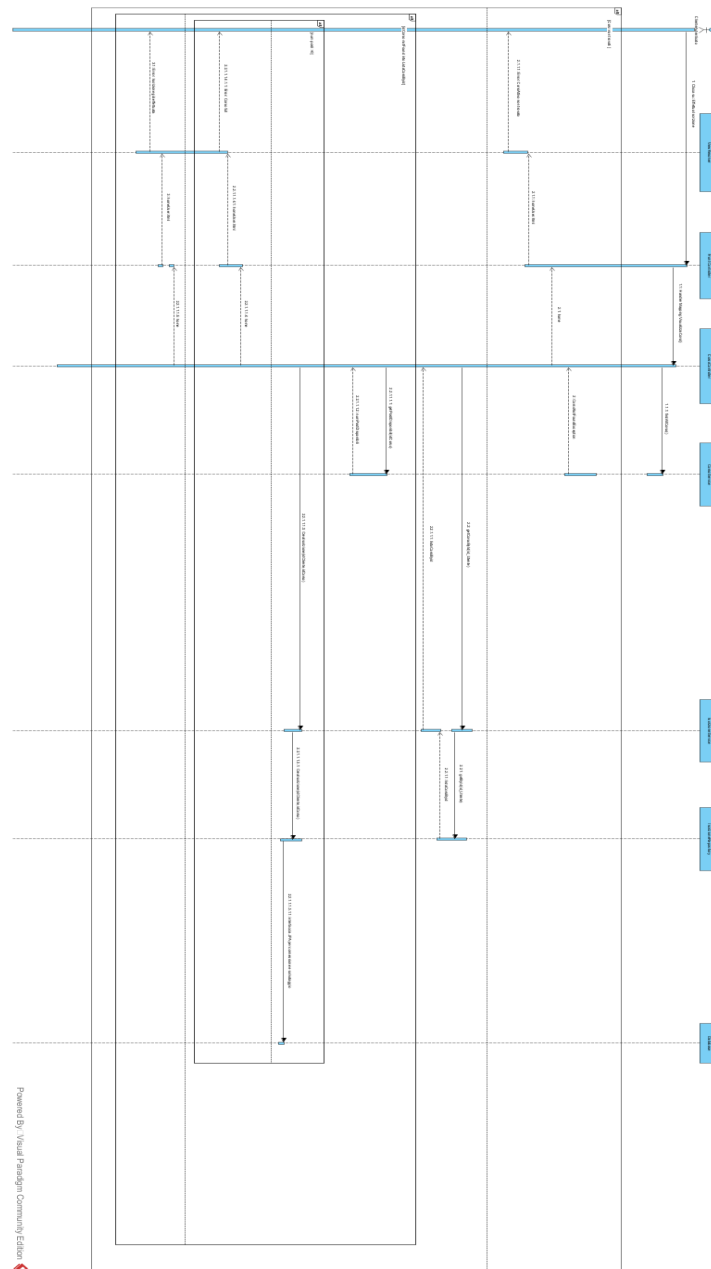


Figura : Sequence Diagram Raffinato – Iscrizione Corso



Capitolo 5: Implementazione

5.1 Documentazione

L'applicazione VirtualGym è stata sviluppata utilizzando il framework di sviluppo Spring. Precisamente è stato utilizzato:

- Spring Tool Suite 4 come ambiente di sviluppo per scrivere il software;
- Spring Boot per creare la struttura di base dell'applicazione già pronta all'uso con tanto di server associato su cui eseguire (internamente all'IDE in fase di sviluppo);
- Spring MVC per usufruire della sua architettura MVC particolarmente indicata per applicazioni web.

I linguaggi utilizzati sono stati:

- Java per la parte di back-end;
- HTML associato a Thymeleaf e CSS per la parte di front-end.

In particolare, Thymeleaf è un template engine che permette di scrivere in maniera semplice la parte dinamica di una pagina web HTML generando lui in automatico gli script relativi, in maniera implicita.

Per effettuare il deploy dell'applicazione VirtualGym occorrono un Server ed un Database installati su un PC su cui gira in esecuzione il Sistema Operativo Windows 10. Per adesso l'accesso al sistema sarà limitato all'interno del PC in cui verrà installato, per poi rendere accessibile l'applicazione anche dall'esterno tramite connessione internet in futuro.

Per poter collegare in maniera semplice ed automatica l'applicazione VirtualGym al database esterno, si è dovuto manipolare due file di configurazione del progetto. Il primo è il file pom.xml (Project Object Model) del progetto Spring, che serve appunto a definire l'identità e la struttura del progetto tramite Maven. In particolare, le aggiunte da operare sono le seguenti, nella sezione description:


```

<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <scope>runtime</scope>
</dependency>

```

Il secondo file è il file `src/main/resources/application.properties`, che gestisce sia la configurazione del framework Spring che dell'applicazione in particolare. In questo file bisogna specificare la connessione al database e le indicazioni ad Hibernate per la creazione ed aggiornamento del database:

```

1  ## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
2  spring.datasource.url = jdbc:mysql://localhost:3306/demo
3  spring.datasource.username = root
4  spring.datasource.password = root
5
6  spring.main.allow-circular-references=true
7
8  ## Hibernate Properties
9  # The SQL dialect makes Hibernate generate better SQL for the chosen database
10 spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5InnoDBDialect
11
12 # Hibernate ddl auto (create, create-drop, validate, update)
13 spring.jpa.hibernate.ddl-auto = update
14
15 debug=true
16
17 logging.level.org.hibernate.SQL=DEBUG
18 logging.level.org.hibernate.type=TRACE

```

5.2 Manuale per la configurazione e l'avvio

Come Application Web Server si è scelto di utilizzare Apache Tomcat 9.0.

Per poter effettuare il corretto deploy dell'applicazione sul server Tomcat 9.0 è necessario modificare il file `pom.xml`. In particolare, le aggiunte da operare sono le seguenti:

- nella sezione *project*:
`<packaging>war</packaging>`
- nella sezione *properties*:
`<tomcat.version>9.0.38</tomcat.version>`
- nella sezione *description*:

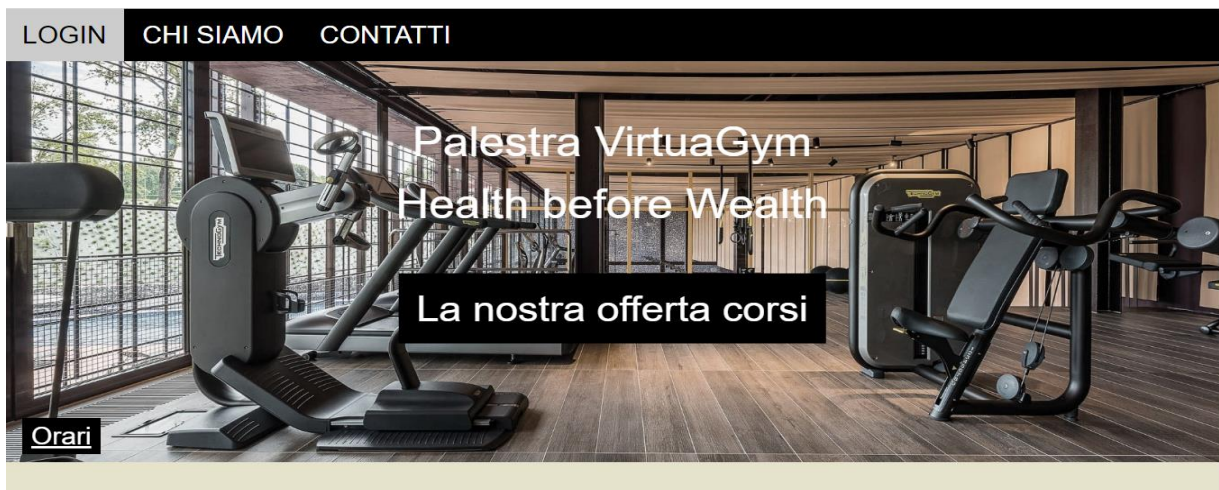
```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-tomcat</artifactId>
  <scope>provided</scope>
</dependency>
```
- nella sezione *build*:
`<finalName>2Service</finalName>`

Per utilizzare il server, occorre innanzitutto scaricarlo dal sito ufficiale <http://tomcat.apache.org/>, sezione download, versione installer per Windows. Dopodiché occorre procedere all'installazione, specificando la porta di ascolto, 8080 di default, e un account per accedere alla gestione del server. Attualmente, le credenziali provvisorie sono username="root", password="root".

Installato il server, bisogna avviare il servizio associato "Tomcat9" ed accedere alla pagina di gestione dello stesso tramite indirizzo localhost:8080/manager. Si noti che il servizio "Tomcat9" deve essere sempre in esecuzione per permettere l'esecuzione dell'applicazione. Dal pannello di management è possibile caricare il file VirtualGym.war ed effettuarne il deploy.

5.3 Esecuzione della web application

A questo punto l'applicazione può essere acceduta tramite un qualunque browser web all'indirizzo localhost:8080/VirtualGym/. In particolare, la homepage è dedicata a tutti i tipi di utenti previsti per il sistema nonché gli utenti non autenticati.



Tutti gli utenti possono visualizzare i corsi disponibili:


Home Page

La Nostra Offerta Corsi

Alla VirtuaGym ci prendiamo cura dei nostri clienti. Oltre ad un'ampia sala attrezzata i Clienti potranno usufruire di una vasta scelta di corsi tenuti da Trainer professionisti. Attualmente sono disponibili i corsi di:

Yoga, Aerobica, Zumba e SoftBox

Si ricorda che nel rispetto delle misure di prevenzione e contenimento della diffusione del coronavirus COVID-19 i clienti sono sempre tenuti a prenotarsi per lo specifico corso che intendono seguire.



Per gli utenti che si sono autenticati tramite credenziali è possibile visualizzare una pagina dove possono iscriversi ad uno dei corsi offerti dalla palestra

Lista dei Corsi

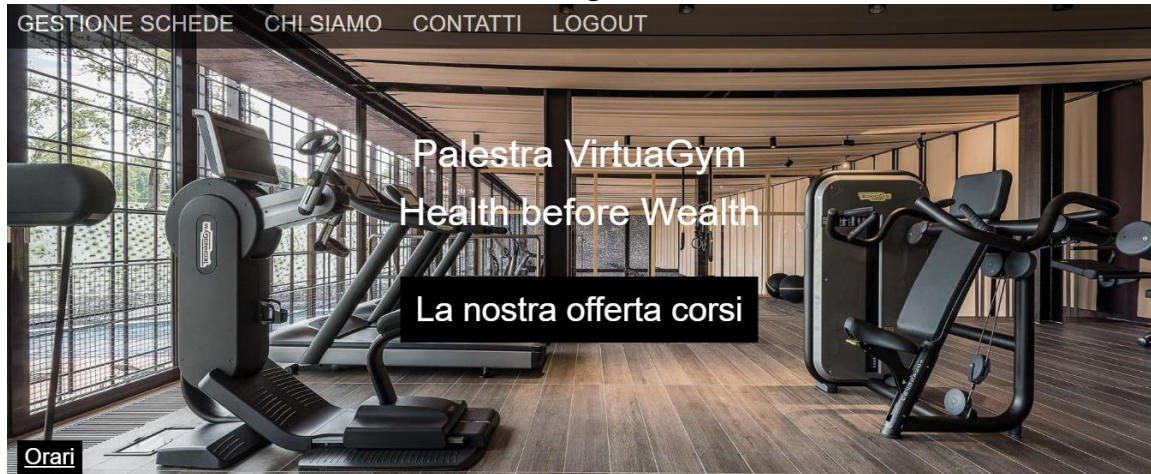
Id Corso	Nome Corso	Descrizione Corso	Numero di posti disponibili del corso	Periodo Corso	
1	zumba	si salta	1	dicembre-gennaio	Iscriviti Cancella Iscrizione
2	yoga	ti rilassai	37	gennaio-maggio	Iscriviti Cancella Iscrizione
3	softbox	prendi a pugni	12	ottobre-gennaio	Iscriviti Cancella Iscrizione
4	aerobica	area	32	aprile-luglio	Iscriviti Cancella Iscrizione

La vista degli utenti che si sono iscritti ad un corso è la seguente:

Lista degli utenti iscritti ai corsi

Id Corso	Id Utente	Id Iscrizione
2	7	1

Nel caso in cui un trainer sia ad autenticarsi la vista che gli sarà mostrata sarà la seguente:



Vista per la gestione delle schede:

Schede List

[Add Scheda](#)

Scheda nome	Scheda cognome	Scheda obiettivo	Scheda gruppi_muscolari	Scheda serie	Scheda ripetizioni	Actions
nicolo	discepolo	mettere massa	Pesi	15	20	Update Delete

La seguente vista invece sarò per la creazione/update di una scheda:

Scheda Management System

Save Scheda

[Save Scheda](#)



Capitolo 6: Testing

6.1 Test eseguiti

Nella produzione e sviluppo di sistemi software una delle fasi fondamentali è quella di testing.

Essa è determinante poiché consente di individuare carenze di correttezza, completezza e affidabilità all'interno del software sviluppato.

È buona pratica effettuare dei test alla fine di ogni iterazione al fine di controllare il corretto funzionamento del sistema dopo l'aggiunta di nuove funzionalità e di nuovi moduli.

Durante lo sviluppo dell'applicativo VirtualGym, il testing è stato svolto in concomitanza del rilascio di una nuova versione del software. Il processo di testing è stato svolto nelle prime fasi di rilascio del software senza l'utilizzo di appositi sistemi automatizzati. Nelle successive iterazioni si predisporrà il sistema in modo tale da automatizzare questa fase.

Al fine di tenere traccia del processo e dei casi di test analizzati si è fatto ricorso ad una tabella su “*Smartsheet*” contenente i casi di test inerenti ai principali scenari. All'interno del nostro foglio di lavoro sono stati etichettati come *passed*

i test per i quali i risultati effettivi combaciavano con quelli attesi.

IdTest	Azione	RisultatiAttesi	RisultatiEffettivi	Stato Test
2.1	Registrazione Nuovo Utente	Apertura di un apposito Form per la registrazione	Reinderizzamento form per la registrazione socio	<input checked="" type="checkbox"/>
2.2	Registrazione Nuovo Trainer	Apertura di un apposito Form per la registrazione del trainer	Reinderizzamento form per la registrazione trainer	<input checked="" type="checkbox"/>
2.3	Login Socio	Dopo il login è caricata la home del Socio	Reinderizzamneto home Socio	<input checked="" type="checkbox"/>
2.4	Login Trainer	Dopo il login è caricata la home del Trainer	Reinderizzamneto home Trainer	<input type="checkbox"/>
2.5	Lougt	Dopo il logout è caricata la home	Reinderizzamneto home	<input checked="" type="checkbox"/>
2.6	Click Menu navbar Chi siamo	Reinderizzamneto sezione Chi Siamo dalla pagina home	Reinderizzamento Chi siamo	<input checked="" type="checkbox"/>
2.7	Click Menu navbar Contatti	Reinderizzamneto sezione Contatti dalla pagina home	Reinderizzamneto Contatti	<input checked="" type="checkbox"/>
2.8	Click Orari	Reinderizzamneto sezione Orari dalla pagina home	Reinderizzamento Orari	<input checked="" type="checkbox"/>
2.9	Click Offerta Corsi	Reinderizzamneto alla pagina che presenta i corsi	Apertura pagina contenete l'offerta Corsi	<input checked="" type="checkbox"/>
2.10	Click Iscrizione ai Corsi	Reinderizzamneto alla pagina che consente la prenotazionei corsi	Reinderizzamneto pagine per la prenotazione	<input checked="" type="checkbox"/>
2.11	Iscrizione Corso	Decremneto Conteggio posti disponibili e iscrizione al corso	decremento Contatore ma memorizzazione non corretta	<input type="checkbox"/>
2.12	Cancellazione Iscrizione Cors	Visualizzazione dei corsi presenti e aggiornamento dei posti	Aggiornament dei Posti	<input type="checkbox"/>
2.13	Gestione Schede Trainer	Apertura di una pagina apposita per la gestione delle schede	Visualizzazione,modifica e eliminazione riuscite	<input checked="" type="checkbox"/>
2.14	Creazione Scheda	Aperura di una pagina contenente un form per le info della scheda	Reinderizzamneto Corretto Pagina e creazione corretta	<input checked="" type="checkbox"/>
2.15	Visualizzazione Iscrizione ai c	Utente loggato puo visualizzare i corsi a cui è iscritto	Reinderizzamnto pagina ma visualizzazione errata	<input type="checkbox"/>

Figura 42: Test Accettazione

6.1 Testing iniziali

Durante lo sviluppo del codice ci si è concentrati sulle fonti di errore più comuni che potevano verificarsi nella funzionalità di iscrizione al corso e cancellazione ad esso nel caso in cui un utente non possa registrarsi per posti insufficiente oppure se vuole cancellare l'iscrizione ad un corso al quale non risulta essere iscritto.

Al fine di risolvere questo problema sono create e configurate delle eccezioni di tipo *throwable* nel package “*exceptions*”. Tali eccezioni vengono lanciate o meno in base al risultato della query **HQL** fatta sul database, in particolare riguardano la classe “*CorsoRepositoryImpl*”.

Oltre a verificare il corretto funzionamento delle eccezioni, nella prima fase di test ci si è concentrati sul corretto funzionamento dell'iscrizione al corso in

presenza di situazioni non leciti. Questo ha permesso di individuare degli errori all'interno del codice, poiché in una prima fase non era stata gestita correttamente il conteggio dei posti disponibili, ciò implicava che

- Un socio poteva iscriversi anche se i posti effettivi erano esauriti
- Un utente poteva disiscriversi ad un corso anche se effettivamente non era iscritto

Prima di effettuare il testing per l'iscrizione ad un corso sono state testate anche le funzioni di Login e Logout realizzate tramite il framework Spring Security.

Pagina Login Socio

username e/o password invalide.

Username :

Password:

Log In

Nuovo Cliente? [Registrati qui](#)



Securedby SpringSecurity

6.2 Test successivi

Terminata la prima fase di test, sono state apportate correzioni e modifiche a codice sulla base dei test che non soddisfacevano i risultati attesi.

Dunque, al termine delle modifiche è stata eseguita un'ulteriore fase di test mirata per validare tutti le funzionalità del sistema, con l'obbiettivo di scoprire eventuali funzionalità non gestite dal software.

Infatti, oltre ai test sugli input di “EffettuaIscrizione”, sono stati testati gli input della funzionalità “UpdateScheda” e si è provato questa volta a verificare che il sistema permettesse di recuperare correttamente la scheda dal nostro database, modificarla e salvare correttamente le modifiche apportate alla scheda stessa.

Questa fase di testing ha evidenziato un problema inerente ad una sbagliata gestione delle *annotation* di java; Infatti, in questa prima fase di test non era possibile recuperare correttamente le informazioni dal database. Questo test è stato successivamente ripetuto dopo aver apportato le giuste modifiche al codice, l'esito di questo test è stato quindi positivo.

6.3 Test per le future iterazioni

Nelle future iterazioni, si andranno quindi a testare tutte le nuove funzionalità che saranno introdotte all'interno dell'applicativo. Ogni funzionalità introdotta dovrà essere opportunamente testata, analizzata e successivamente se richiesto corrette se dovesse presentare un comportamneto diverso da quello atteso. Dopo aver

testato tutte le nuove componenti reintrodotte nel sistema verrà effettuato una validazione del sistema complessivamente, provando tutti i possibili casi che potrebbero far incombere in una situazione non gestiti