Iterazione 2

1.0 Introduzione

La seconda iterazione del progetto TeamManager si focalizza sull'ampliamento delle funzionalità analitiche, attraverso la raccolta e l'elaborazione di dati utili a supportare le decisioni dell'allenatore. Dopo aver consolidato, nella prima iterazione, la gestione di base dei giocatori, della rosa, degli eventi e delle disponibilità, questa nuova fase si propone di introdurre strumenti avanzati per la registrazione dei parametri prestazionali e fisici, la personalizzazione degli allenamenti e il confronto tra singoli giocatori. Queste funzionalità mirano a potenziare la componente strategica dell'applicazione, fornendo un valido supporto nell'analisi delle performance individuali e nella pianificazione tecnica della squadra. In particolare, l'Iterazione 2 prevede l'analisi e l'implementazione dei seguenti casi d'uso:

- **UC6: Gestisci progressi giocatore** per registrare e consultare i dati relativi ai parametri fisici e prestazionali dei singoli giocatori durante gli allenamenti e le amichevoli;
- **UC9: Suggerisci sessioni mirate** che consente di generare piani di allenamento personalizzati in base alle lacune evidenziate nei dati raccolti dei singoli giocatori;
- **UC10: Confronta giocatore**, per effettuare un'analisi comparativa delle performance fisiche tra due giocatori selezionati.

Tutte le nuove funzionalità manterranno la struttura e la coerenza dell'architettura definita nella prima iterazione, facendo del pattern **Facade Controller** tramite la classe **TeamManager**. L'obiettivo di questa iterazione è rafforzare l'affidabilità del sistema e la sua capacità di fornire dati e supporto decisionale basato su dati oggettivi e analisi automatizzate.

1.1 Affinamento delle Regole di Dominio

Durante questa fase, le regole di dominio del sistema sono state oggetto di una revisione approfondita per migliorarne la coerenza e la robustezza. In particolare: Sono state eliminate alcune regole considerate superflue. Queste erano precedentemente legate a funzionalità specifiche, permessi utente o requisiti di sicurezza, ma non contribuivano direttamente alla logica di dominio essenziale. Sono state introdotte nuove regole, significativamente più stringenti e focalizzate sull'unicità del giocatore sia come entità che come elemento interno alla rosa e aspetti legati alla gestione delle statistiche. Di seguito le regole di dominio aggiornate:

ID	Regola	Modificabilità	Sorgente
R1	Ogni giocatore registrato non può essere rimosso dall'allenatore senza una motivazione valida (es. infortunio prolungato, uscita dalla squadra, comportamento inadeguato, etc.).	Media, potrebbe essere necessario prevedere la gestione di ex-giocatori.	Regola interna della squadra.
R2	Il giocatore deve avere un identificativo univoco, non sono consentiti duplicati e non può essere inserito più volte all'interno della stessa rosa.	Bassa, strettamente legata alla gestione della rosa.	Regola interna del sistema.
R3	La rosa può contenere un massimo di **22 giocatori**, qualora la rossa fosse piena nessun altro giocatore potrà essere inserito al suo interno	Bassa, strettamente legata alla gestione della rosa.	Regola interna della squadra.
R4	Un giocatore può avere uno e un solo **stato** tra: **Disponibile**, **Infortunato** o **Sospeso**.	Media, nuove categorie potrebbero essere aggiunte in futuro.	Regola interna del sistema.
R5	Non ci possono essere due giocatori con lo stesso numero di maglia all'interno della rosa	Bassa, strettamente legata alla gestione della rosa.	Regola interna della squadra.
R6	Ogni giocatore inserito nella rosa deve avere un ruolo assegnato dall'enumerazione predefinita (**PORTIERE, DIFENSORE, ecc**.).	Media, nuove categorie potrebbero essere aggiunte in futuro.	Regola interna del sistema.

R7	Più eventi possono essere pianificati nello stesso giorno, ma non possono **sovrapporsi**.	Bassa, modificabile solo con una revisione del modello degli eventi.	Regola interna del sistema per evitare conflitti.
R8	L'analisi delle performance può essere effettuata solo su **giocatori esistenti in rosa**.	Bassa, strettamente legata alla gestione della rosa.	Regola tecnica interna.
R9	Le statistiche di performance relative a un evento possono essere registrate solo per i giocatori che hanno confermato la loro presenza e vi hanno effettivamente partecipato.	bassa, strettamente legata alla partecipazione dei giocatori al suddetto evento	Regola tecnica interna.

Di seguito è presentata la cronologia delle varie versioni delle regole di dominio del sistema, con un focus sulle eventuali modifiche effettuate a cui il sistema dovrà eventualmente adeguarsi

Versione	**data**	**Descrizione**	**Autori**
1.0	02/06/2025	Prima bozza presente all'interno del file "TeamManager"	Antonio Nicolò Scarvaglieri, Giuseppe Ravesi, Vincenzo Venezia
2.0	20/06/2025	Aggiunta R2,R3,R5.R6, e eliminazione regole superflue	Antonio Nicolò Scarvaglieri, Giuseppe Ravesi, Vincenzo Venezia

2.0 Fase di Analisi

2.1 Estensione del modello di Dominio

- Eseguito un refactoring di **rinominazione** per quanto riguarda attributi non correttamente definiti, esempio num_Maglia rinominato numMaglia o Nome

rinominato correttamente nome, pratica essenziale per migliorare la qualità interna del codice in un contesto di sviluppo software.

- Eseguito un **Refactoring di rinominazione** della classe Giocatore_Rosa in GiocatoreInRosa un passo fondamentale per migliorare la leggibilità e la chiarezza del codice, rendendo il nome della classe più esplicito e più aderente al suo significato nel contesto della programmazione orientata agli oggetti.
- Durante la progettazione ci si è resi conto che L'attributo Num_Maglia non è un attributo del Giocatore in sé ma bensì di un Giocatore all'interno di una rosa. È stato eseguito un **Refactoring di movimento** spostando l'attributo Num_Maglia dalla classe Giocatore alla classe GiocatoreInRosa ciò migliora l'incapsulamento e la coerenza del modello

Nella fase di Analisi della seconda iterazione, il Modello di Dominio è stato esteso significativamente per supportare nuove funzionalità, in particolare il tracciamento dettagliato delle statistiche e una migliore definizione concettuale di alcune entità.

Sono state introdotte le seguenti classi ed enumerazioni:

- **Statistica:** Una classe astratta che funge da base per le statistiche raccolte durante i diversi tipi di eventi.
- **StatisticaAllenamento:** Generalizzazione della classa statistica modellata per registrare dati specifici relativi agli allenamenti a cui possono partecipare i vari giocatori presenti in rosa.
- **StatisticaAmichevole:** Generalizzazione della classa statistica modellata per registrare dati specifici relativi alle amichevoli a cui possono partecipare i vari giocatori presenti in rosa.

Enumeration Stato: Rappresentano, rispettivamente, gli stati di un giocatore:

- DISPONIBILE
- SOSPESO
- INFORTUNATO

Enumeration Ruolo: Rappresentano, rispettivamente, i ruoli che un giocatore può ricoprire:

- PORTIERE
- DIFENSORE
- CENTROCAMPISTA
- ATTACCANTE

Inoltre, sono stati aggiornati attributi e associazioni di classi preesistenti:

- Sostituzione di alcuni attributi String con enumeration; Una delle modifiche più rilevanti è stata la sostituzione di attributi di tipo string con tipi enumeration per una maggiore robustezza e manutenibilità del codice.
- Nella classe GiocatoreInRosa, l'attributo ruolo è ora un'enumerazione Ruolo. Questa scelta garantisce che solo valori predefiniti e validi possano essere assegnati, a differenza di un campo stringa libero.
- Analogamente sempre nella classe GiocatoreInRosa, l'attributo status è stato trasformato in un'enumerazione Stato. Questo permette di gestire in modo controllato e scalabile la condizione dei singoli giocatori.
- Eliminato l'attributo ruoloPreferito sulla classe Giocatore poiché ritenuto ridondante poiché già presente in GiocatoriInRosa.

Queste modifiche non solo arricchiscono il modello con nuove funzionalità, ma migliorano anche la robustezza e la chiarezza concettuale del sistema, gettando le basi per future implementazioni più complesse e strutturate.

2.2 Implementazione regole di dominio

In questa seconda iterazione ci si è concentrati anche sull'implementazione delle **regole di dominio** descritte nel paragrafo *1.1 – Affinamento delle Regole di Dominio*. In particolare, sono state affrontate:

- la **gestione dell'unicità del giocatore** (*R2*), per impedire che vi siano giocatori duplicati all'interno del sistema, con duplicati si intende giocatori con lo stesso indirizzo e-mail.
- il **controllo sul numero massimo di giocatori presenti in rosa** (*R3*); Una Rosa completa sarà composta da un totale di 22 GiocatoriInRosa
- la **verifica della non sovrapposizione tra eventi** (*R7*), per garantire che due eventi non si svolgano nello stesso giorno e orario.
- ci si è assicurati che **non si inserisca un giocatore in rosa con lo stesso numeroMaglia** di un altro già presente (R5).

Tali vincoli sono stati implementati tramite l'introduzione di **eccezioni specifiche**, ovvero **classi dedicate alla segnalazione di errori legati alla violazione delle regole di dominio**. Nello specifico:

- la classe GiocatoreDuplicatoException.java viene utilizzata per segnalare il tentativo di inserimento di un giocatore già esistente nel sistema;
- la classe RosaPienaException.java viene chiamata quando si tenta di superare il numero massimo consentito di giocatori nella rosa;
- la classe EventoSovrappostoException.java gestisce i casi in cui si tenta di pianificare un evento che si sovrappone temporalmente a un altro già presente in calendario.
- La classe NumeroMagliaDuplicatoException.java impedisce l'inserimento di un Giocatore che abbia lo stesso numMaglia di un Giocatore già presente in Rosa

Queste eccezioni incapsulano le informazioni relative all'errore e vengono **lanciate automaticamente** dal sistema qualora si verifichi una delle condizioni sopra indicate. In questo modo, il rispetto delle regole di dominio viene garantito in fase di esecuzione, contribuendo alla coerenza e affidabilità dell'applicazione.

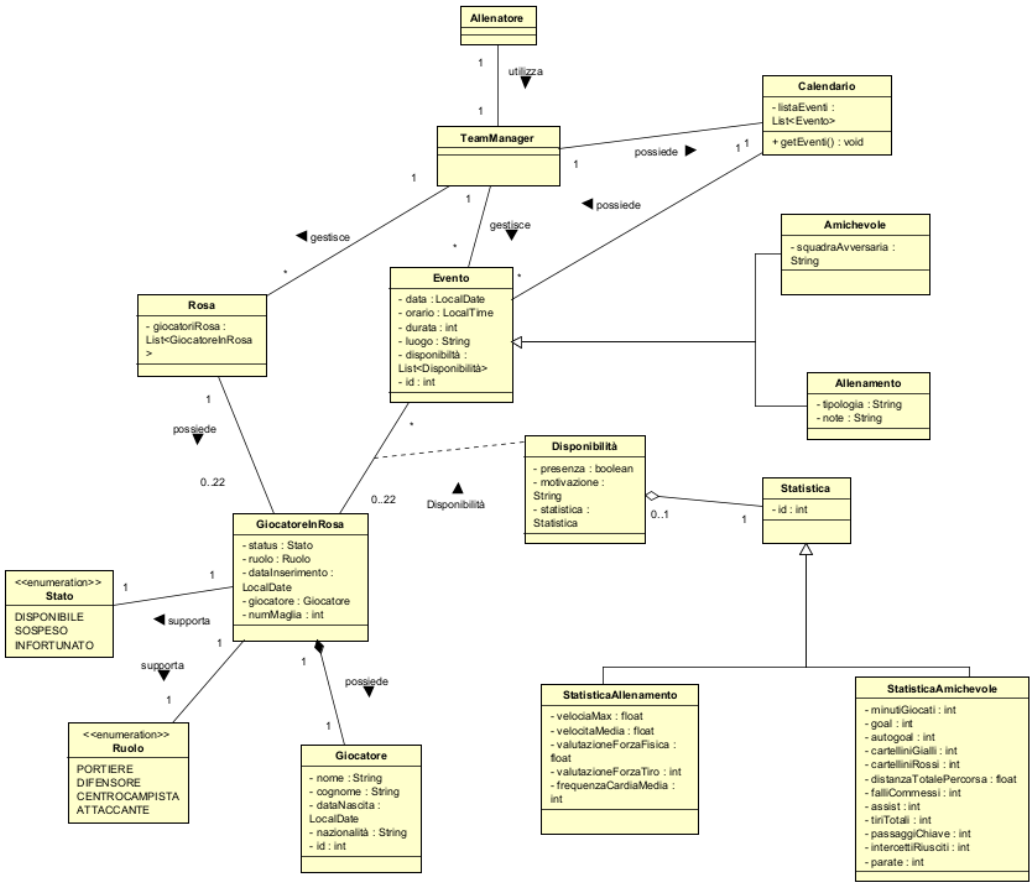
2.3 Classi concettuali derivate dai casi d'uso e modello di dominio

Come nella precedente iterazione anche in questa fase individuiamo le principali **classi concettuali** coinvolte nei casi d'uso previsti per questa seconda iterazione che saranno successivamente trasformate in componenti software all'interno del modello di dominio. Di seguito sono elencate le classi concettuali emerse dall'analisi dei casi d'uso **UC6, UC9 e UC10**, accompagnate da una breve descrizione:

Classe	Descrizione
Giocatore	Rappresenta un individuo iscrivibile alla squadra. Contiene informazioni anagrafiche (nome, cognome, e-mail), tecniche (ruolo).
GiocatoreInRosa	Rappresenta il giocatore pronto ad essere inserito all'interno della rosa a cui vengono associati parametri esclusivi.
Rosa	Contiene i giocatori attualmente attivi nella squadra. È gestita dall'allenatore ed è derivata dall'archivio generale.
Evento	Rappresenta una sessione di squadra (allenamento o partita amichevole), con informazioni su data, orario, luogo e obiettivo.
Disponibilità	Associa un giocatore a un evento specifico, indicandone la disponibilità (presente/assente) e una possibile motivazione.
Allenatore	Figura responsabile della gestione della rosa, della creazione degli eventi e della consultazione delle disponibilità.
Statistica	Raccoglie i parametri individuali del giocatore in seguito a una sua partecipazione ad un evento.

Classe	Descrizione
TeamManager	Rappresenta il sistema TeamManager

Da cui tenendo conto dei vari attributi e associazioni di ogni entità, si è generato il seguente modello di dominio:



Quando l'allenatore registra i progressi di un giocatore, il sistema verifica preventivamente la sua effettiva partecipazione all'evento selezionato, sfruttando la relazione già esistente tra GiocatoreInRosa ed Evento, mediata dalla classe di associazione Disponibilità. Solo nel caso in cui il giocatore risulti **presente** all'evento, è consentita la registrazione delle statistiche. Tali statistiche si differenziano in base alla **tipologia di evento**, portando alla creazione di due generalizzazioni della classe Statistica:

- StatisticaAllenamento, per gli eventi di tipo **allenamento**;

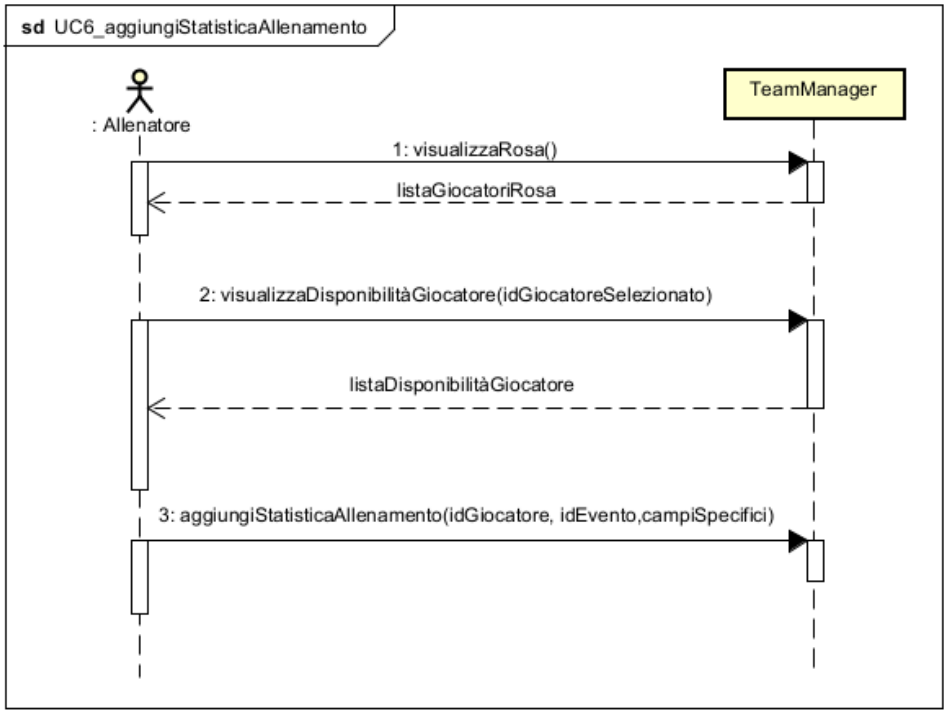
- StatisticaAmichevole, per gli eventi di tipo **amichevole**.

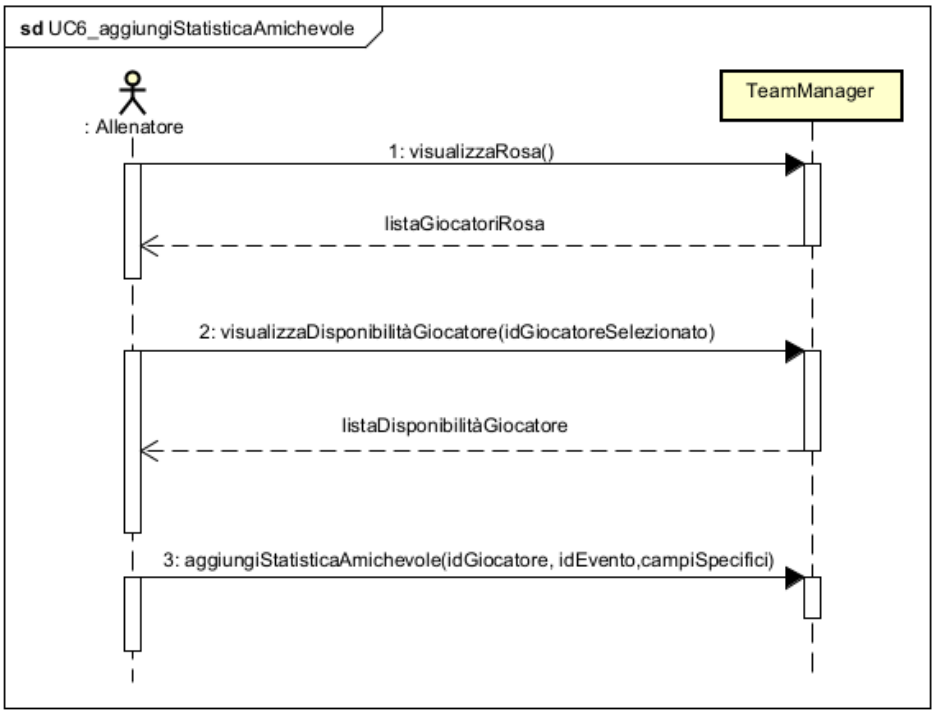
A partire da questi dati, il sistema mette a disposizione dell'allenatore una serie di **funzionalità avanzate di supporto decisionale**. In particolare, il caso d'uso **"Suggerisci sessioni mirate"** analizza le prestazioni registrate per ciascun giocatore, identificando **eventuali carenze** rispetto a soglie di riferimento predefinite. In base al **ruolo** del giocatore e ai **punti deboli rilevati**, il sistema genera una o più sessioni di allenamento **personalizzate**, mirate al miglioramento delle aree critiche. Analogamente, il caso d'uso **"Confronta giocatore"** consente all'allenatore di selezionare due membri della rosa e ottenere un **confronto dettagliato** tra i loro parametri fisici e tecnici. L'analisi si basa sull'**aggregazione delle statistiche raccolte** nei diversi eventi, fornendo una panoramica comparativa che evidenzia punti di forza e aspetti da migliorare. Entrambe le funzionalità rappresentano **strumenti strategici per la gestione della squadra**, supportando l'allenatore nelle scelte operative, nella pianificazione tecnica e nella valorizzazione delle risorse disponibili.

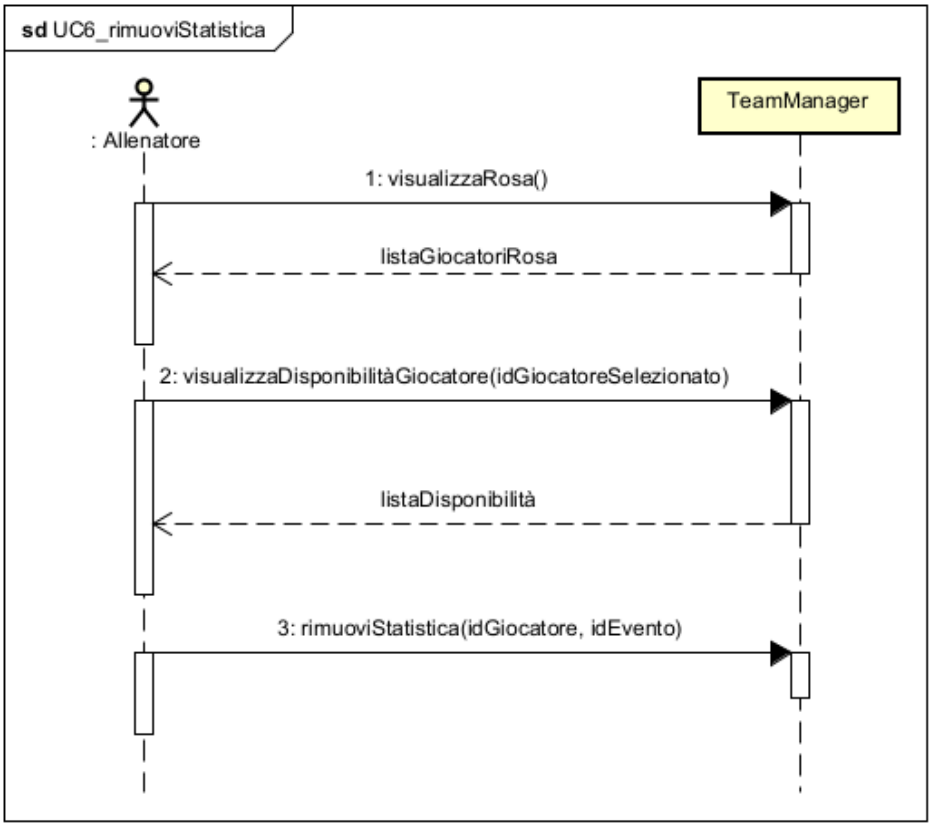
2.4 Diagrammi di Sequenza di Sistema e Contratti delle Operazioni

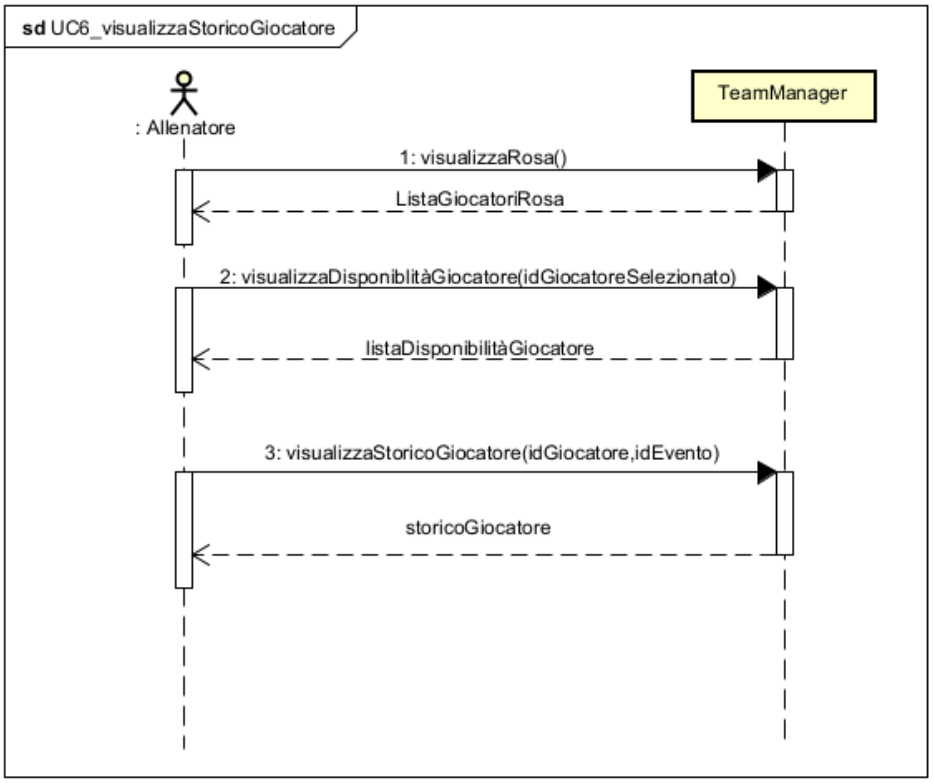
UC6 – Gestisci Progressi Giocatore

I seguenti Diagrammi di Sequenza di Sistema illustrano le interazioni I/O dell'utente con il sistema TeamManager relative al Caso d'Uso 6 con i vari scenari di successo:




sd UC6_aggiungiStatisticaAmichevole
: Allenatore
TeamManager
1: visualizzaRosa()
listaGiocatoriRosa
2: visualizzaDisponibilitàGiocatore(idGiocatoreSelezionato)
listaDisponibilitàGiocatore
3: aggiungiStatisticaAmichevole(idGiocatore, idEvento,campiSpecifici)


sd UC6_rimuoviStatistica
: Allenatore
TeamManager
1: visualizzaRosa()
listaGiocatoriRosa
2: visualizzaDisponibilitàGiocatore(idGiocatoreSelezionato)
listaDisponibilità
3: rimuoviStatistica(idGiocatore, idEvento)



Contratti delle operazioni UC6

I contratti delle operazioni descrivono i principali metodi identificati nei SSD di UC6:

CONTRATTO 1 UC6: Gestisci Progressi Giocatore

- **Operazione:** visualizzaDisponibilitàGiocatore(idGiocatoreSelezionato)
- **Riferimento:** UC6: Gestisci Progressi Giocatore
- **Pre-condizione:**
  1. L'allenatore ha effettuato correttamente l'accesso al sistema.
  2. Dal menu principale l'allenatore ha selezionato l'opzione rosa dal menu.
  3. L'allenatore visualizza correttamente la lista dei vari Giocatori presenti in Rosa.
  4. Il file contenente i vari giocatori presenti in rosa è accessibile in lettura
  5. L'allenatore può selezionare un Giocatore per consultarne le relative disponibilità.
- **Post-condizione:**
  1. L'allenatore visualizza correttamente a schermo la Lista delle Disponibilità associate al giocatore selezionato ovvero l'elenco dei relativi eventi al quale ha partecipato compresi i relativi dettagli.
  2. Nessuna modifica viene apportata ai dati.

CONTRATTO 2 UC6: Gestisci Progressi Giocatore

- **Operazione:** aggiungiStatisticaAmichevole(idGiocatore, idEvento,campiSpecifici)

- **Riferimento:** UC6: Gestisci Progressi Giocatore
- **Pre-condizione:**
    1. Tutti i campi richiesti sono compilati correttamente.
    2. L'allenatore visualizza correttamente la lista delle disponibilità.
    3. Il giocatore esiste nella rosa e ha partecipato all'evento.
    4. Non esiste già una registrazione di statistiche per quel giocatore in quello specifico evento.
- **Post-condizione:**
    1. Viene creata un'istanza di StatisticaAmichevole contentamente i vari parametri.
    2. L'istanza di StatisticaAmichevole è associata al giocatore e all'evento specifico a cui ha partecipato.

CONTRATTO 3 UC6: Gestisci Progressi Giocatore

- **Operazione:** aggiungiStatisticaAllenamento(idGiocatore, idEvento,campiSpecifici)
- **Riferimento:** UC6: Gestisci Progressi Giocatore
- **Pre-condizione:**
    1. Tutti i campi richiesti sono compilati correttamente.
    2. L'allenatore visualizza correttamente la lista delle disponibilità.
    3. Il giocatore esiste nella rosa e ha partecipato all'evento.
    4. Non esiste già una registrazione di statistiche per quel giocatore in quello specifico evento.
- **Post-condizione:**
    1. Viene creata un'istanza di StatisticaAllenamento contentamente i vari parametri.
    2. L'istanza di StatisticaAllenamento è associata al giocatore e all'evento specifico a cui ha partecipato.

CONTRATTO 4 UC6: Gestisci Progressi Giocatore

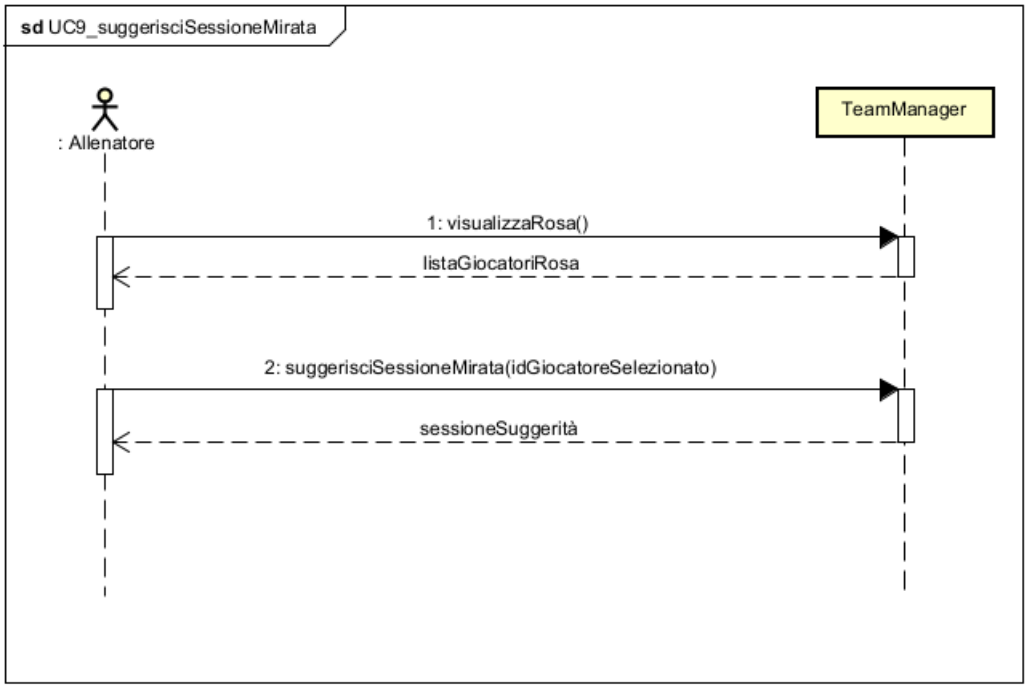
- **Operazione:** rimuoviStatistica(idGiocatore, idEvento)
- **Riferimento:** UC6: Gestisci Progressi Giocatore
- **Pre-condizione:**
    1. L'allenatore visualizza correttamente la lista delle disponibilità.
    2. Esiste un'istanza di Statistica.
    3. L'istanza di statistica è associata ad uno specifico giocatore e ad uno specifico evento.
- **Post-condizione:**
    1. L'istanza di Statistica viene eliminata correttamente.

CONTRATTO 5 UC6: Gestisci Progressi Giocatore

- **Operazione:** visualizzaStoricoGiocatore(idGiocatore,idEvento)
- **Riferimento:** UC6: Gestisci Progressi Giocatore
- **Pre-condizione:**
    1. Il giocatore esiste nella rosa e ha partecipato a uno o più eventi.
    2. L'allenatore visualizza correttamente la lista delle disponibilità.
- **Post-condizione:**
    1. L'allenatore visualizza lo storico del giocatore.
    2. Nessuna modifica viene apporta ai dati.

UC9 – Suggerisci Sessioni mirate

I seguenti Diagrammi di Sequenza di Sistema illustrano le interazioni I/O dell'utente con il sistema TeamManager relative al Caso d'Uso 9:



Contratti dell'operazioni UC9

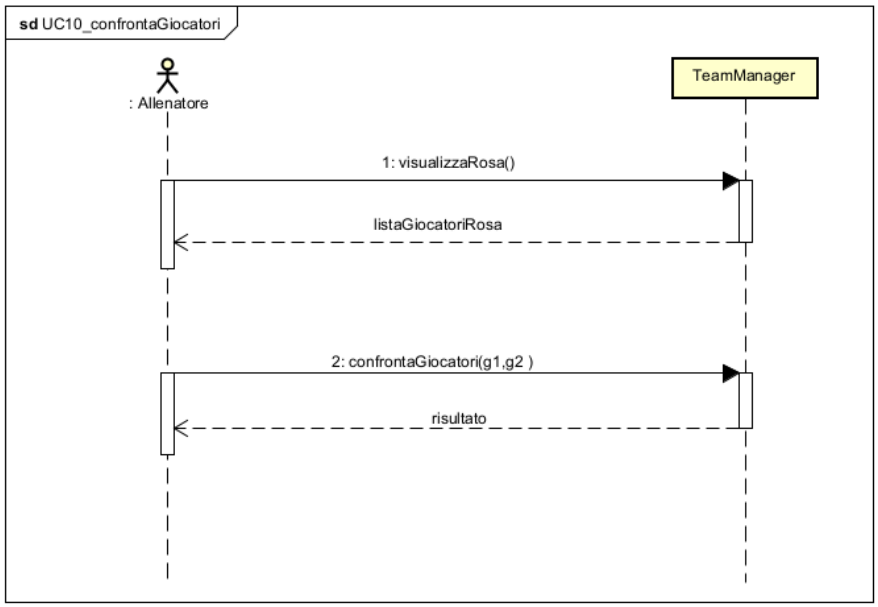
I contratti delle operazioni descrivono i principali metodi identificati nei SSD di UC9:

CONTRATTO 2 UC9: Pianifica Sessioni Mirate

- **Operazione:** suggerisciSessioneMirata(GiocatoreSelezionato)
- **Riferimento:** UC9-Pianifica Sessioni Mirate
- **Pre-condizione:**
  1. L'allenatore ha effettuato correttamente l'accesso al sistema.
  2. Dal menu principale l'allenatore ha selezionato l'opzione Sessione mirata.
  3. L'allenatore visualizza correttamente la lista dei vari Giocatori presenti in Rosa.
  4. Al giocatore selezionato è associata almeno un'istanza di statisticaAllenamento.
  5. Esiste un algoritmo per l'elaborazione dei parametri per identificare le varie lacune
- **Post-condizione:**
  1. I dati del giocatore selezionati sono stati analizzati per identificare eventuali lacune, secondo una logica specifica per parametro.
  2. I suggerimenti vengono mostrati a video

UC10 – Confronta giocatore

I seguenti Diagrammi di Sequenza di Sistema illustrano le interazioni I/O dell'utente con il sistema TeamManager relative al Caso d'Uso 10:



Contratti dell'operazioni UC10

I contratti delle operazioni descrivono i principali metodi identificati nei SSD di UC10:

CONTRATTO 2 UC10: Confronta Giocatori

- **Operazione:** confrontaGiocatori(g1, g2)
- **Riferimento:** UC10-ConfrontaGiocatori
- **Pre-condizione:**
  1. L'allenatore ha effettuato correttamente l'accesso al sistema.
  2. Dal menu principale l'allenatore ha selezionato l'opzione Confronta Giocatori.
  3. L'allenatore visualizza correttamente la lista dei vari Giocatori presenti in Rosa.
  4. Ai giocatori selezionati è associata almeno un'istanza di statisticaAmichevole.
- **Post-condizione:**
  1. Il sistema effettua un confronto basato sulle statisticheAmichevole dei due giocatori.
  2. L'allenatore visualizza il confronto fra i due giocatori e ne trae le informazioni desiderate.

3. Il confronto viene visualizzato in maniera grafica.

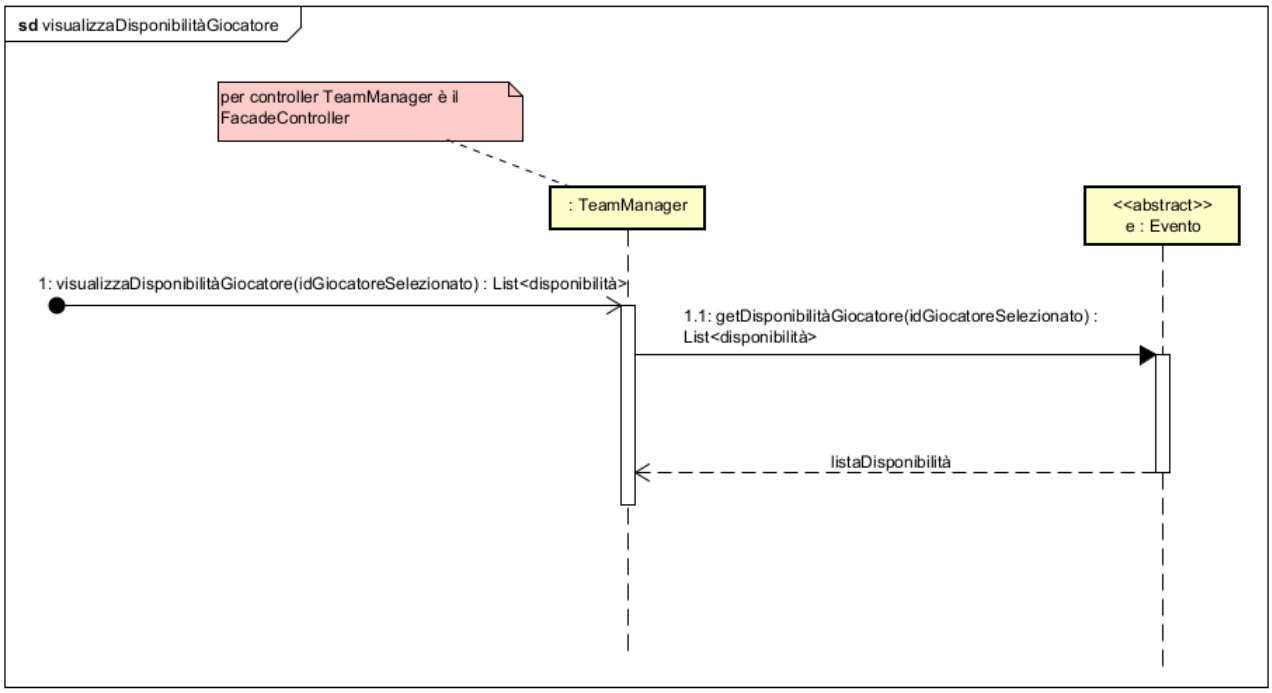
3.0 Fase di Progettazione

Gli elaborati principali della fase di progettazione comprendono i diagrammi di sequenza, utilizzati per descrivere il comportamento dinamico del sistema durante l'esecuzione dei casi d'uso selezionati per la seconda iterazione. A supporto di questi, il diagramma delle classi offre una rappresentazione statica del sistema, evidenziando le principali entità coinvolte, i relativi attributi e metodi, nonché le associazioni tra di esse. Di seguito vengono riportati i diagrammi realizzati.

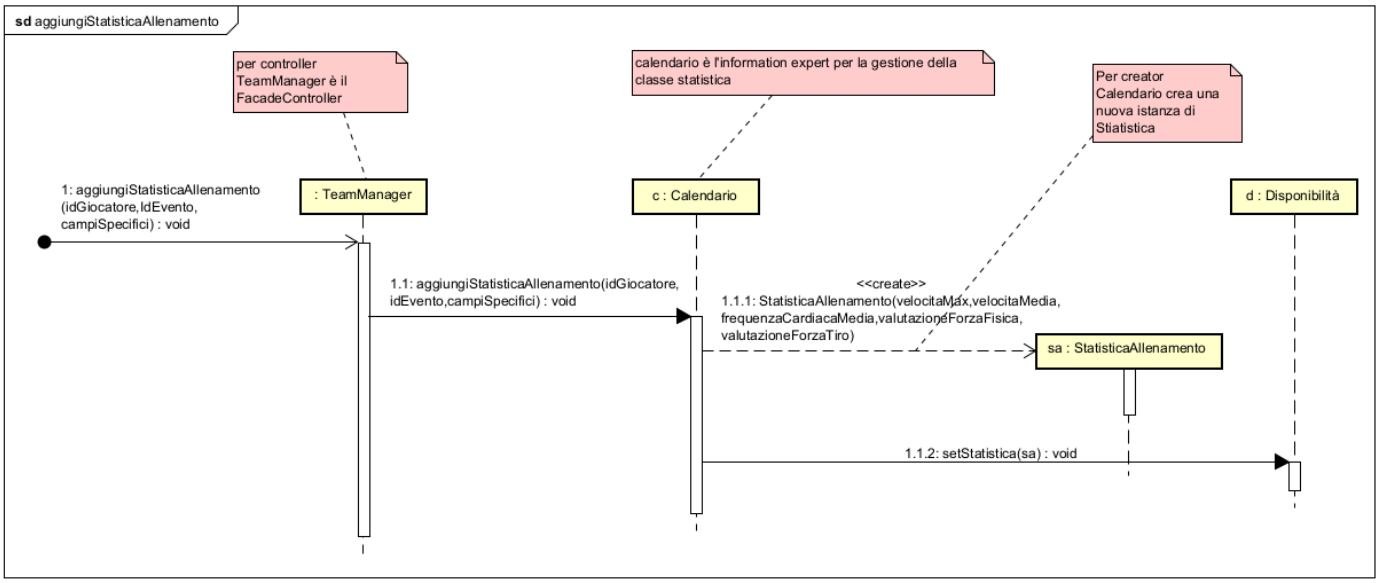
3.1 Diagrammi di Sequenza UC6

La classe TeamManager è stata progettata secondo il pattern GoF **Façade Controller**, fungendo da punto di accesso centrale alle funzionalità principali del sistema. In particolare, per il caso d'uso **Gestisci Progressi Giocatore**, TeamManager coordina l'interazione tra l'interfaccia utente e le classi del dominio, delegando a Calendario la creazione concreta delle statistiche associate ai singoli giocatori attraverso il pattern **Creator** che si differenzieranno in base alla tipologia di evento (allenamento o amichevole) in StatisticaAllenamento e StatisticaAmichevole. Calendario fungerà anche da InformationExpert e si occuperà lui stesso di eliminare istanze di Statica già esistenti. Il caso d'uso inoltre prevede anche la possibilità di visualizzare lo storico dei progressi di un giocatore per uno specifico evento.

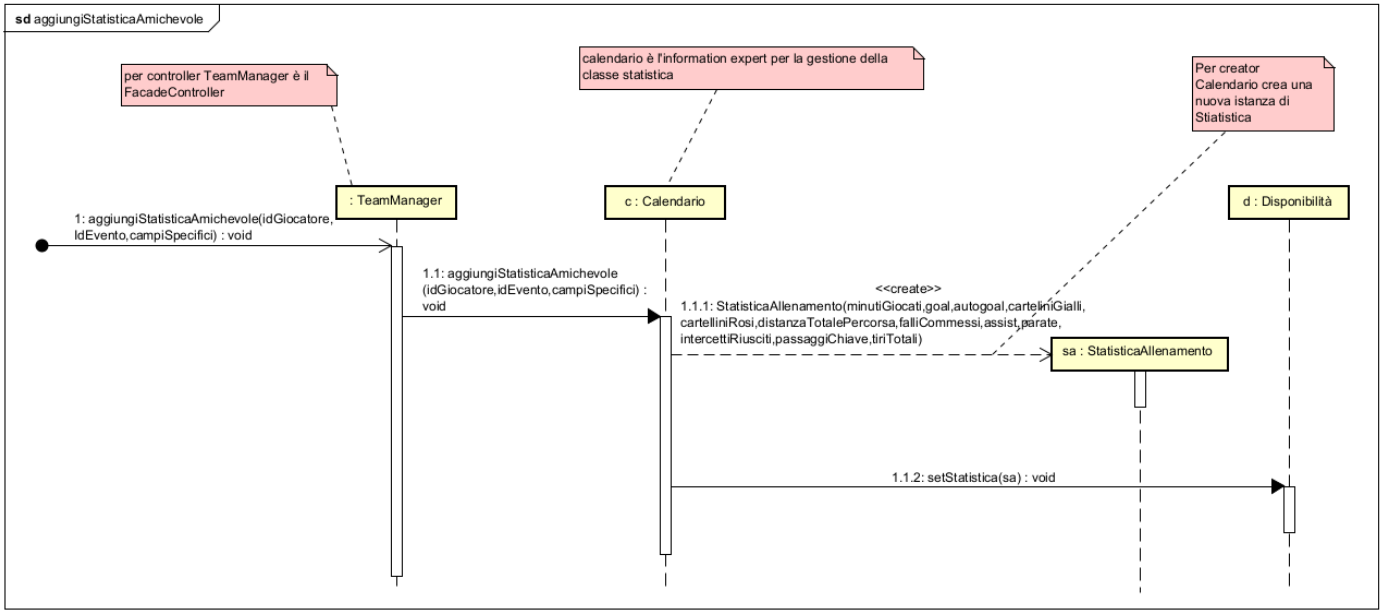
3.1.1 visualizzaDisponibilitàGiocatore(idGiocatoreSelezionato)



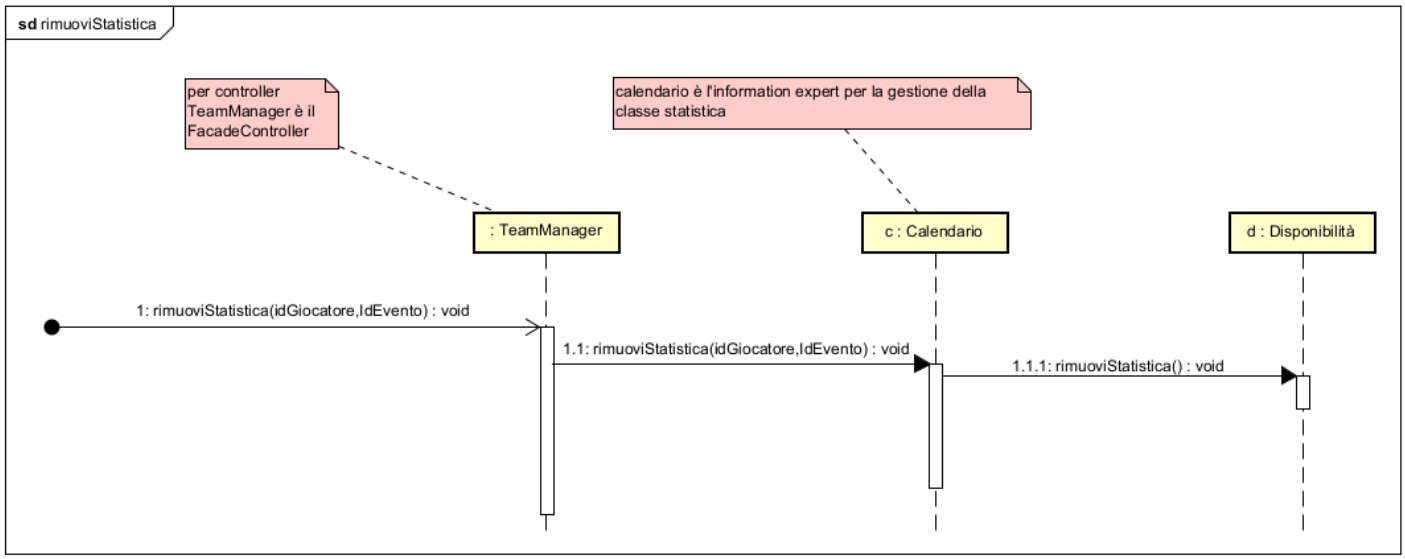
3.1.2 aggiungiStatisticaAllenamento(idGiocatore,IdEvento)



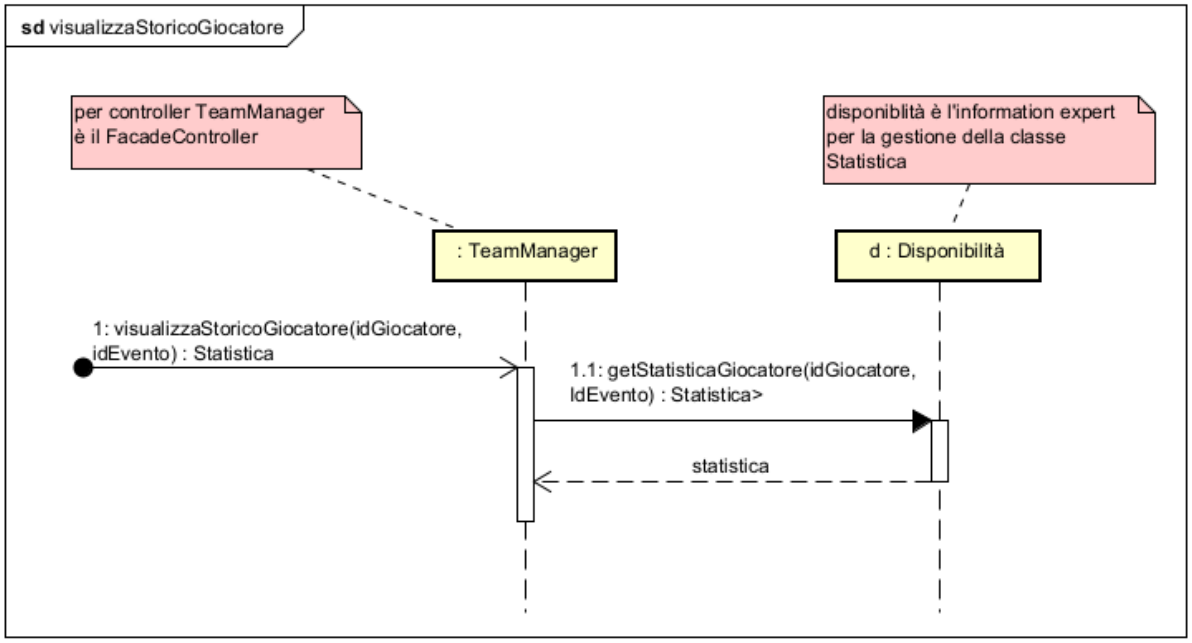
3.1.3 aggiungiStatisticaAmichevole(idGiocatore,IdEvento)



3.1.4 rimuoviStatistica(idGiocatore,IdEvento)



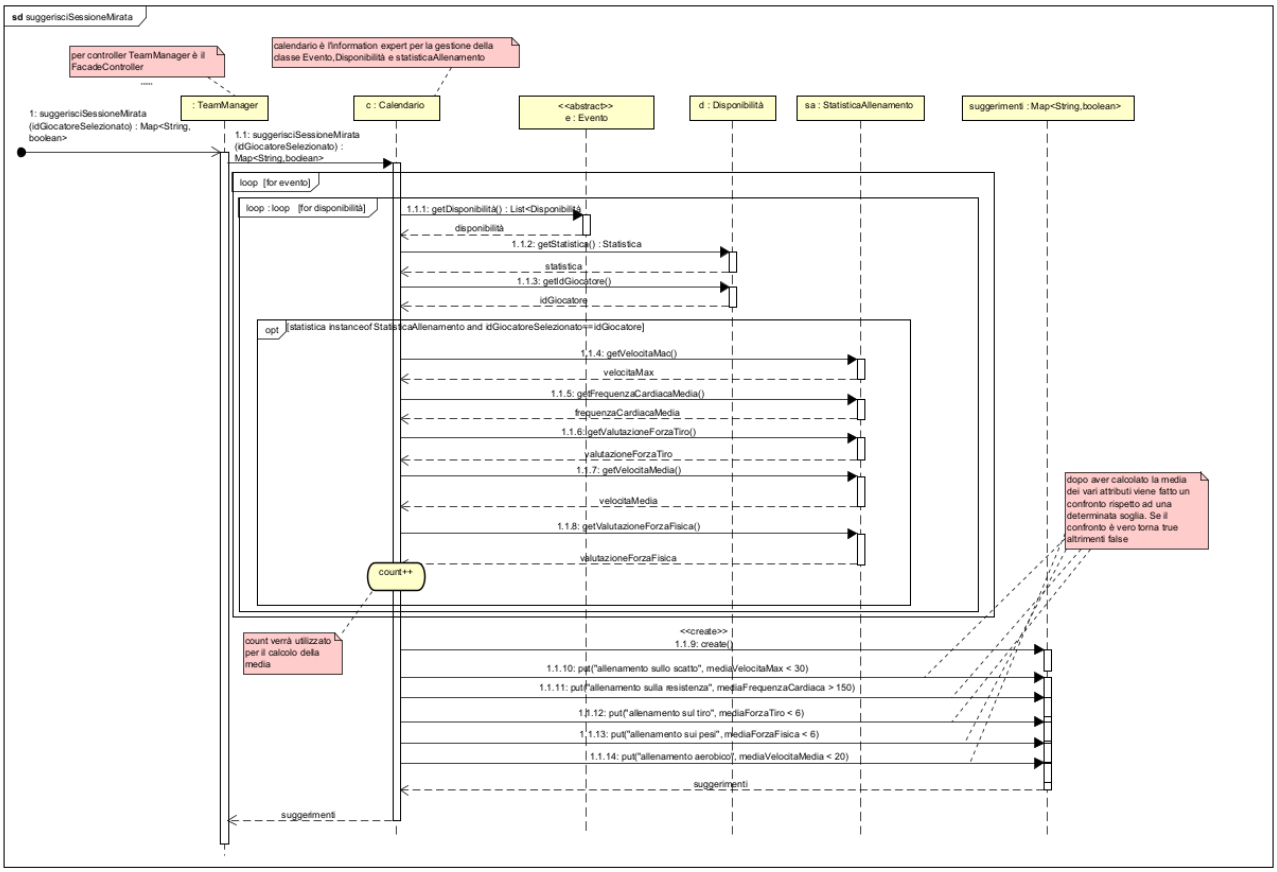
3.1.5 visualizzaStoricoGiocatore(idGiocatore,IdEvento)



3.2 Diagrammi di Sequenza UC9

La classe TeamManager è stata progettata secondo il pattern GoF **Façade Controller**, fungendo da punto di accesso centrale alle funzionalità principali del sistema. Nel caso d'uso **UC9 – Suggerisci Sessioni Mirate**, TeamManager coordina l'interazione tra l'interfaccia utente e le classi del dominio, orchestrando il flusso delle operazioni necessarie all'analisi delle performance dei giocatori. In particolare, TeamManager si occupa di recuperare la lista di GiocatoriInRosa filtrati per ruolo, interfacciandosi con la classe Rosa. Successivamente, il sistema ottiene, per ciascun giocatore, le StatisticheAllenamento registrate attraverso la classe Calendario che assume il ruolo di **Information Expert** per la loro gestione. Sulla base dei dati raccolti, TeamManager elabora le performance e identifica eventuali carenze rispetto a soglie predefinite. In presenza di lacune, il sistema procede con la creazione di una Sessione Mirata, ovvero un piano di allenamento personalizzato e progettato per colmare le specifiche debolezze emerse.

3.2.1 suggerisciSessioneMirata (idGiocatoreSelezionato)



3.3 Diagrammi di Sequenza UC10

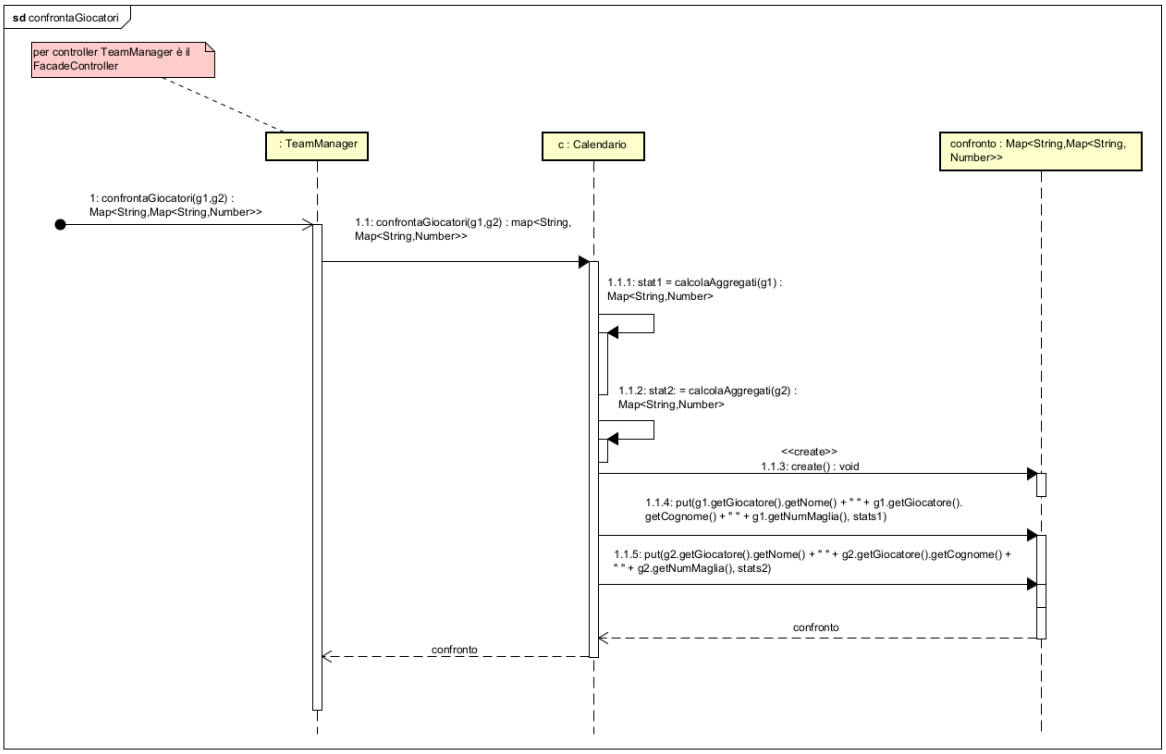
La classe TeamManager è stata progettata secondo il pattern GoF Façade Controller, fungendo da punto di accesso unificato alle funzionalità principali del sistema. Nel caso d'uso UC10 – Confronta Giocatori, TeamManager coordina l'interazione tra l'interfaccia utente e le classi del dominio per consentire all'allenatore di eseguire un'analisi comparativa tra due giocatori attivi nella rosa.

Il processo inizia con la selezione di due istanze di **GiocatoreInRosa**, che possono essere recuperate interrogando la classe **Rosa** attraverso i metodi di ricerca già disponibili. Successivamente, **TeamManager** si avvale della classe **Calendario**, che agisce come **Information Expert**, per raccogliere tutte le disponibilità associate a ciascun giocatore e, attraverso queste, ottenere le relative statistiche di tipo **StatisticaAmichevole**.

Le statistiche raccolte vengono aggregate (calcolando medie o somme in base alla natura dell'attributo) e organizzate in una mappa che restituisce, per ciascun giocatore, i valori confrontabili. La logica del confronto vero e proprio (evidenziare chi ha il valore migliore parametro per parametro) è demandata all'interfaccia utente, che interpreta la mappa e applica stili grafici per mostrare punti di forza o debolezza.

Questa funzionalità rappresenta uno strumento strategico di supporto alle decisioni tecniche, utile per analisi individuali, confronti di rendimento e scelte di formazione più consapevoli.

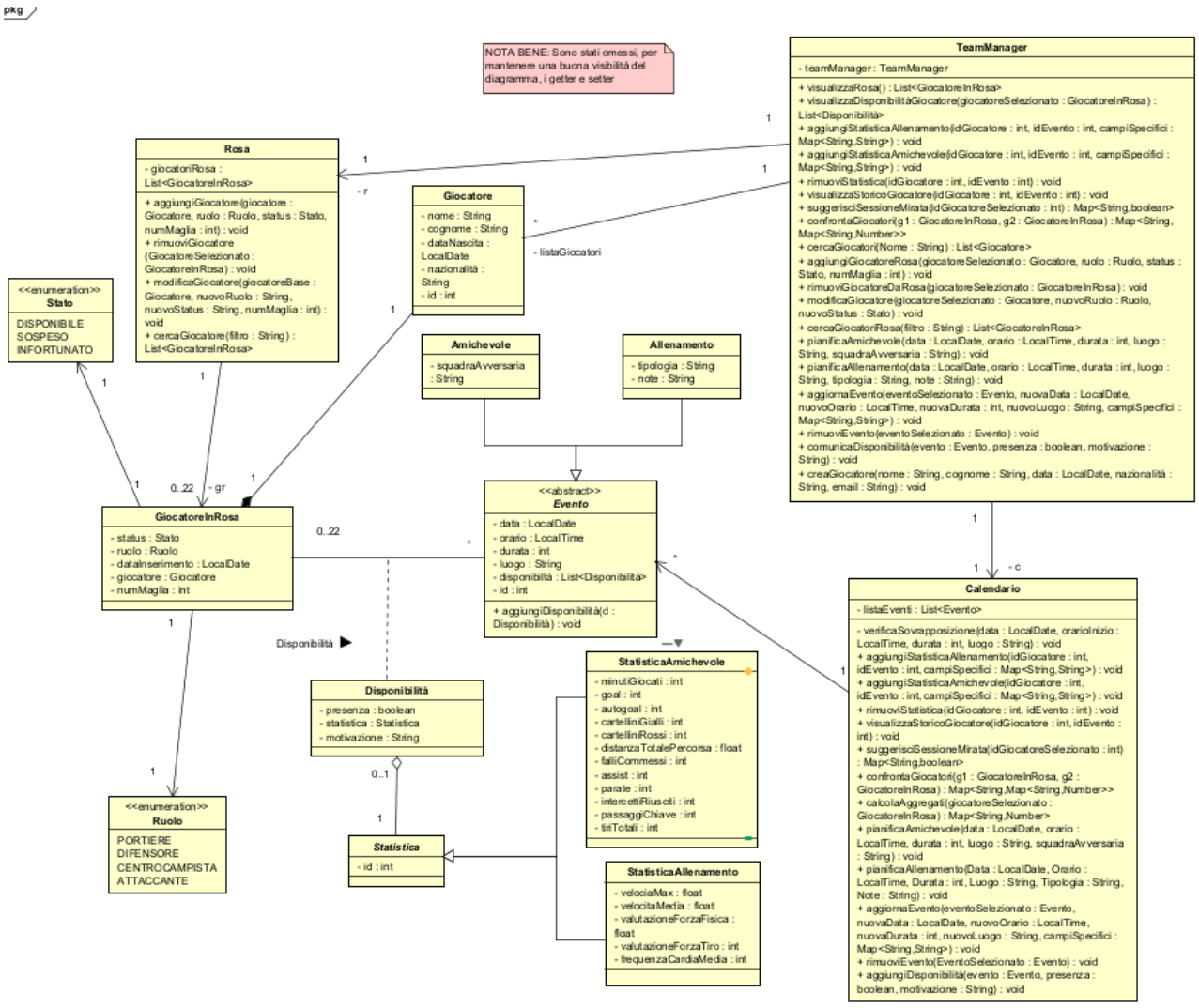
3.3.1 confrontaGiocatori(g1, g2)



4.0 Diagramma delle Classi

Il diagramma delle classi rappresenta la struttura statica del sistema TeamManager, evidenziando le principali entità coinvolte nella seconda iterazione e le relazioni tra esse. La classe TeamManager, progettata come Singleton secondo i pattern GoF, funge da Facade Controller, centralizzando il coordinamento delle operazioni tra l'interfaccia

utente e il modello di dominio. Per semplicità di rappresentazione, nei diagrammi UML non sono riportati esplicitamente i metodi getter e setter, i quali saranno comunque implementati nel codice per garantire l'accesso controllato agli attributi delle classi. Rispetto alla prima iterazione, sono state abolite le relazioni dirette di Calendario con le classi Amichevole e Allenamento, associando a Calendario solamente una lista di eventi di classe Evento , la quale possiede due generalizzazioni, ovvero, Amichevole ed Allenamento.



5.0 Testing

Durante la seconda iterazione, alcuni metodi di test sviluppati nella prima fase sono stati rivisti e aggiornati per allinearsi alle modifiche introdotte nel sistema. Inoltre, sono stati implementati e verificati nuovi metodi di test per coprire le funzionalità aggiuntive previste dai casi d'uso di questa iterazione. Di seguito si riportano i nuovi test definiti:

5.1 Calendario

- **testConfrontaGiocatoriCompleto():**

Questo test verifica in modo completo la funzionalità di confronto tra due giocatori in rosa. Simula la creazione di due eventi di tipo amichevole con relative disponibilità e statistiche, esegue il confronto aggregando correttamente i parametri e controlla che i risultati calcolati (somme o medie) corrispondano alle attese. Inoltre, verifica la struttura della mappa restituita, la correttezza dei tipi di valore e la gestione di più statistiche per giocatore.

- **testAggiungiStatisticaAllenamento():**
  Il test controlla la corretta aggiunta di una statistica di tipo allenamento a un evento già pianificato. Simula i casi di errore come l'aggiunta a un evento o disponibilità inesistente, verificando che vengano sollevate le eccezioni appropriate. Infine, verifica che la statistica venga effettivamente associata alla disponibilità quando le condizioni sono corrette.
- **testAggiungiStatisticaAmichevole():**
  Questo test verifica la funzionalità di aggiunta di una statistica per eventi di tipo amichevole. Ricrea uno scenario realistico con pianificazione di un evento, gestione delle disponibilità e test degli edge case (evento o disponibilità mancanti). Infine, accerta che la statistica venga aggiunta correttamente alla disponibilità selezionata.
- **testRimuoviStatistica():**
  Il test verifica la corretta rimozione di una statistica da una disponibilità. Viene simulato un flusso completo: creazione dell'evento, aggiunta della disponibilità e associazione di una statistica. Controlla i casi di errore (evento o disponibilità non trovati) e verifica che dopo la rimozione la statistica risulti effettivamente nulla.
- **testVisualizzaStorico():**
  Questo test garantisce che la funzionalità di visualizzazione dello storico di una statistica associata a una disponibilità funzioni correttamente. Dopo aver creato un evento, una disponibilità e aver aggiunto una statistica, controlla che la statistica venga recuperata correttamente e che, dopo la rimozione, risulti nulla come previsto.

- **testSuggerisciSessioneMirata():**

  Il test controlla il comportamento del metodo che suggerisce sessioni di allenamento mirate per un giocatore. Simula due eventi di allenamento con statistiche di prova, calcola le medie dei parametri significativi e verifica che la logica di confronto restituisca i suggerimenti corretti. Gestisce anche il caso in cui non siano presenti dati per il giocatore, controllando che venga sollevata un'eccezione.