Shotokan A.S.D

Introduzione

Il progetto schematizza ed enfatizza quelle che sono le fasi tipiche del processo di creazione di un sistema software.
Nello specifico sono state usate alcune delle più importanti tecniche viste durante il corso.
La metodologia dell'Unified Process (UP), vista per la realizzazione, è sicuramente complessa ma fornisce ottime linee guida utili per usare al meglio i diagrammi UML.

FASE DI IDEAZIONE

1. Documento di visione

Descrizione aziendale

La Shotokan A.S.D S.r.l, è una società operante nel settore fitness e benessere dal 1992, che offre servizi (sauna, massaggio, riabilitazione ecc…) e mette a disposizione strutture per poter svolgere una molteplicità di attività sportive (Palestre, Campi da calcio, Campo da tennis, Campo Basket, Piscina, Sala Danza/Karate).
La società prevede lo svolgimento di diverse attività sportive quali Calcio, Tennis, Basket, BodyBuilding, Nuoto, Danza, Karate, Boxe.
Vengono inoltre offerti servizi aggiuntivi da poter integrare al semplice piano di allenamento o alle attività svolte (Doccia, Sauna, Massaggi, Bevande energetiche, Shaker proteico, Personal Trainer).

Obiettivi e problemi delle parti interessate

Vista la totale assenza di software informatici capaci di gestire le iscrizioni, le prenotazioni dei vari campi e l'aggiunta dei vari servizi supplementari l'azienda richiede lo sviluppo di un software capace di informatizzare tutti i processi.
Il software richiesto è un sistema informatico di gestione che deve afferire alle seguenti operazioni:

1) Deve permettere l'aggiunta di nuovi soci
2) Deve permettere l'aggiunta di nuove attività sportive
3) Deve permettere l'aggiunta di nuovi servizi
4) Deve permettere la prenotazione degli impianti sportivi
5) Deve permettere il rinnovo della mensilità ai soci
6) Deve permettere di verificare disponibilità di un impianto sportivo nel giorno scelto
7) Ad ogni iscrizione o rinnovo elabora una nuova tessera con i dati utente.

Obiettivo	Priorità	Problemi	Soluzione
Gestione operazioni, flessibilità e velocità	alta	Il salvataggio dei dati su database potrebbe determinare rallentamenti dovuto al carico eccessivo dei dati.	Si potrebbe prevedere l'uso di soluzioni cloud atte a ridurre il problema.
Facilità d'utilizzo	alta	Il software potrebbe prevedere conoscenze aggiuntive su codici relative alle singole attività	Si potrebbe corredare il software di relativa documentazione al fine di semplificare il processo.

2. Glossario

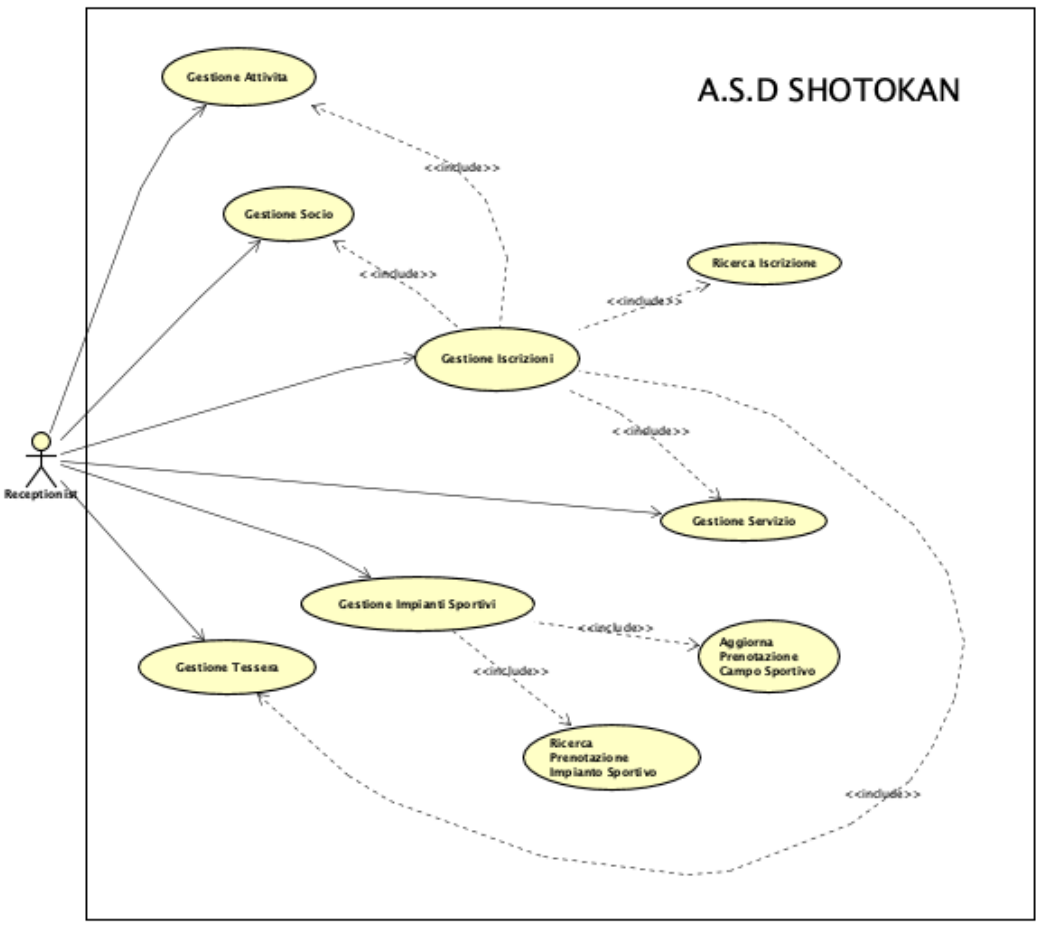
Centro fitness

Receptionist: E' un utilizzatore del sistema. Si occupa di svolgere tutte le operazioni per mezzo del software che gli viene messo a disposizione.

Operazioni svolte:

UC1 : Gestione Iscrizione (CRUD + Ricerca Iscrizione per Utente)
UC2 : Gestione Impianti Sportivi (CRUD + Aggiunta Prenotazione)
UC3 : Gestione Attività (CRUD + Ricerca attività)
UC4 : Gestione Servizio (CRUD + Ricerca servizio)
UC5 : Aggiorna Prenotazione Impianto Sportivo
UC6 : Ricerca Prenotazione Impianto Sportivo

Il diagramma dei casi d'uso fino ad ora descritti risulta essere il seguente.



Clienti o soci: soci che aderiscono alle attività della palestra.

Caratterizzazione:

Identificativo Utente
Nome
Cognome
Email
Anno
Numero Tessera

Operazioni svolte (non su applicazione):

Scelta attività
Scelta servizi aggiuntivi
Effettuano il pagamento relativo alle attività più i servizi scelti
Visualizzano il catalogo
Visualizzano regole d'uso degli impianti sportivi si autenticano al servizio

Attività sportive: sono gli elementi o prodotti (che possono essere intesi come oggetti) forniti dalla Shotokan A.S.D, che l'utente può scegliere e definiscono l'attività sportiva che questo andrà a svolgere.

Caratterizzazione:

Codice identificativo
Nome
Tariffa

Servizi aggiuntivi : anche in questo caso identificano degli elementi che l'utente può decidere o meno di inserire nel suo piano mensile. Costituiscono un plus sul costo delle attività scelte (la quantità massima scelta può essere 30).

Caratterizzazione:

Codice identificativo
Nome
Tariffa

Impianto sportivo: costituisce un elemento (oggetto) che l'utente potrebbe prenotare e ha una valenza giornaliera. Ogni impianto sportivo viene prenotato in modo giornaliero per garantire la corretta gestione post attività (igienizzazione campo e spogliatoi)

Caratterizzazione:

Codice identificativo
Nome
Tariffa

3. Analisi funzionale e modello dei casi d'uso

Tra i casi d'uso individuati si è scelto di dettagliare i casi d'uso relativi alla Gestione Iscrizione, Gestione Prenotazione Impianti Sportivi e Gestione delle Attività e dei Servizi. I restanti casi d'uso verranno illustrati in formato breve o informale.

Caso d'uso UC1 : Gestione Iscrizione

Nome caso d'uso	UC1 : Gestione Iscrizione
Portata	Applicazione A.S.D.Shotokan
Livello	Obiettivo utente
Attore Primario	Receptionist
Parti interessate e Interessi	Receptionist Vuole gestire l'intero processo di iscrizione dell'utente in modo semplice, corretto e veloce. NO-CRUD Inoltre si vuole che anche le informazioni relative alle attività e servizi degli utenti siano aggiornate. Socio: Vuole concludere il processo di iscrizione nella maniera più veloce possibile.
Pre-condizioni	Il cliente è a conoscenza del piano attività da scegliere fin dal momento in cui si reca in palestra per la prima volta.
Garanzia di successo	Il processo risulta completo quando il nuovo socio sarà correttamente iscritto in palestra.
Scenario principale di successo	1) Il receptionist inizia una nuova operazione 2) Sceglie attività "Iscrizione utente" 3) Il receptionist aggiunge nome, cognome, anno di nascita, genere e data iscrizione Il cliente sceglie attività da seguire più eventuali servizi offerti dalla struttura 4) Il receptionist inserisce il codice identificativo e durata (in giorni) dell'attività scelta Il sistema mostra i dati relativi all'attività scelta (titolo, genere e costo) Il passo 3 è ripetuto fin quando non vengono inserite tutte le attività 5) Il receptionist conferma le attività inserite 6) Il receptionist aggiunge eventuali servizi indicando codice identificativo e quantità Il sistema registra l'aggiunta dei dati ed elabora il totale. Il cliente effettua il pagamento 7) Il receptionist conferma l'iscrizione all'attività
Estensioni	Se vengono inseriti formati errati il programma smette di funzionare e l'utente verrà notificato dell'errore supposto.

	Inoltre il Receptionist avrà la possibilità di vedere la lista delle iscrizioni con i dati degli utenti e la singola iscrizione inserendo nome e cognome del singolo utente.
Frequenza di ripetizioni	Il tutto dipende dal nuovo socio e dalle proposte in termini di attività e servizi che intende sovrascrivere.
Varie	

Caso d'uso UC2 : Gestione Impianti Sportivi

Nome caso d'uso	UC2: Gestione Impianti Sportivi
Portata	Applicazione A.S.D.Shotokan
Livello	Obiettivo utente
Attore Primario	Receptionist
Parti interessate e Interessi	Receptionist Vuole gestire un impianto sportivo (può essere un campo o una sala con ring per l'attività di boxe). Nello specifico vuole effettuare le classiche operazioni CRUD di Aggiunta, Aggiornamento (per poter cambiare il prezzo ad esempio), Eliminazione (in quanto l'impianto sportivo non è più agibile). NO-CRUD Vorrebbe anche avere la possibilità di ricercare tutti gli impianti sportivi e le informazioni del singolo impianto sportivo.
Pre-condizioni	
Garanzia di successo	Il processo risulta completo quando il receptionist avrà potuto completare il processo di inserimento/aggiornamento/eliminazione di uno specifico impianto sportivo o avrà potuto controllare tutte le prenotazioni relative ad un impianto sportivo.
Scenario principale di successo	1) Il receptionist vuole inserire un nuovo campo sportivo. *N.B* *Alternativamente potrebbe aggiornare le informazioni o eliminarlo* 2) Il receptionist sceglie l'attività "Aggiungi Impianto Sportivo" o qualunque altra. 3) Il receptionist inserisce (codice identificativo, descrizione, tariffa) 4) Il receptionist conferma l'inserimento/aggiornamento/rimozione dell'impianto sportivo. Il sistema registra l'operazione eseguita.
Estensioni	Se vengono inseriti formati errati il programma smette di funzionare e il Receptionist verrà notificato dell'errore supposto.
Frequenza di ripetizioni	Il tutto dipende dal numero di inserimenti che il receptionist vuole eseguire.
Varie	

Caso d'uso UC3 : Gestione Attività

Nome caso d'uso	UC3: Gestione Attività
Portata	Applicazione A.S.D.Shotokan
Livello	Obiettivo utente
Attore Primario	Receptionist
Parti interessate e Interessi	Receptionist Vuole gestire le attività disponibili. Nello specifico vuole effettuare le classiche operazioni CRUD di Aggiunta, Aggiornamento (per poter cambiare il prezzo o la descrizione dell'attività), Eliminazione (nel caso in cui un'attività non fosse più disponibile). <u>NO-CRUD</u> Vorrebbe anche avere la possibilità di ricercare tutte le attività svolte da un utente specifico (al solito inserendo nome e cognome del suddetto utente).
Pre-condizioni	
Garanzia di successo	Il processo risulta completo quando il receptionist avrà potuto completare il processo di inserimento/aggiornamento/eliminazione di una specifica attività o avrà potuto controllare le attività svolte da uno specifico cliente.
Scenario principale di successo	1) Il receptionist vuole inserire una nuova attività *N.B* *Alternativamente potrebbe aggiornarla/eliminarla* 2) Il receptionist sceglie l'attività "Aggiungi attività" o qualunque altra. 3) Il receptionist inserisce (codice identificativo, descrizione, tariffa) 4) Il receptionist conferma l'inserimento/aggiornamento/rimozione dell'attività Il sistema registra l'operazione eseguita.
Estensioni	Se vengono inseriti formati errati il programma smette di funzionare e il Receptionist verrà notificato dell'errore supposto.
Frequenza di ripetizioni	Il tutto dipende dal numero di attività che il Receptionist vuole gestire.
Varie	

Caso d'uso UC4 : Gestione Servizio

Nome caso d'uso	UC4: Gestione Servizio
Portata	Applicazione A.S.D.Shotokan
Livello	Obiettivo utente
Attore Primario	Receptionist

Parti interessate e Interessi	Receptionist Vuole gestire i servizi disponibili. Nello specifico vuole effettuare le classiche operazioni CRUD di Aggiunta, Aggiornamento (per poter cambiare il prezzo o la descrizione del servizio), Eliminazione (nel caso in cui un servizio non fosse più disponibile). Vorrebbe anche avere la possibilità di ricercare tutti i servizi attivi di un utente specifico (al solito inserendo nome e cognome del suddetto utente).
Pre-condizioni	
Garanzia di successo	Il processo risulta completo quando il receptionist avrà potuto completare il processo di inserimento/aggiornamento/eliminazione di una specifica attività o avrà potuto controllare le attività svolte da uno specifico cliente.
Scenario principale di successo	1) Il receptionist vuole inserire una nuova attività *N.B* *Alternativamente potrebbe aggiornarla/eliminarla* 2) Il receptionist sceglie l'attività "Aggiungi attività" o qualunque altra. 3) Il receptionist inserisce (codice identificativo, descrizione, tariffa) 4) Il receptionist conferma l'inserimento/aggiornamento/rimozione dell'attività Il sistema registra l'operazione eseguita.
Estensioni	Se vengono inseriti formati errati il programma smette di funzionare e il Receptionist verrà notificato dell'errore supposto.
Frequenza di ripetizioni	Il tutto dipende dal numero di attività che il Receptionist vuole gestire.
Varie	

Caso d'uso UC5: Gestione Prenotazione Impianto Sportivo

Attore Primario	***Attore di supporto***	***Attore fuori scena***
Receptionist	*Sistema*	*Cliente*

Scenari:

1) Il receptionist vuole inserire una nuova prenotazione
2) Il receptionist sceglie l'attività "Aggiungi prenotazione" o "Modifica Prenotazione" o "Elimina prenotazione".
3) Il receptionist inserisce (nome, cognome, descrizione, giorno, tariffa).

Nel caso in cui si scegliesse di eliminare un elemento basta inserire solo la descrizione e il giorno.

4) Il receptionist conferma l'operazione.

Il sistema registra / aggiorna / elimina la prenotazione.

Caso d'uso UC6: Ricerca Prenotazione Impianto Sportivo

Attore Primario	*Attore di supporto*	*Attore fuori scena*
Receptionist	*Sistema*	*Cliente*

Scenari:

1) *Il receptionist vuole effettuare la ricerca relativa alla prenotazione di un impianto sportivo.*
   N.B. Alternativamente potrebbe anche voler mostrare la lista di tutte le prenotazioni di quello specifico impianto sportivo (in questo caso gli verra chiesto solo il nome dell'impianto sportivo).
2) *Sceglie attività "Ricerca Prenotazione Impianto Sportivo".*
3) *Inserisce il nome dell'impianto sportivo e il giorno.*
4) *Il sistema elabora le informazioni inserite e restituisce (se esistenti le informazioni della prenotazione con nome e cognome dell'utente che lo ha prenotato).*

Caso d'uso UC6: Gestione tessera

Attore Primario	*Attore di supporto*	*Attore fuori scena*
Receptionist	*Sistema*	*Cliente*

Scenari:

1) Il receptionist sceglie l'attività "Crea tessera socio"
2) Il receptionist inserisce il codice univoco tessera, nome, cognome, email, durata
3) Il receptionist conferma la creazione della nuova tessera

4. Specifiche Supplementari Richieste

Come specifiche supplementari sono presentate le seguenti richieste:

Utilizzabilità del prodotto

- L'interfaccia grafica deve essere intuitiva e semplice
- L'interazione deve essere immediata e semplice

Affidabilità del prodotto

- Il software deve essere stabile e deve mantenere le informazioni relative alla gestione della palestra anche in mancanza di elettricità o qualsiasi altro problema che ne dovesse intralciare il corretto funzionamento.

Vincoli HW e SW

- Vincoli hardware:

  Il sistema deve essere prodotto per essere correttamente eseguito su dispositivo aziendale esistente con le seguenti caratteristiche:

  - Macbook Pro 13" CD-ROM A1378 (anno 2012)
    Con Intel Core i5 dual-core 2,4 GHz
    4GB di memoria DDR3 a 1333MHz

SSD da 256GB

Su personal computer aziendale è installato il seguente sistema operativo:

- Mac OS X 10.15
- Virtual Machine con Windows 10 Entreprise

- Vincoli software:

  Tutto il software è scritto usando Java e verrà sfruttato un Database (PostgresSQL) per gestire la persistenza dei dati.

Aspetti legali

- Le tecnologie utilizzate per la progettazione e realizzazione del sistema proposto sono di tipo OpenSource e Freeware
- A.S.D Shotokan verrà rilasciato con licenza open source GPL v3.

Fase di Elaborazione – 1° Iterazione

2. Introduzione

Dopo aver terminato la fase di ideazione si procede alla fase di elaborazione in cui si ha lo scopo di affinare le idee precedentemente viste analizzando e implementando in modo iterativo lo sviluppo software. Scopo dell'elaborazione sarà inoltre correggere alcuni errori non considerati durante la fase di ideazione.
Durante la prima iterazione ci si focalizzerà sui casi d'uso UC1 relativo al modulo di Gestione dell'iscrizione.

2.2 Analisi orientata agli oggetti

Un primo approccio che permette di verificare e fornire i dettagli sul dominio è la Modellazione di Business che comprende il completamento di un **Modello di Dominio** che contenga nello specifico i Grafici, i Concetti, gli Attributi e le associazioni più significative per il caso d'uso scelto.
Inoltre si avranno ulteriori elementi significativi per definire i tratti salienti del caso d'uso, ovvero il **Diagramma di sequenza** e i **Contratti delle operazioni**.
Nel nostro caso (relativo all'UC1), per poter garantire lo scenario di successo sarà necessario focalizzarsi sulle seguenti classi concettuali:

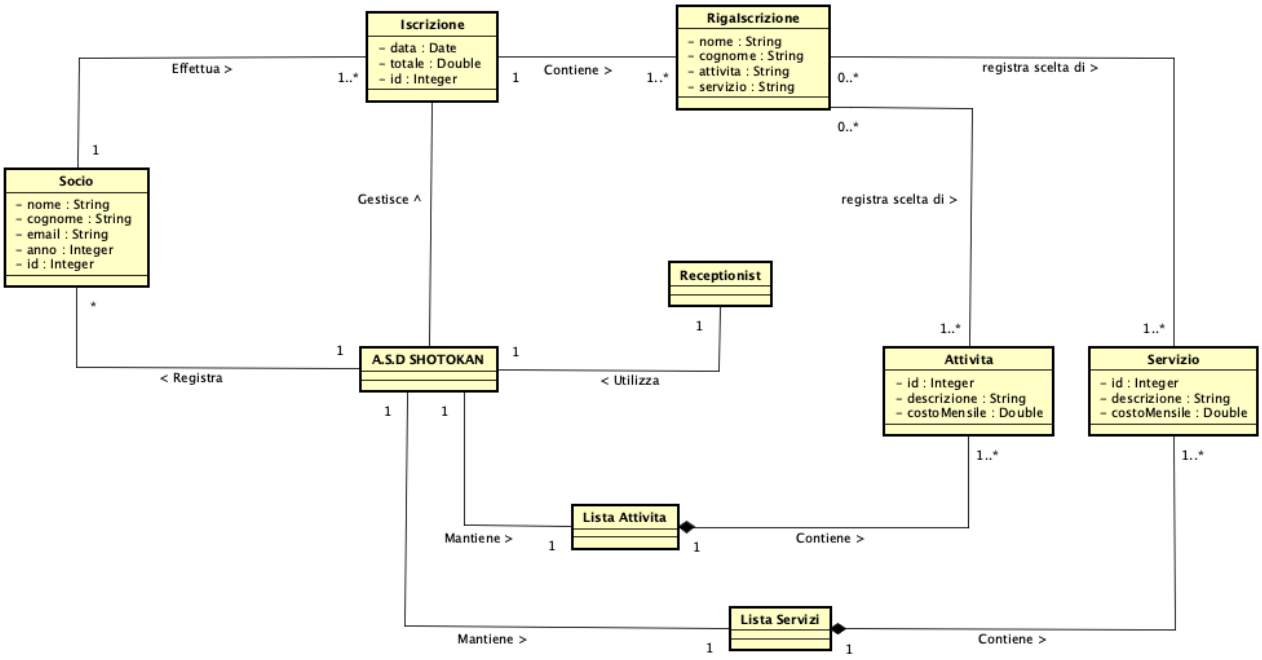
- **Iscrizione**
  Indica l'iscrizione effettuata da un cliente
- **Socio**
  Cliente della palestra
- **Receptionist**
  Attore primario, che interagisce col sistema
- **Attività**
  Rappresenta la prima scelta effettuata dall'utente
- **Servizio**
  Rappresenta la seconda scelta effettuata dall'utente
- **A.S.D.SHOTOKAN**
  Il sistema utilizzato per effettuare le operazioni
- **Lista delle attività**
  Catalogo che mostra tutte le attività disponibili
- **Lista dei servizi**
  Catalogo che mostra tutti i servizi disponibili

2.2.1 Applicazione Pattern (Prototype, Singleton e Command)

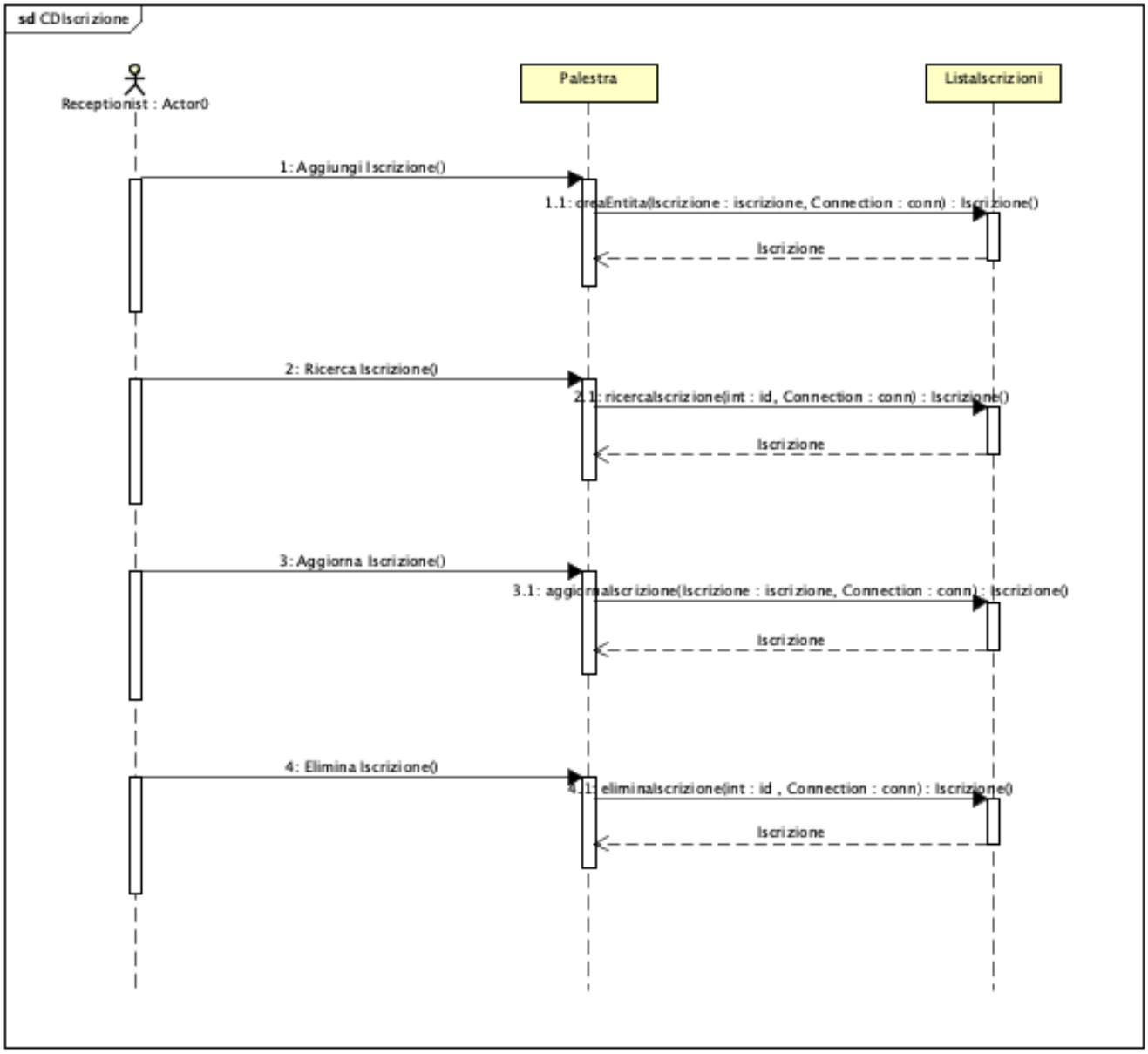
L'applicazione e la descrizione dei pattern utilizzati si trova nella sezione dedicata di questa documentazione (Documentazione Generale > Pattern e Test.pdf).

2.2.2 Analisi orientata ad oggetti

Tenuto conto degli elementi sopra avremo il seguente **Modello di Dominio**:



Come detto in precedenza l'analisi orientata agli oggetti è definita nella sua completezza dalla presenza di 3 elementi. Il secondo da introdurre è il **Diagramma di Sequenza (SSD)**, che nel nostro caso sarà il seguente:



Il terzo elemento fondamentale per la definizione completa dell'analisi orientata agli oggetti è dato dai **Contratti delle operazioni**, che descrivono gli eventi di sistema elaborati nel Diagramma di Sequenza (SSD).

Contratto CO1 : creaEntita(iscrizione: Iscrizione, conn : Connection)

Operazione:

- **creaEntita(iscrizione :Iscrizione, conn : Connection) : Iscrizione**

Riferimenti:

- UC1: Gestisci Iscrizione

Pre-Condizioni:

- è in corso l'operazione di Iscrizione

Post-condizioni:

- l'attributi nome, cognome della suddetta istanza sono riempiti con il nome e cognome del socio.
- l'attributo attivita e servizio sono compilati con il campo descrizione delle relative classi

Contratto CO2 : ricercaEntita(id : int, conn: Connection)

Operazione:
- **ricercaEntita(id : int, conn: Connection) : Iscrizione**

Riferimenti:
- UC1: Gestisci Iscrizione

Pre-Condizioni:
- è in corso l'operazione di Iscrizione

Contratto CO3 : aggiornaEntita(iscrizione :Iscrizione, conn : Connection)

Operazione:
- **modificaEntita(iscrizione :Iscrizione, conn : Connection) : Iscrizione**

Riferimenti:
- UC1: Gestisci Iscrizione

Pre-Condizioni:
- è in corso l'operazione di iscrizione

Contratto CO4 : eliminaEntita(id : int , conn : Connection) : Iscrizione

Operazione:
- **eliminaEntita(id : int , conn : Connection)**

Riferimenti:
- UC1: Gestisci Iscrizione

Pre-Condizioni:
- è in corso l'operazione di iscrizione

Contratto CO5: calcolaTotale(conn : Connection)

Operazione:
- **calcolaTotale(conn : Connection) : double**

Riferimenti:
- UC1: Gestisci Iscrizione

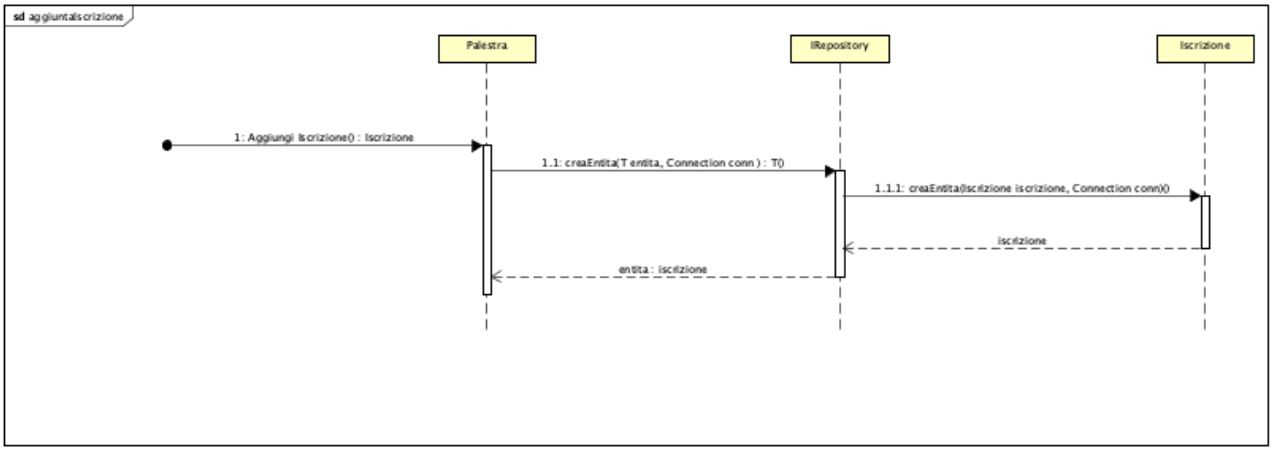
Pre-Condizioni:
- è in corso l'operazione di Iscrizione

2.3 Progettazione

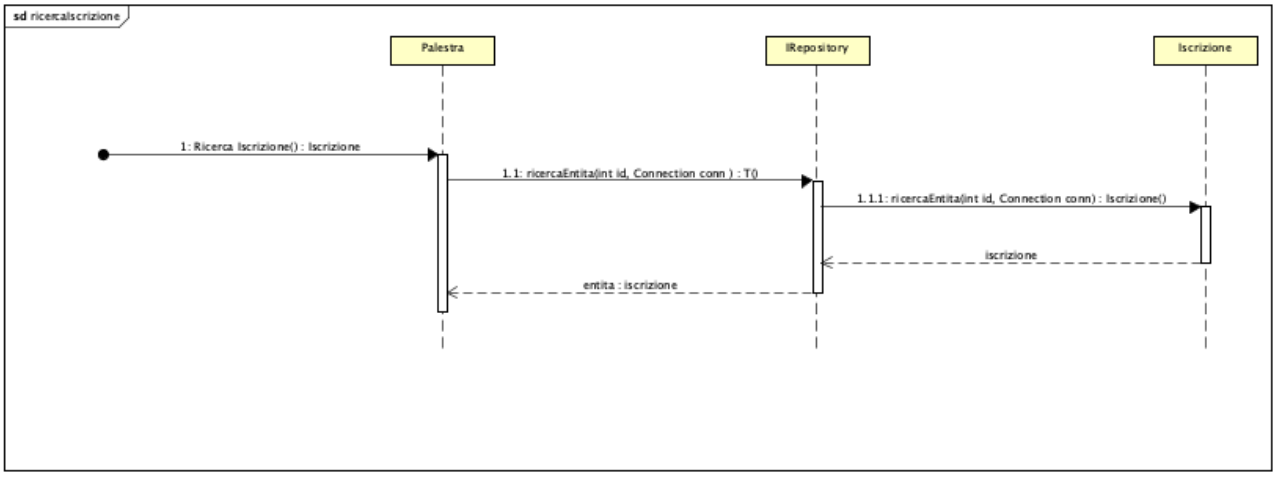
Al fine di poter definire gli oggetti software e le loro responsabilità, da un punto di vista statico o dinamico, si definisce un Modello di progetto, consistente in una serie di Diagrammi di Sequenza (SSD) e delle Classi.

2.3.1 Diagrammi di Sequenza relativi al caso d'uso UC1 : Gestione Iscrizione

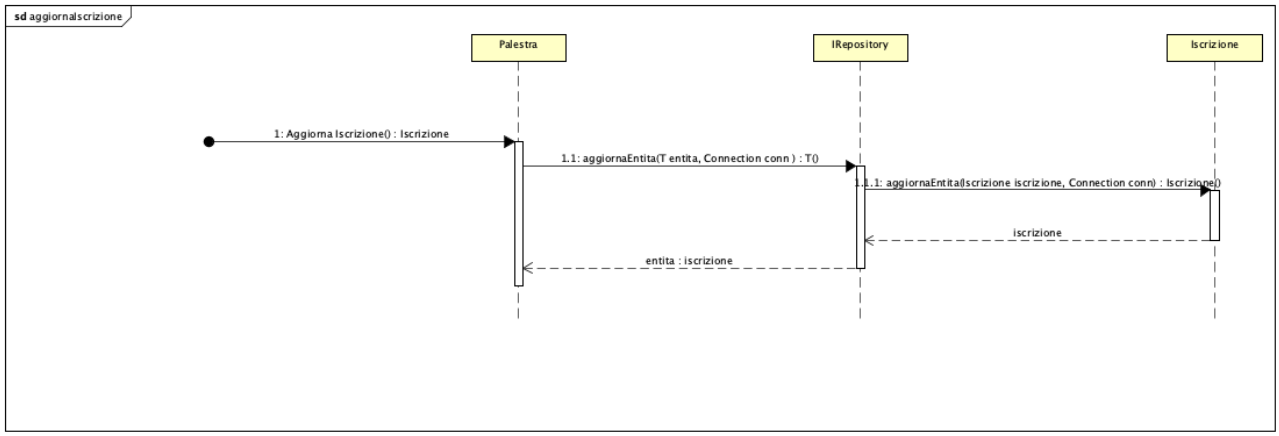
- **Aggiungi Iscrizione**



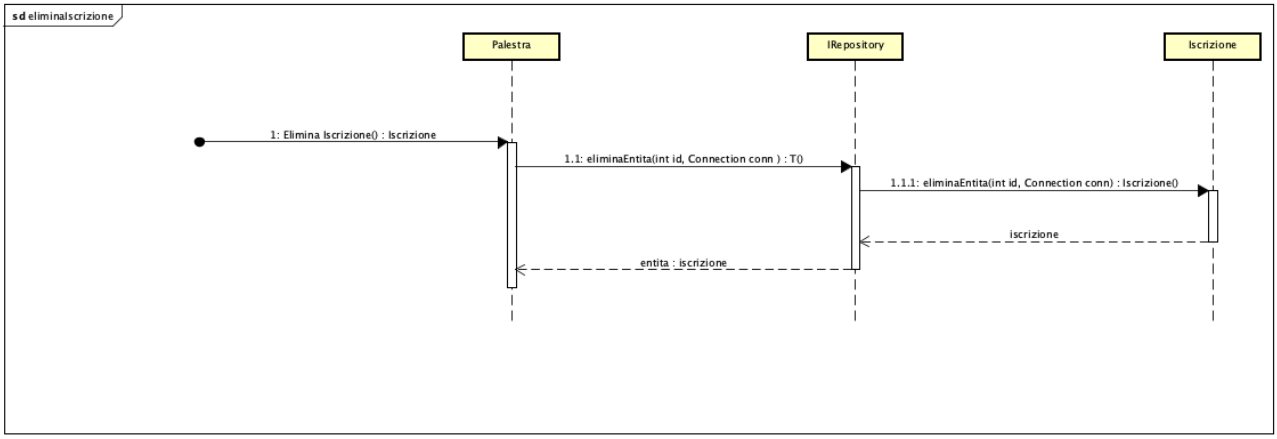
- **Ricerca Iscrizione**



- **Aggiorna Iscrizione**

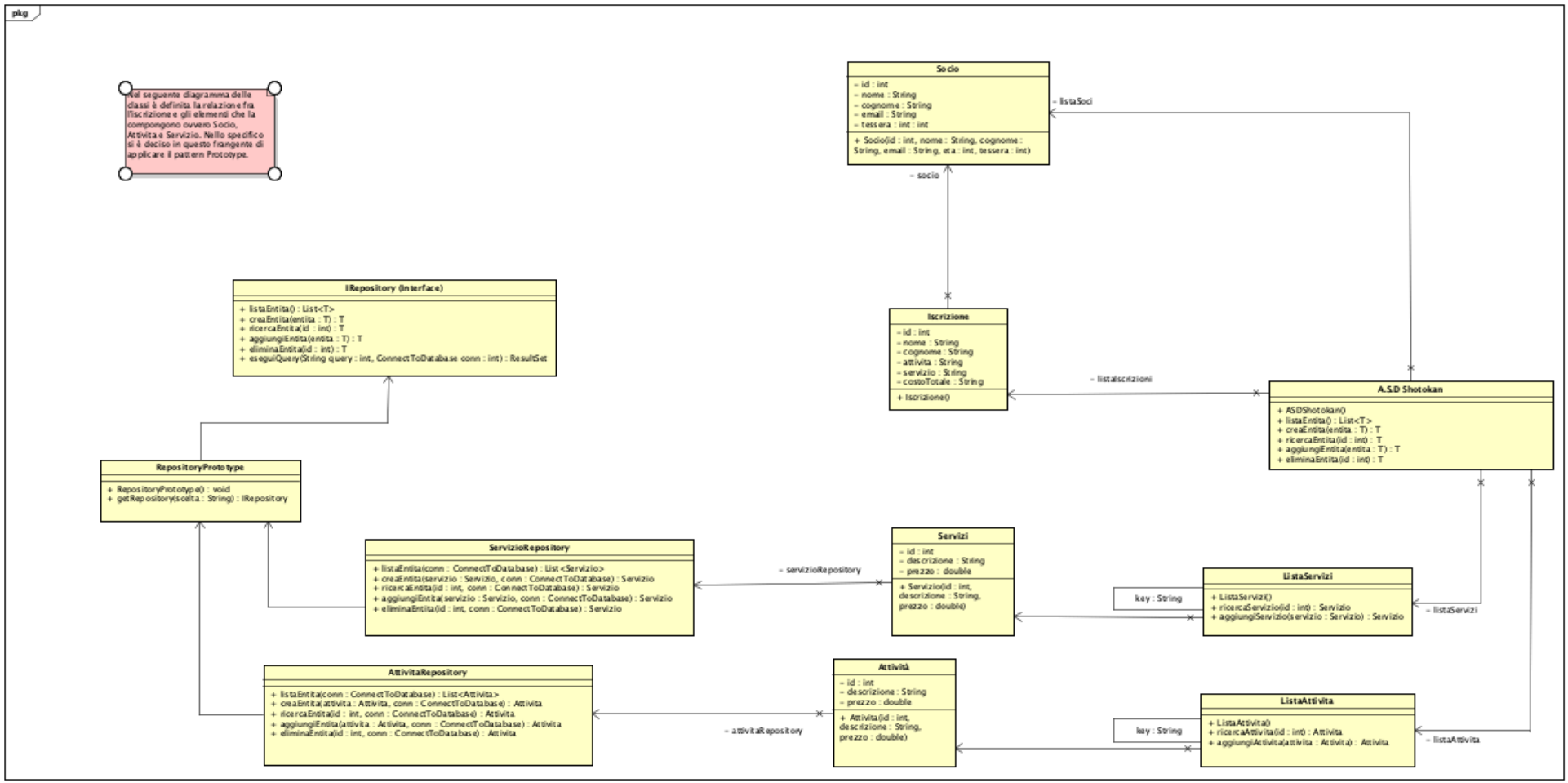


- **Elimina Iscrizione**



2.3.1 Diagramma delle classi

Al fine di rendere più chiara l'immagine si dispone la stessa in verticale e nella pagina successiva (l'immagine e il file in formato asta si trovano nella cartella Elaborazione (Iter1) > Diagramma delle classi).


pkg
Nel seguente diagramma delle classi è definita la relazione fra l'iscrizione e gli elementi che la compongono ovvero Socio, Attività e Servizio. Nello specifico si è deciso in questo frangente di applicare il pattern Prototype.
Socio
- id : int
- nome : String
- cognome : String
- email : String
- tessera : int : int
+ Socio(id : int, nome : String, cognome : String, email : String, eta : int, tessera : int)
- listaSoci
- socio
IRepository (Interface)
+ listaEntita() : List<T>
+ creaEntita(entita : T) : T
+ ricercaEntita(id : int) : T
+ aggiungiEntita(entita : T) : T
+ eliminaEntita(id : int) : T
+ eseguiQuery(String query : int, ConnectToDatabase conn : int) : ResultSet
Iscrizione
- id : int
- nome : String
- cognome : String
- attivita : String
- servizio : String
- costoTotale : String
+ Iscrizione()
- listaIscrizioni
A.S.D Shotokan
+ ASDShotokan()
+ listaEntita() : List<T>
+ creaEntita(entita : T) : T
+ ricercaEntita(id : int) : T
+ aggiungiEntita(entita : T) : T
+ eliminaEntita(id : int) : T
RepositoryPrototype
+ RepositoryPrototype() : void
+ getRepository(scelta : String) : IRepository
ServizioRepository
+ listaEntita(conn : ConnectToDatabase) : List<Servizio>
+ creaEntita(servizio : Servizio, conn : ConnectToDatabase) : Servizio
+ ricercaEntita(id : int, conn : ConnectToDatabase) : Servizio
+ aggiungiEntita(servizio : Servizio, conn : ConnectToDatabase) : Servizio
+ eliminaEntita(id : int, conn : ConnectToDatabase) : Servizio
- servizioRepository
Servizi
- id : int
- descrizione : String
- prezzo : double
+ Servizio(id : int, descrizione : String, prezzo : double)
key : String
ListaServizi
+ ListaServizi()
+ ricercaServizio(id : int) : Servizio
+ aggiungiServizio(servizio : Servizio) : Servizio
- listaServizi
AttivitaRepository
+ listaEntita(conn : ConnectToDatabase) : List<Attivita>
+ creaEntita(attivita : Attivita, conn : ConnectToDatabase) : Attivita
+ ricercaEntita(id : int, conn : ConnectToDatabase) : Attivita
+ aggiungiEntita(attivita : Attivita, conn : ConnectToDatabase) : Attivita
+ eliminaEntita(id : int, conn : ConnectToDatabase) : Attivita
- attivitaRepository
Attività
- id : int
- descrizione : String
- prezzo : double
+ Attivita(id : int, descrizione : String, prezzo : double)
key : String
ListaAttivita
+ ListaAttivita()
+ ricercaAttivita(id : int) : Attivita
+ aggiungiAttivita(attivita : Attivita) : Attivita
- listaAttivita

Fase di Elaborazione – 2° Iterazione

3. Introduzione

Dopo aver terminato la 1° iterazione della fase di elaborazione si procede alla 2° iterazione della fase di elaborazione in cui ci si focalizzerà sul secondo e sul quinto caso d'uso, ovvero quelli relativi alla gestione degli Impianti Sportivi e alla prenotazione degli stessi.

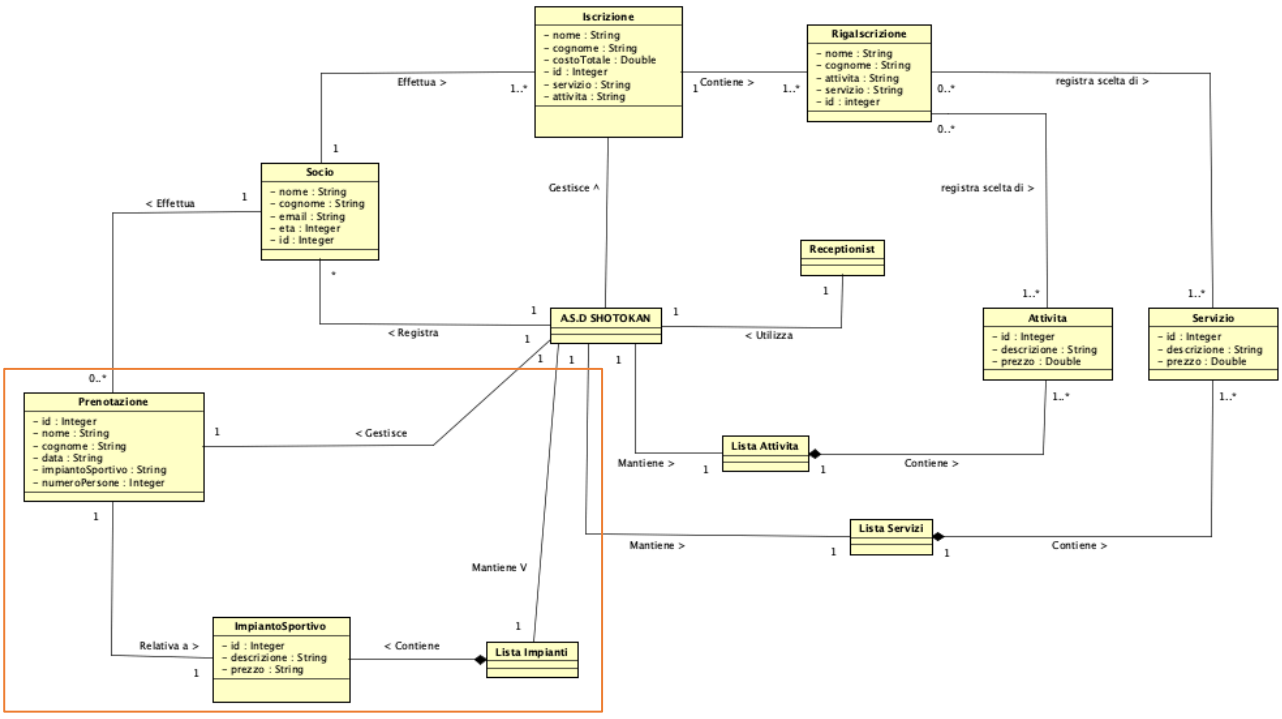
3.2 Analisi orientata agli oggetti

Proprio come per la fase 1 dell'elaborazione si e utilizzano gli stessi strumenti (Modello di Dominio, Diagrammi di Sequenza SSD e Contratti delle operazioni) per affinare l'analisi orientata agli oggetti.
Nel caso relativo al UC2 (Gestione Impianti Sportivi) e UC5 (Gestione Prenotazione Impianto Sportivo), per poter garantire lo scenario di successo vengono aggiunte allo schema di partenza ulteriori classi (si veda Prenotazione e Impianti Sportivi):

- **Prenotazione Impianti Sportivi**
  Indica la nuova operazione che l'operatore di sala fitness (receptionist) potrà effettuare
- **Impianti Sportivi**
  Sono strutture che possono essere prenotate dai soci (e non) della palestra. Basterà in effetti comunicare all'operatore un nominativo al fine di poter usufruire dell'impianto desiderato.

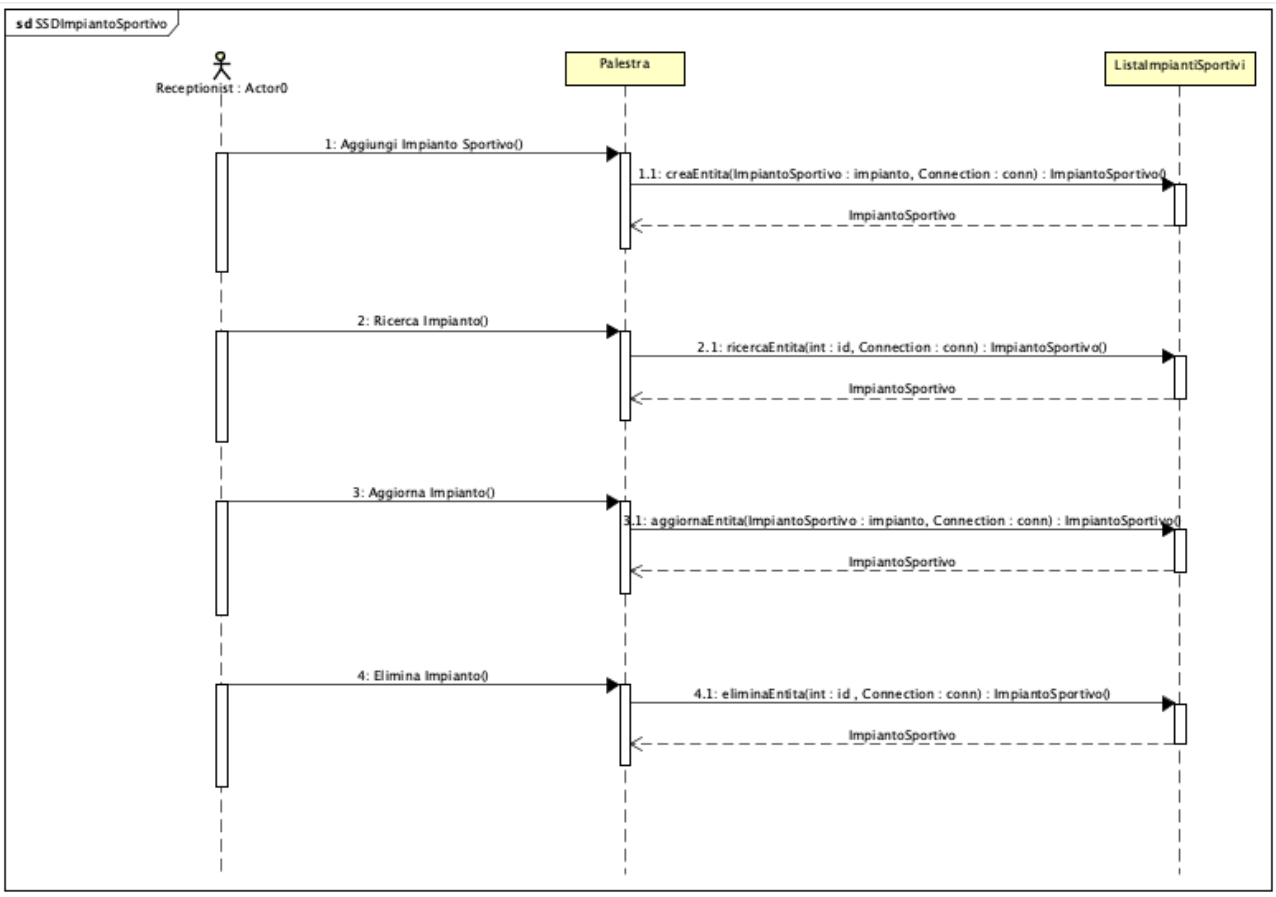
3.2.1 Modello di dominio

Tenuto conto degli elementi sopra, per i due casi d'uso considerati avremo il seguente nuovo **Modello di Dominio** :

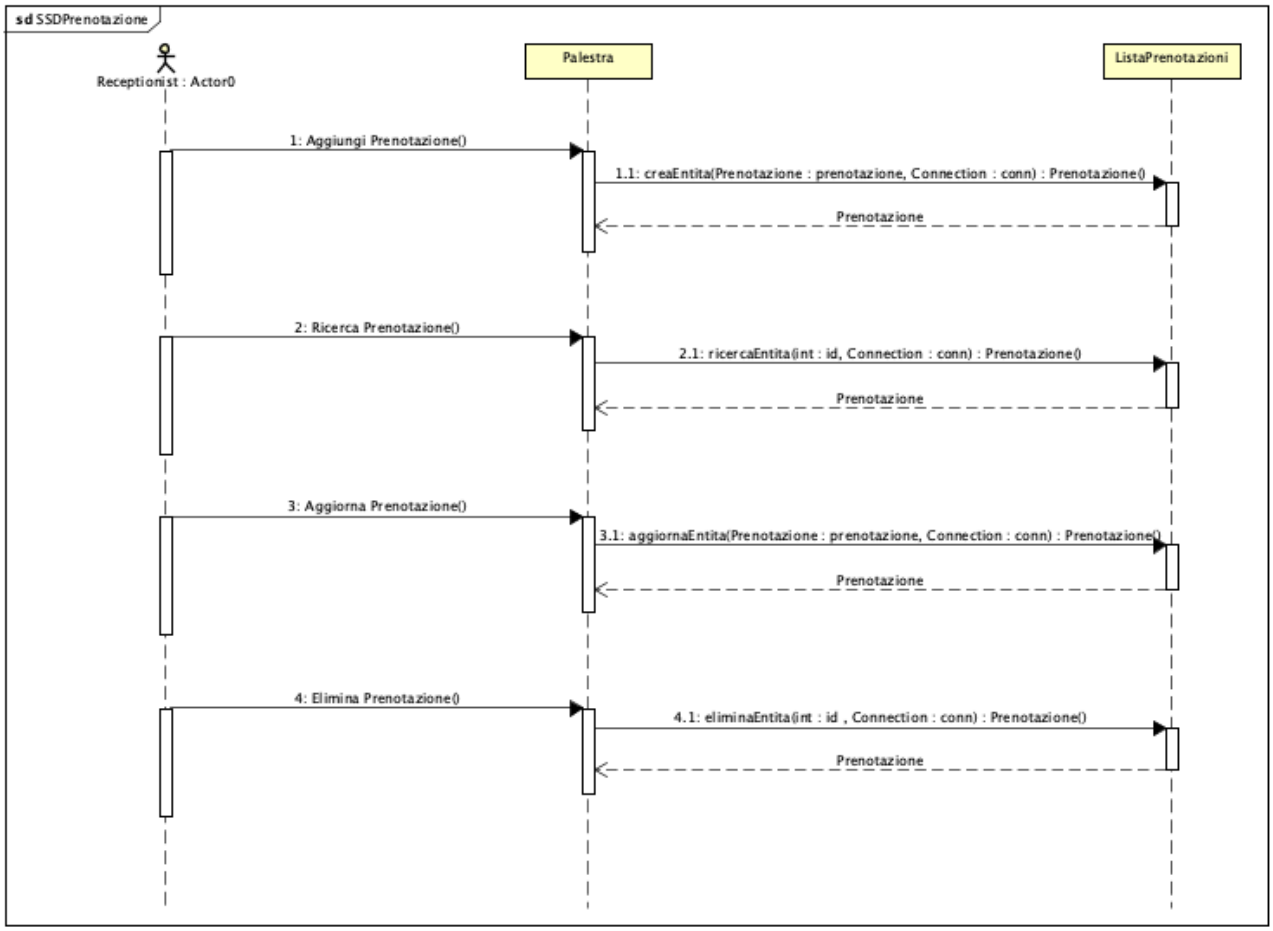


3.2.2 Diagrammi di sequenza

Per quanto riguarda invece il Diagramma di sequenza (SSD), per lo UC2 avremo :



Per l'UC5 avremo invece:



3.2.3 Contratti delle operazioni

Il terzo elemento fondamentale per la definizione completa dell'analisi orientata agli oggetti è dato dai **Contratti delle operazioni**, che descrivono gli eventi di sistema elaborati nel Diagramma di Sequenza (SSD).

Contratto CO1 : creaEntita(impianto : ImpiantoSportivo, conn : Connection)

Operazione:

- **creaEntita(iscrizione :Iscrizione, conn : Connection) : ImpiantoSportivo**

Riferimenti:

- UC2: Gestisci Impianto

Pre-Condizioni:

- è in corso l'operazione di creazione impianto

Contratto CO2 : ricercaEntita(id : int, conn: Connection)

Operazione:

- **ricercaEntita(id : int, conn: Connection) : ImpiantoSportivo**

Riferimenti:

- UC2: Gestisci Impianto

Pre-Condizioni:
- è in corso l'operazione di ricerca impianto

Contratto CO3 : aggiornaEntita(impianto : ImpiantoSportivo, conn : Connection)

Operazione:
- **modificaEntita(impianto : ImpiantoSportivo, conn : Connection) : Impianto Sportivo**

Riferimenti:
- UC2: Gestisci Impianto

Pre-Condizioni:
- è in corso l'operazione di modifica delle informazioni dell'impianto

Contratto CO4 : eliminaEntita(id : int , conn : Connection)

Operazione:
- **eliminaEntita(id : int , conn : Connection) : Impianto Sportivo**

Riferimenti:
- UC2: Gestisci Impianto

Pre-Condizioni:
- è in corso l'operazione di eliminazione dell'impianto

Contratto CO5 : listaEntita(conn : Connection)

Operazione:
- **listaEntita (conn : Connection) : List<ImpiantoSportivo>**

Riferimenti:
- UC2: Gestisci Impianto

Pre-Condizioni:
- è in corso l'operazione che restituisce una lista di Impianti sportivi

Quelli relativi allo UC5 (prenotazione Impianti sportivi) sono:

Contratto CO1 : creaEntita(prenotazione : Prenotazione, conn : Connection)

Operazione:
- **creaEntita(prenotazione : Prenotazione, conn : Connection) : Prenotazione**

Riferimenti:
- UC5: Gestisci Prenotazione Impianto

Pre-Condizioni:
- è in corso l'operazione di creazione Prenotazione

Contratto CO2 : ricercaEntita(id : int, conn: Connection)

Operazione:
- **ricercaEntita(id : int, conn: Connection) : Prenotazione**

Riferimenti:
- UC5: Gestisci Prenotazione Impianto

Pre-Condizioni:

- è in corso l’operazione di ricerca impianto

Contratto CO3 : aggiornaEntita(prenotazione : Prenotazione, conn : Connection)

Operazione:
- **modificaEntita(prenotazione : Prenotazione, conn : Connection) : Prenotazione**

Riferimenti:
- UC5: Gestisci Prenotazione Impianto

Pre-Condizioni:
- è in corso l’operazione di modifica della prenotazione dell’impianto

Contratto CO4 : eliminaEntita(id : int , conn : Connection)

Operazione:
- **eliminaEntita(id : int , conn : Connection) : Prenotazione**

Riferimenti:
- UC5: Gestisci Prenotazione Impianto

Pre-Condizioni:
- è in corso l’operazione di eliminazione di una Prenotazione

Contratto CO5 : listaEntita(conn : Connection)

Operazione:
- **listaEntita (conn : Connection) : List<Prenotazione>**

Riferimenti:
- UC5: Gestisci Prenotazione Impianto

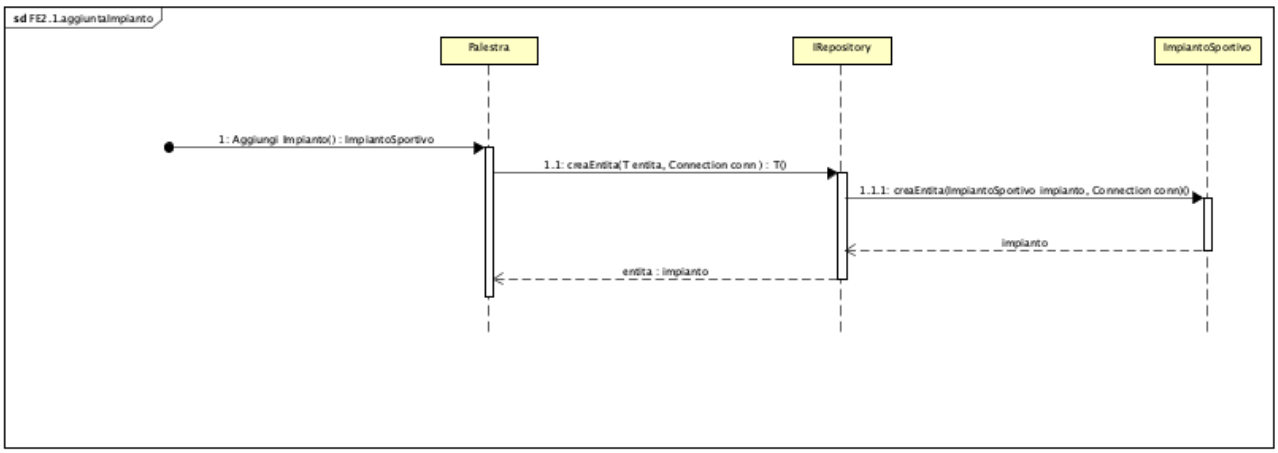
Pre-Condizioni:
- è in corso l’operazione che restituisce una lista di prenotazioni dell’impianto

3.3 Progettazione

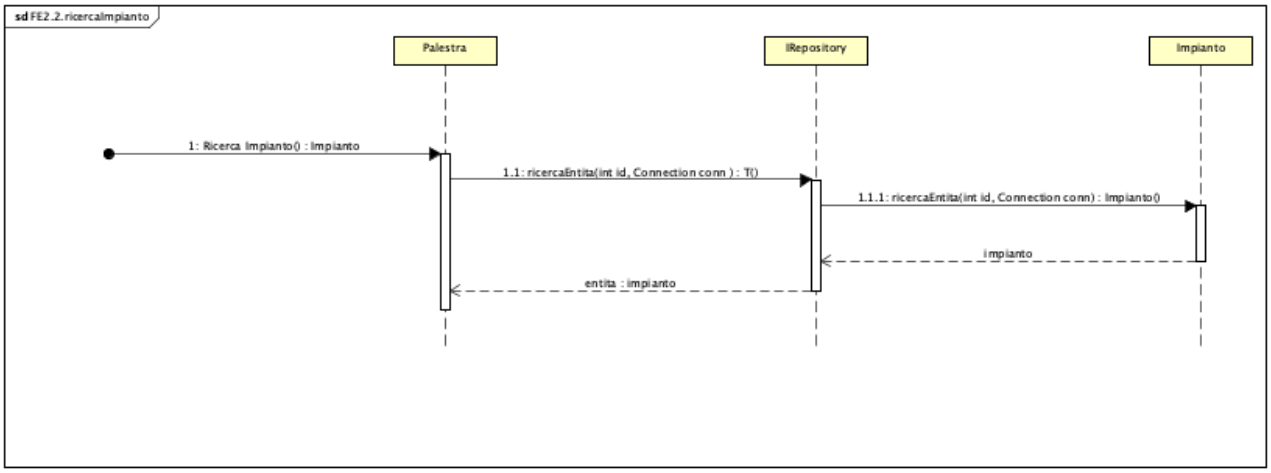
Al fine di poter definire gli oggetti software e le loro responsabilità, da un punto di vista statico o dinamico, come fato anche durante la prima fase di elaborazione, si definisce un Modello di progetto, consistente in una serie di Diagrammi di Sequenza (SSD) e delle Classi.

3.3.1 Diagrammi di Sequenza relativi al caso d'uso UC2 : Gestione Impianti Sportivi

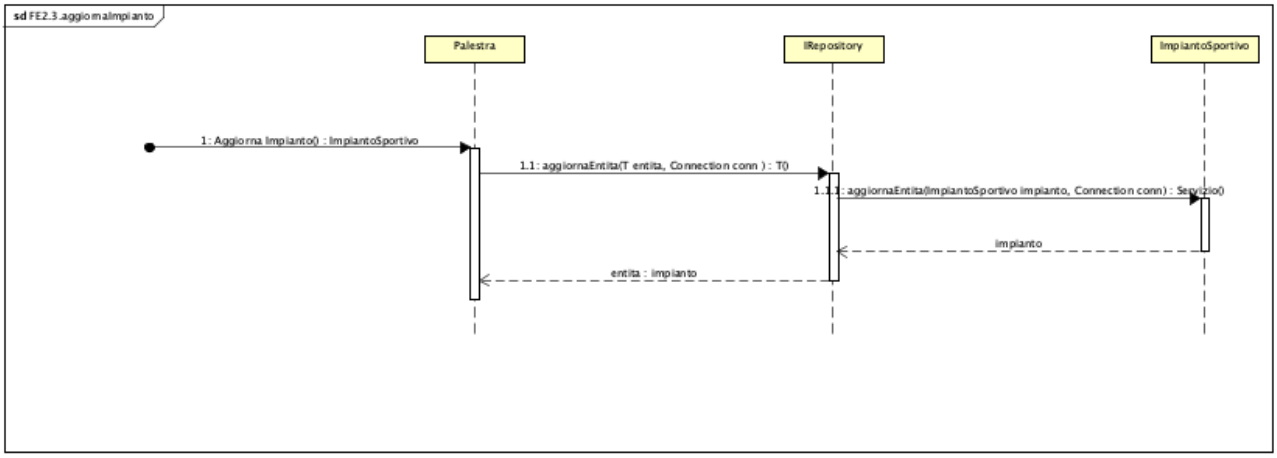
- Aggiungi Impianto



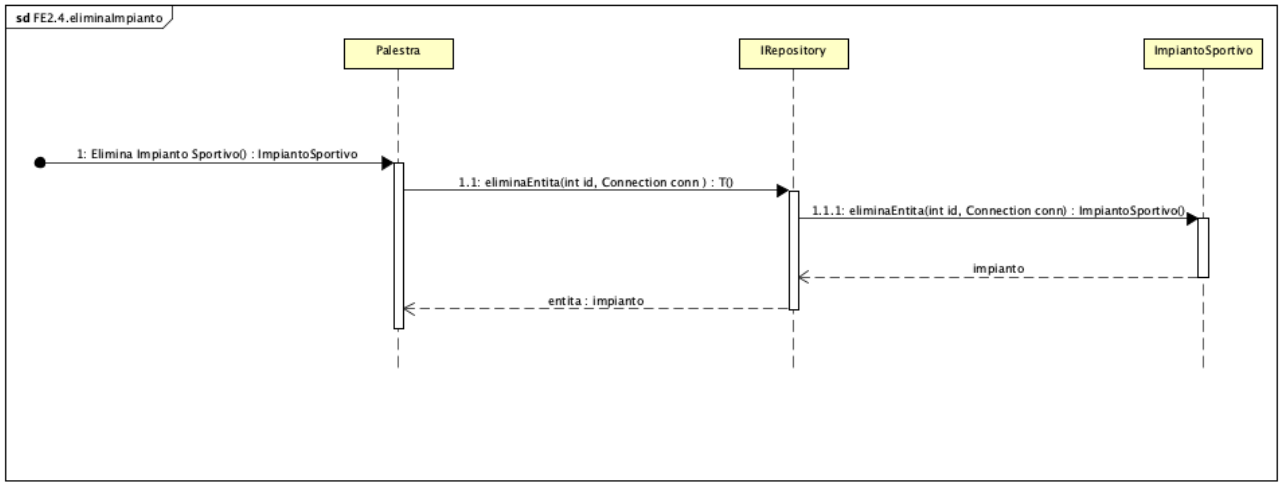
- Ricerca Impianto



- Aggiorna Impianto

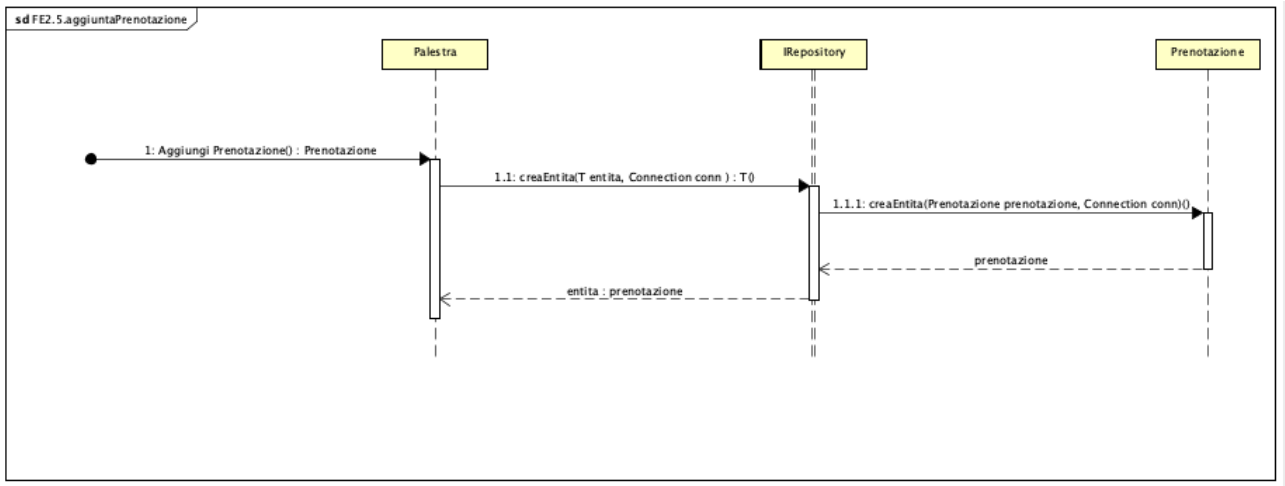


- Elimina Impianto

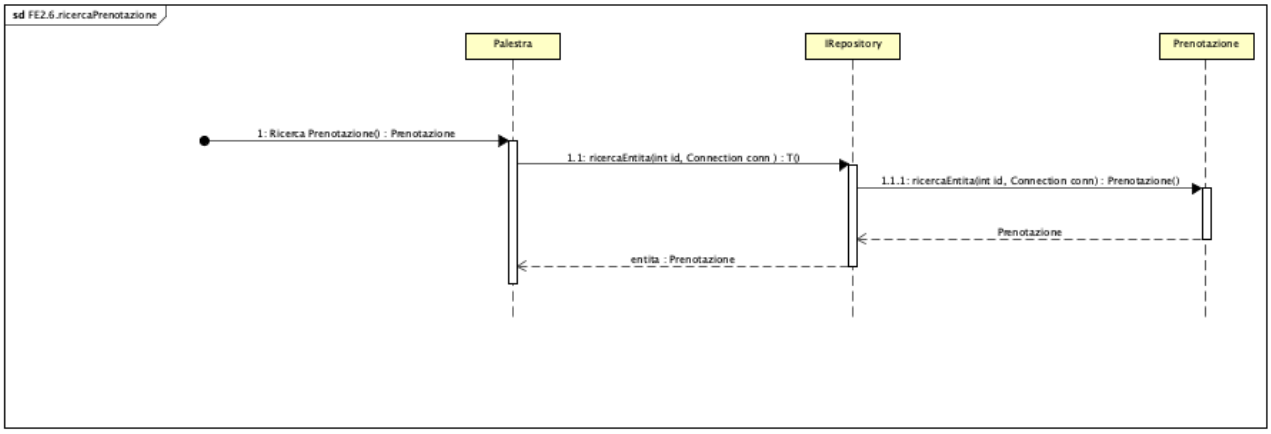


3.3.2 Diagrammi di Sequenza relativi al caso d'uso UC5 : Gestione Prenotazione Impianti Sportivi

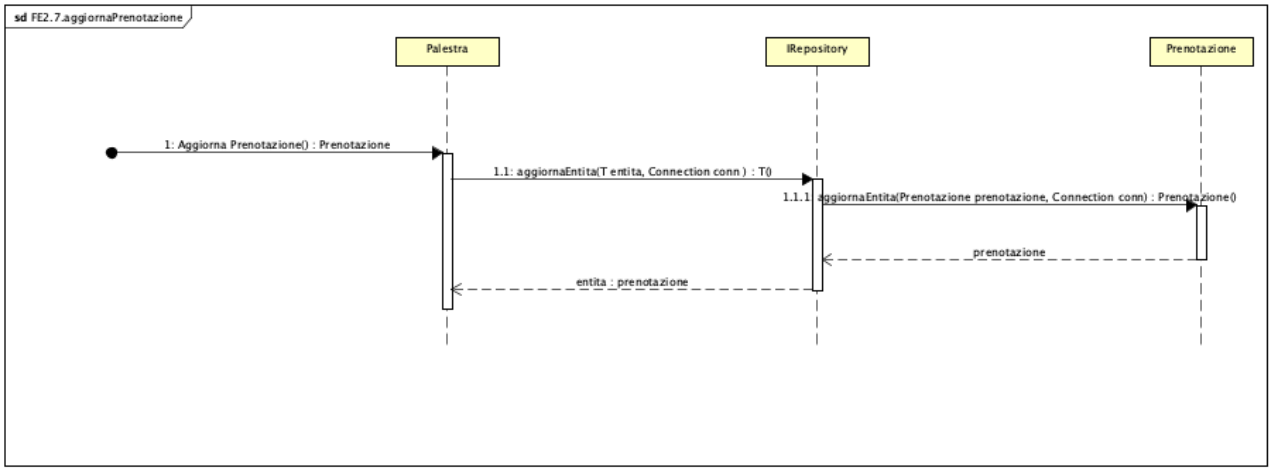
- Aggiungi Prenotazione



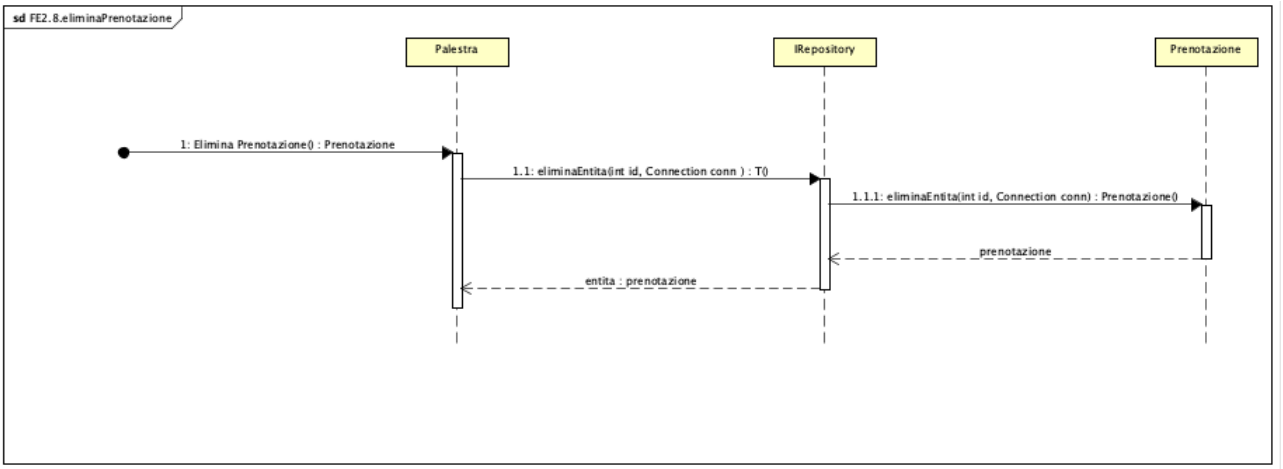
- Ricerca Prenotazione



- Aggiorna Prenotazione

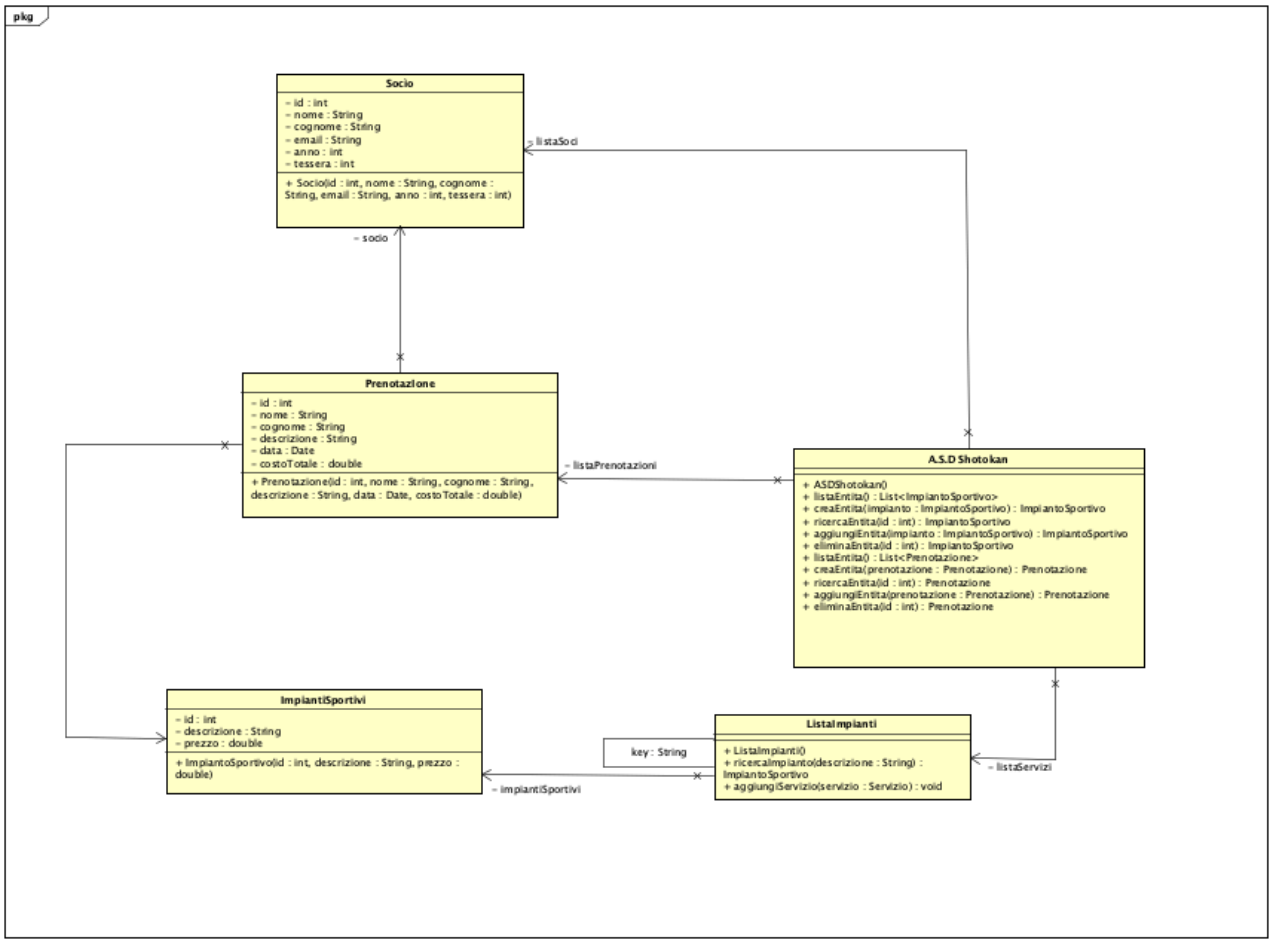


- Elimina Prenotazione



3.3.3 Diagramma delle classi

Al fine di rendere più chiara l'immagine si dispone la stessa in verticale sulla prossima pagina (l'immagine e il file in formato asta si trovano nella cartella Elaborazione (Iter2) > Diagramma delle classi).



Fase di Elaborazione – 3° Iterazione

4. Introduzione

Dopo aver terminato la 2° iterazione della fase di elaborazione si procede alla 3° iterazione in cui ci si focalizzerà sul terzo e sul quarto caso d'uso, ovvero quelli relativi alla gestione delle Attività e dei servizi.

4.2 Analisi orientata agli oggetti

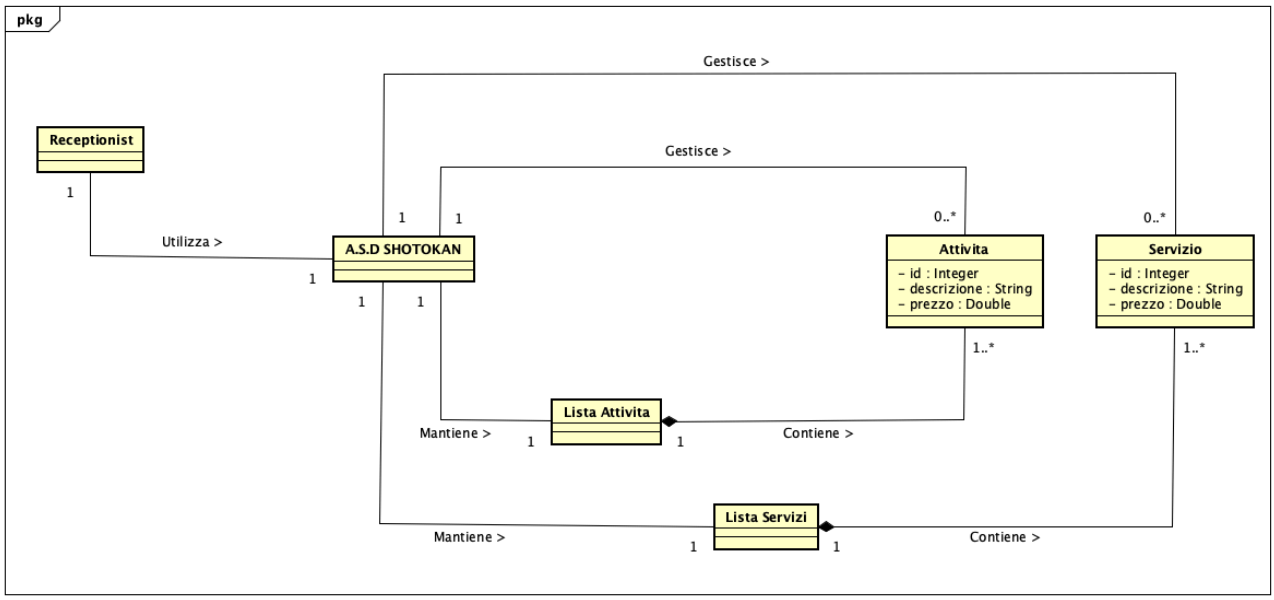
Proprio come per la prime 2 fasi dell'elaborazione si utilizzano gli stessi strumenti (Modello di Dominio, Diagrammi di Sequenza SSD e Contratti delle operazioni)per affinare l'analisi orientata agli oggetti.
Nel caso relativo al UC3 (Gestione Attività) e UC4 (Gestione Servizi), per poter garantire lo scenario di successo vengono usate le classi Attività e Servizio relative alla prima iterazione:

- **Gestione Attività**
  Implica il livello di interazione e le operazioni (CRUD + ricerca delle attivita e catalogo) che vengono eseguite dal sistema circa le attività tra le quali l'utente può scegliere in fase di iscrizione.

- **Gestione Servizi**
  Implica il livello di interazione e le operazioni (CRUD + ricerca dei servizi e catalogo) che vengono eseguite dal sistema circa i servizi tra le quali l'utente può scegliere in fase di iscrizione.

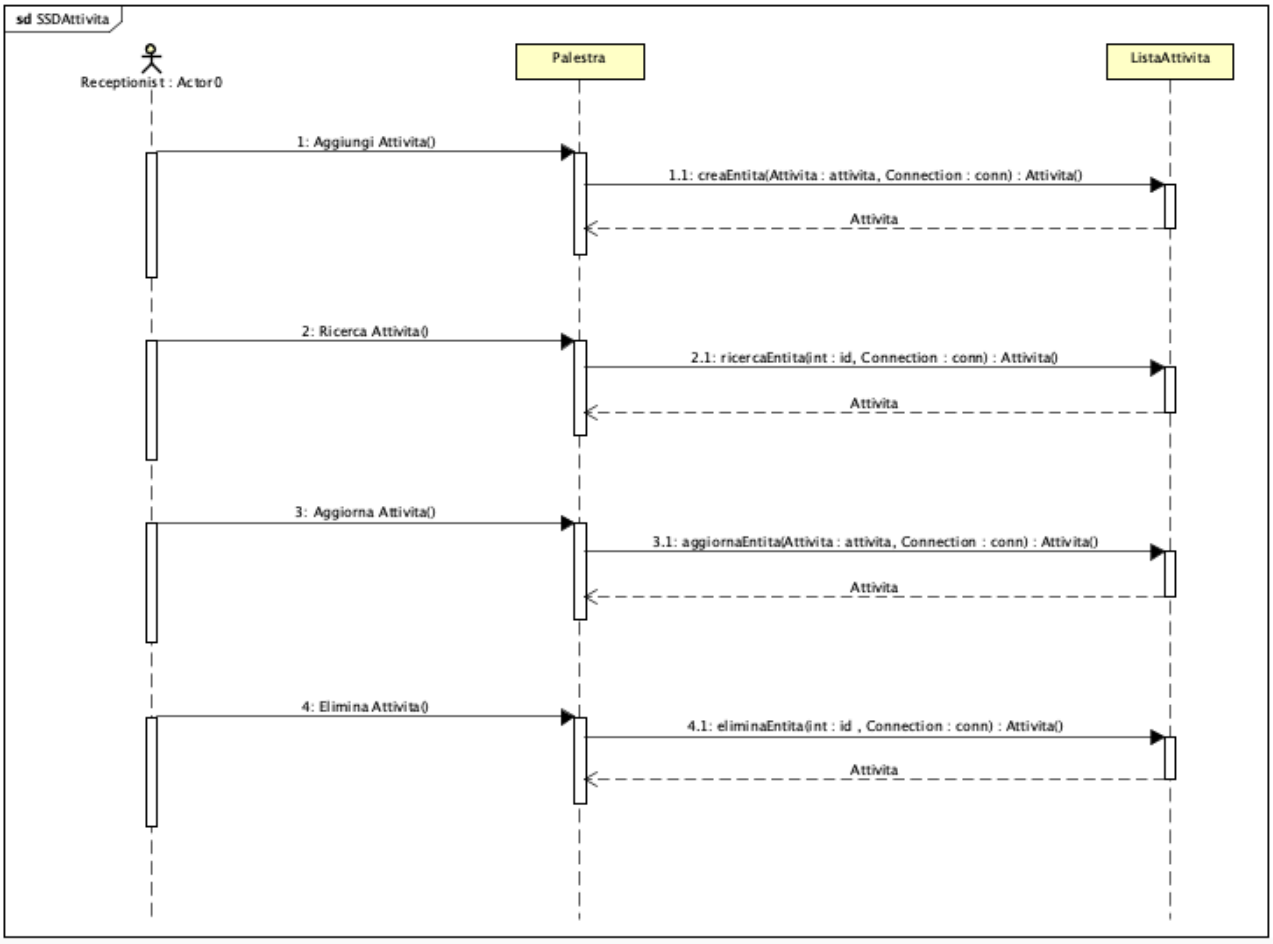
4.2.1 Modello di dominio

Tenuto conto degli elementi sopra, per i due casi d'uso considerati avremo il seguente nuovo **Modello di Dominio** :

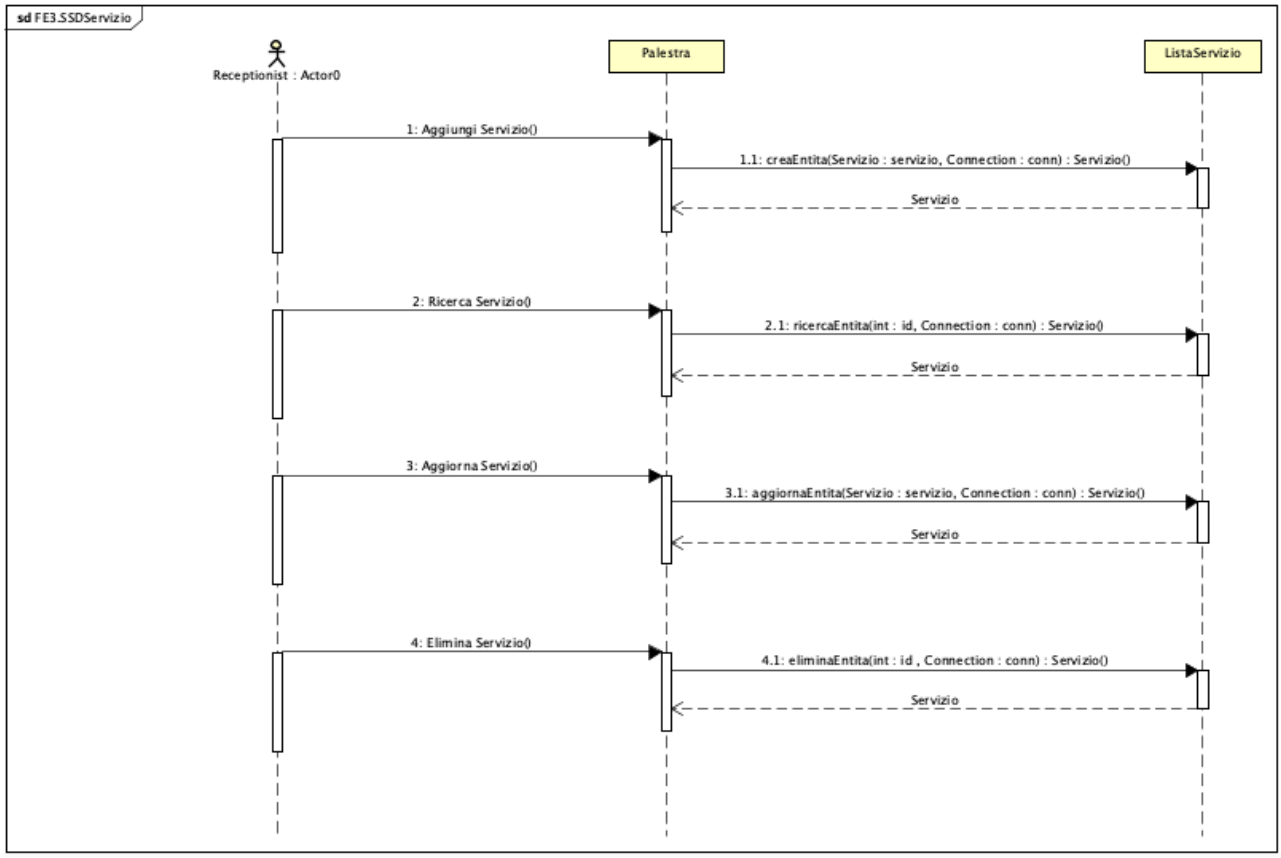


4.2.2 Diagramma di Sequenza (SSD)

Il secondo da introdurre è il Diagramma di Sequenza (SSD), che nell'UC3 relativo alla gestione delle attività sarà il seguente:



Mentre nel caso UC4 relativo alla gestione dei Servizi avremo :



4.2.3 Contratti delle operazioni

Quelli relativi allo UC3 (gestione attività) sono:

Contratto CO1 : creaEntita(attivita: Attivita, conn : Connection)

Operazione:

- **creaEntita(attivita: Attivita, conn : Connection) : Attivita**

Riferimenti:

- UC3: Gestisci Attivita

Pre-Condizioni:

- è in corso l'operazione di creazione di una nuova Attivita

Contratto CO2 : ricercaEntita(id : int, conn: Connection)

Operazione:

- **ricercaEntita(id : int, conn: Connection) : Attivita**

Riferimenti:

- UC3: Gestisci Attivita

Pre-Condizioni:

- è in corso l'operazione di ricerca Attivita

Contratto CO3 : aggiornaEntita(attivita: Attivita, conn : Connection)

Operazione:
- **modificaEntita(attivita: Attivita, conn : Connection) : Attivita**

Riferimenti:
- UC3: Gestisci Attivita

Pre-Condizioni:
- è in corso l'operazione di modifica dell'attivita

Contratto CO4 : eliminaEntita(id : int , conn : Connection) : Attivita

Operazione:
- **eliminaEntita(id : int , conn : Connection)**

Riferimenti:
- UC3: Gestisci Attivita

Pre-Condizioni:
- è in corso l'operazione di eliminazione attivita

Quelli relativi allo UC4 per la gestione dei servizi sono:

Contratto CO1 : creaEntita(servizio: Servizio, conn : Connection)

Operazione:
- **creaEntita(servizio: Servizio, conn : Connection) : Servizio**

Riferimenti:
- UC4: Gestisci Servizio

Pre-Condizioni:
- è in corso l'operazione di creazione di un nuovo Servizio

Contratto CO2 : ricercaEntita(id : int, conn: Connection)

Operazione:
- **ricercaEntita(id : int, conn: Connection) : Servizio**

Riferimenti:
- UC4: Gestisci Servizio

Pre-Condizioni:
- è in corso l'operazione di ricerca Servizio

Contratto CO3 : aggiornaEntita(servizio: Servizio, conn : Connection)

Operazione:
- **modificaEntita(servizio: Servizio, conn : Connection) : Servizio**

Riferimenti:
- UC4: Gestisci Servizio

Pre-Condizioni:
- è in corso l'operazione di modifica servizio

Contratto CO4 : eliminaEntita(id : int , conn : Connection) : Servizio

Operazione:
- **eliminaEntita(id : int , conn : Connection)**

Riferimenti:
- UC4: Gestisci Servizio

Pre-Condizioni:
- è in corso l'operazione di eliminazione servizio

Contratto CO5 : listaServizio()

Operazione:
- **listaServizio** () : List<Servizio>

Riferimenti:
- UC4: Gestisci Servizio

Pre-Condizioni:
- è in corso l'operazione di ricerca di tutti i servizi

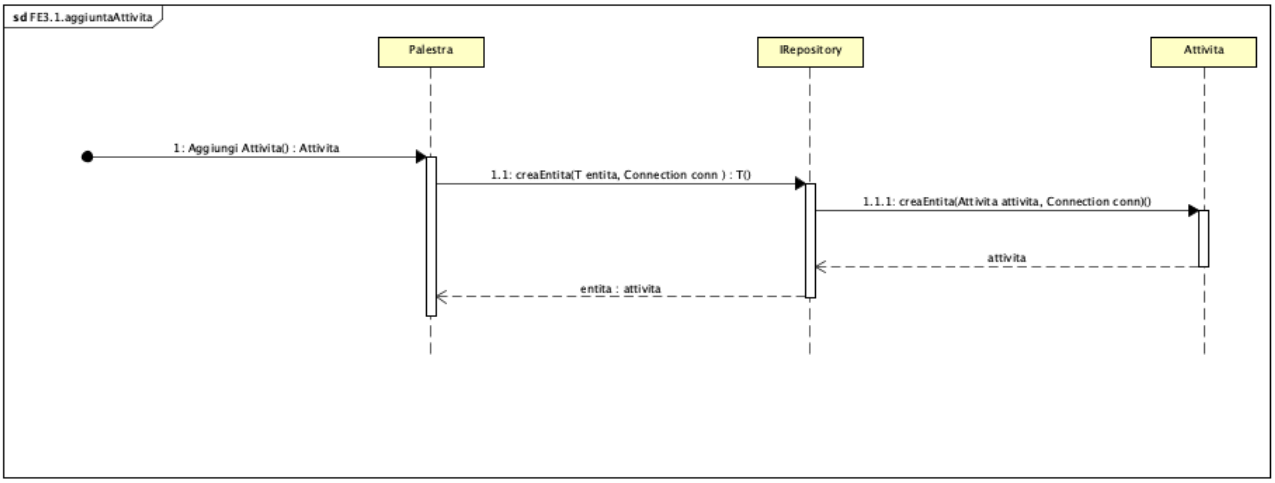
Post-condizioni:
- Restituisce una lista di Servizi

4.3 Progettazione

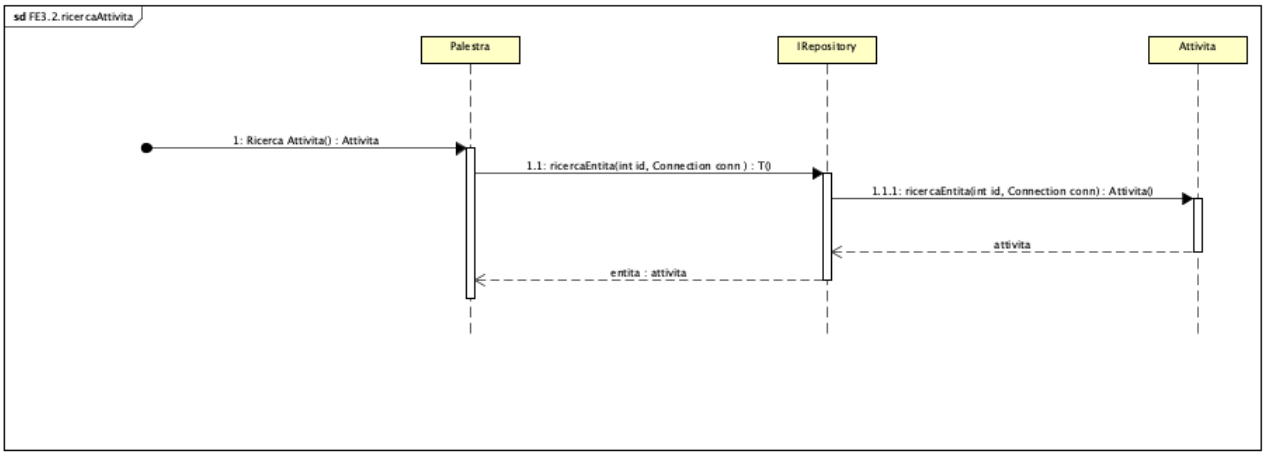
Si definisce il Modello di progetto, consistente in una serie di Diagrammi di Sequenza (SSD) e delle Classi.

4.3.1 Diagrammi di Sequenza relativi al caso d'uso UC3 : Gestione Attività

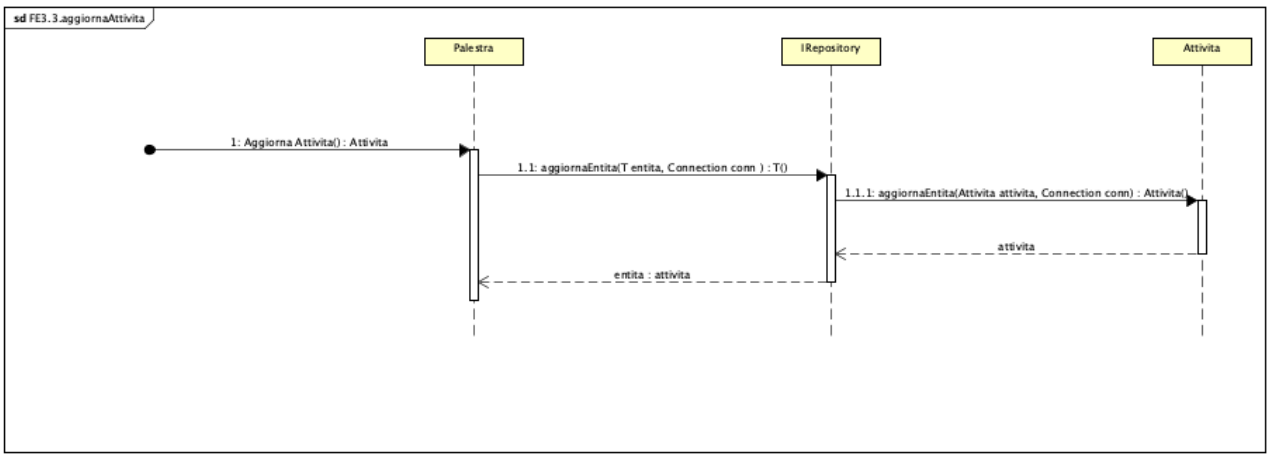
- Aggiungi Attività



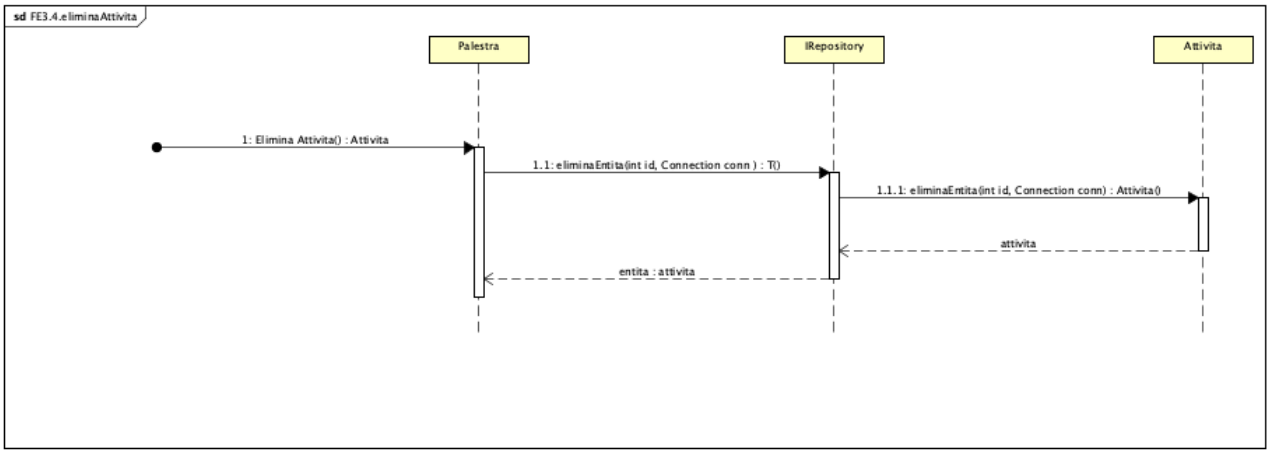
- Ricerca Attività



- Aggiorna Attività

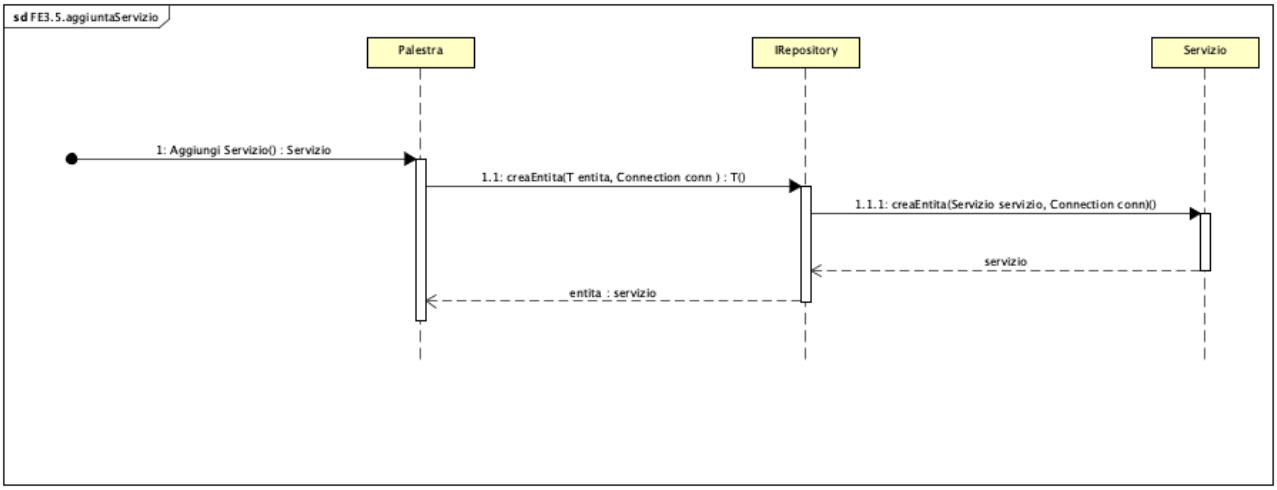


- Elimina Attività

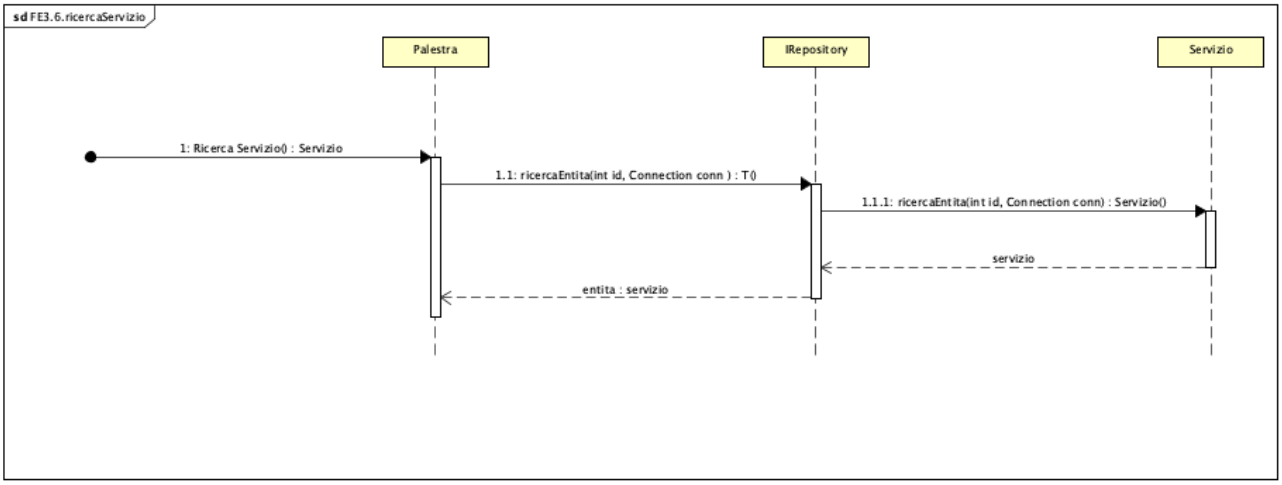


4.3.2 Diagrammi di Sequenza relativi al caso d'uso UC4 : Gestione Servizio

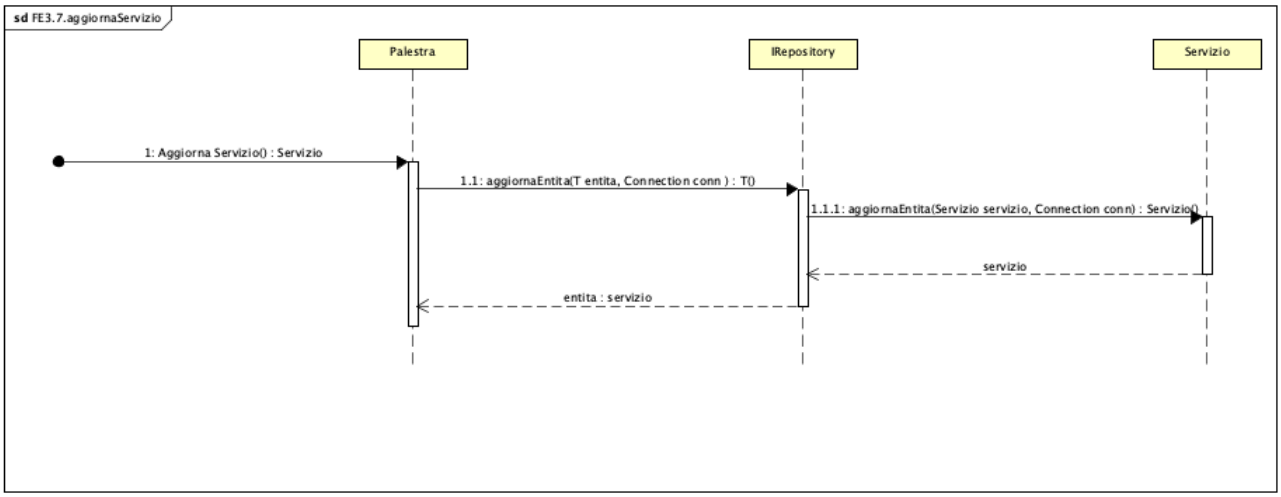
- Aggiungi Servizio



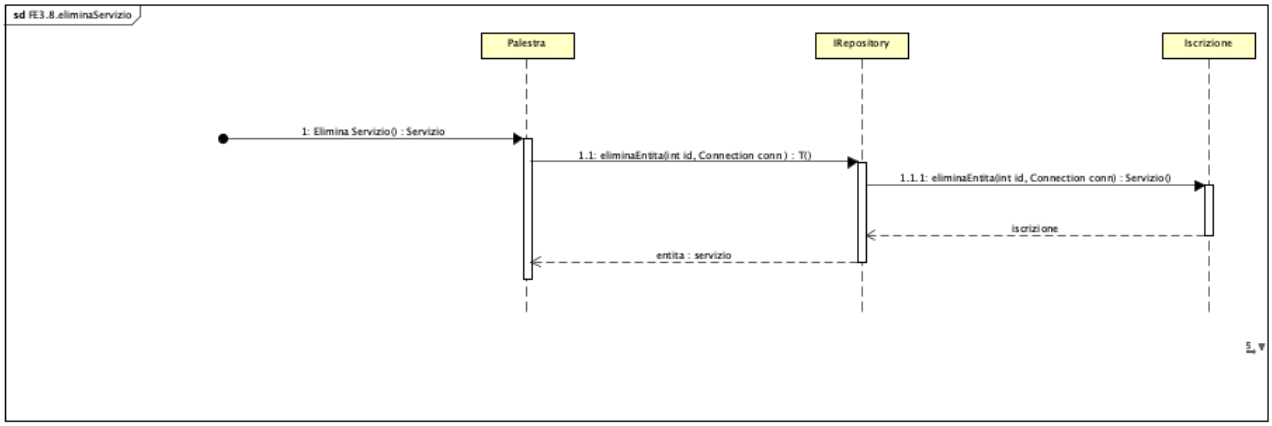
- Ricerca Servizio



- Aggiorna Servizio

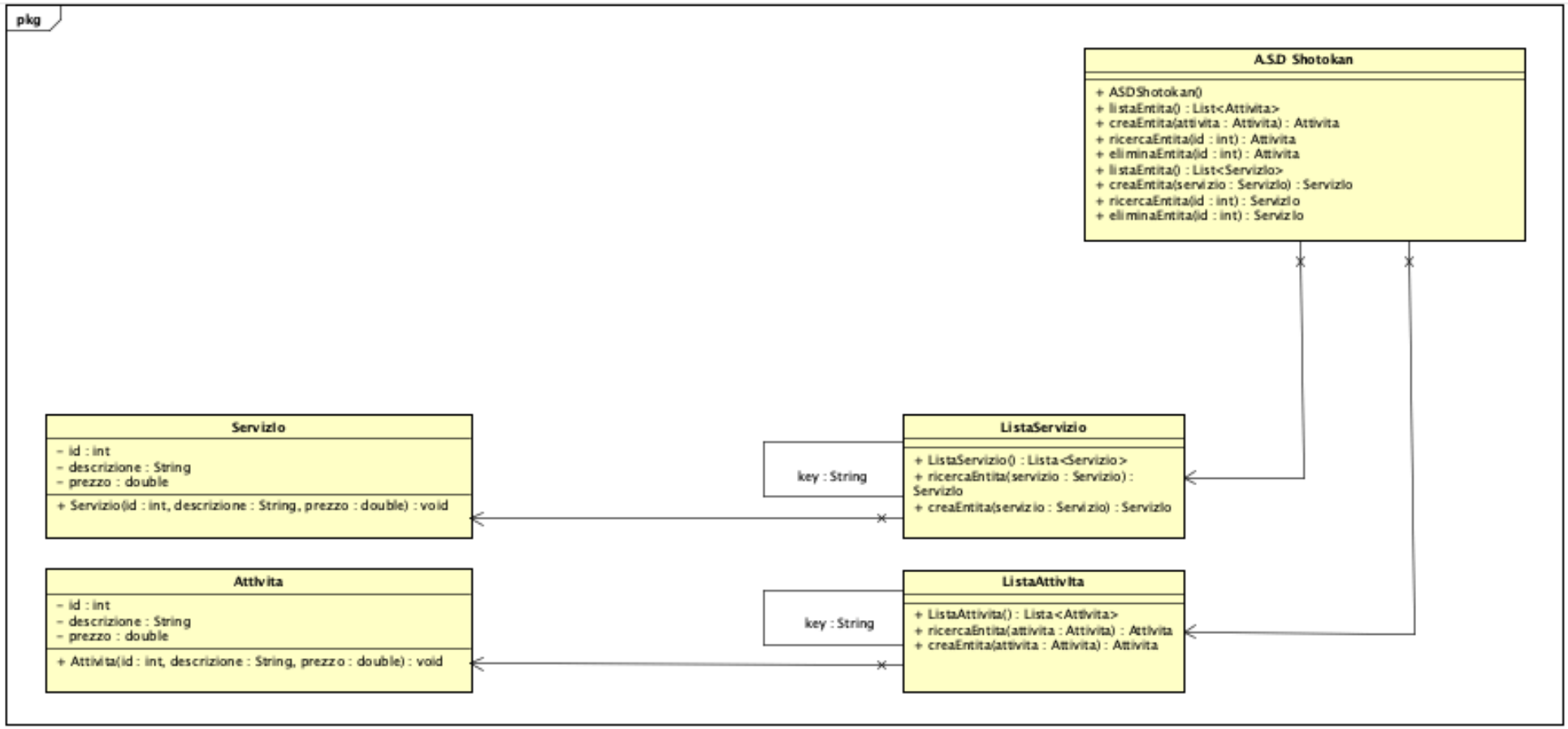


- Elimina Servizio



2.3.1 Diagramma delle classi

Al fine di rendere più chiara l'immagine si dispone la stessa in verticale e nella pagina successiva (l'immagine e il file in formato asta si trovano nella cartella Elaborazione (Iter3) > Diagramma delle classi).


pkg
A.S.D Shotokan
+ ASDShotokan()
+ listaEntita() : List<Attivita>
+ creaEntita(attivita : Attivita) : Attivita
+ ricercaEntita(id : int) : Attivita
+ eliminaEntita(id : int) : Attivita
+ listaEntita() : List<Servizio>
+ creaEntita(servizio : Servizio) : Servizio
+ ricercaEntita(id : int) : Servizio
+ eliminaEntita(id : int) : Servizio
Servizio
- id : int
- descrizione : String
- prezzo : double
+ Servizio(id : int, descrizione : String, prezzo : double) : void
key : String
ListaServizio
+ ListaServizio() : Lista<Servizio>
+ ricercaEntita(servizio : Servizio) : Servizio
+ creaEntita(servizio : Servizio) : Servizio
Attivita
- id : int
- descrizione : String
- prezzo : double
+ Attivita(id : int, descrizione : String, prezzo : double) : void
key : String
ListaAttivIta
+ ListaAttivita() : Lista<Attivita>
+ ricercaEntita(attivita : Attivita) : Attivita
+ creaEntita(attivita : Attivita) : Attivita

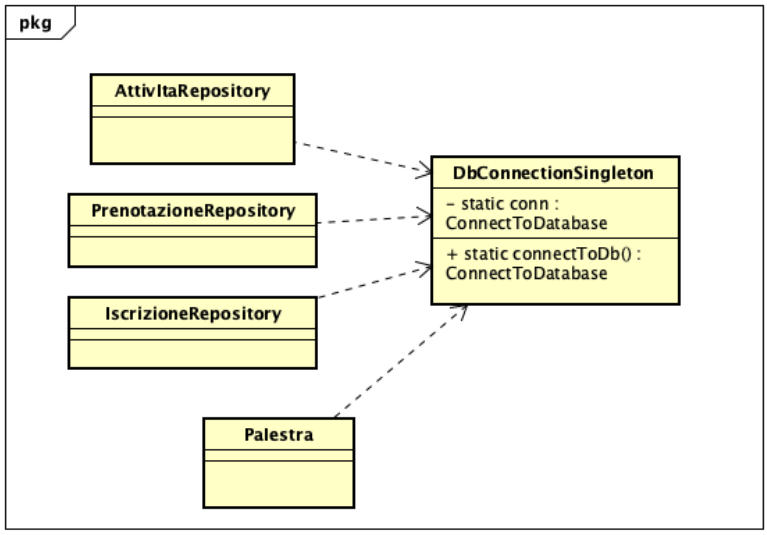
Pattern utilizzati per la progettazione

Per effettuare in maniera ottimale la progettazione sono stati implementati 3 pattern tra cui due creazionali (**Singleton e Prototype**) e uno comportamentale (**Command**).

1 . Singleton

Al fine di gestire correttamente la connessione al DB si è deciso di applicare il Singleton Pattern per creare un'unica istanza che verrà poi richiamata nella classe del metodo main() o in qualunque altra classe che ne abbia bisogno.

Diagramma delle classi Singleton:



Codice Singleton (che richiama la classe di connessione al DB → ConnectToDatabase):

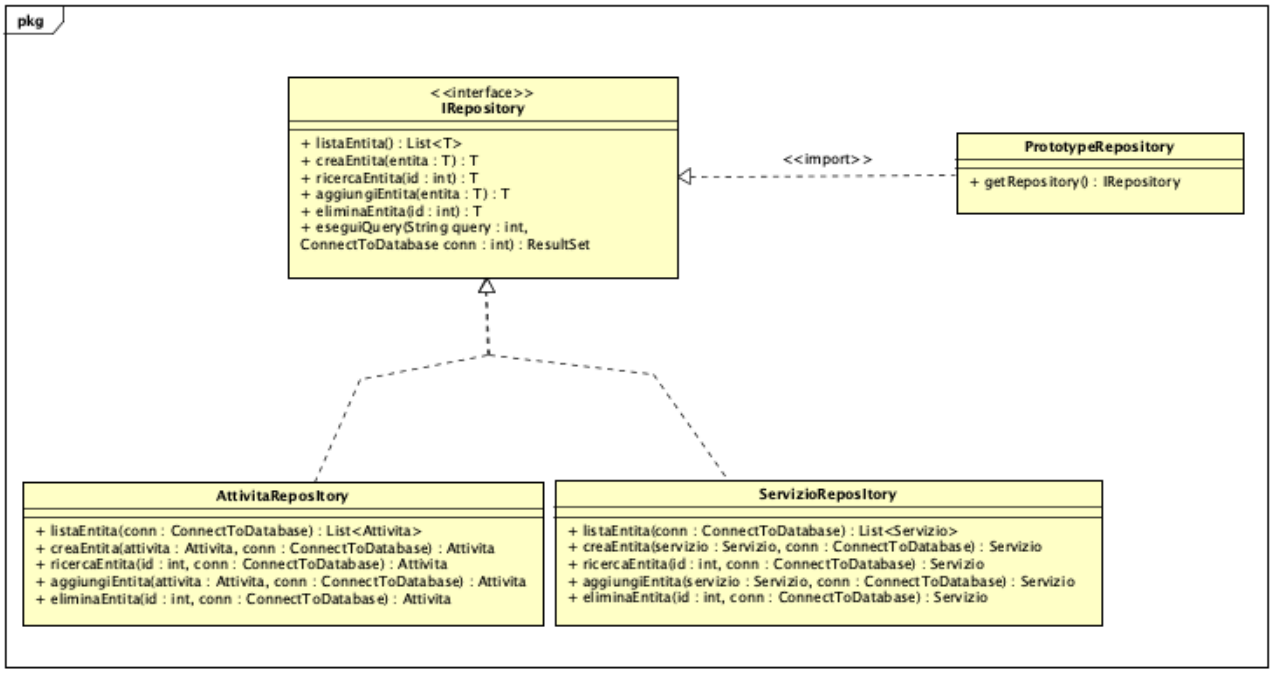
```
*/
public class DbConnectionSingleton {

    public static class Connect
    {
        private static ConnectToDatabase conn = null;  //questa è la mia instanza

        public static ConnectToDatabase connectToDb(){
            if(conn == null)
            {
                conn = new ConnectToDatabase();
            }
            return conn;
        }

    }

}
```

2 . Prototype

Al fine di poter effettuare una progettazione di tipo Generic , si è deciso di implementare il Prototype Pattern cosi da avere un'interfaccia univoca (**IRepository**) che definisca i metodi comuni necessari a tutte le classi.
Il principio base è quello di fare l'override di tali metodi nelle classi EntitaRepository (dove Entita verrà sostituito col nome delle diverse classi) e sulla base di ciò che si vuole gestire il file **RepositoryPrototype** creerà la giusta istanza della classe che potrà essere dunque utilizzata.
Ovviamente all'interno di ogni repository verranno implementati i metodi col tipo effettivo di classe.

N.B
La notazione "Repository" è stata usata al fine di identificare la corretta separazione fra lo strato di modello (Classi base come Attivita, Servizio, Socio, ecc ….) e quello di persistenza dei dati (ovvero il DB) tipico del Repository Pattern utilizzato per la progettazione di software secondo la struttura tipo del MVC (Model-View-Controller).

Diagramma delle classi Prototype:



Codice Prototype (per semplicità con una sola classe che implementa l'interfaccia, da esempio) :

```java
public interface IRepository<T> {

    public List<T> listaEntita() throws SQLException;
    public T creaEntita(T entity) throws SQLException;
    public T aggiornaEntita(T entity) throws SQLException;
    public boolean eliminaEntita(String[] chiavi) throws SQLException;
    public ResultSet eseguiQuery(String query) throws SQLException;
    public T ricercaEntita(String[] chiavi) throws SQLException;
}
```

Interfaccia -> IRepository

```java
public class PrenotazioneRepository implements IRepository<Prenotazione>  {

    List<Prenotazione> listaPrenotazione = new ArrayList<Prenotazione>();
    Statement stmt = null;
    ConnectToDatabase conn;



    @Override
    public List<Prenotazione> listaEntita() throws SQLException {...30 lines }

    @Override
    public Prenotazione creaEntita(Prenotazione entity) throws SQLException {...14 lines }

    @Override
    public Prenotazione aggiornaEntita(Prenotazione entity) throws SQLException {...13 lines }

    @Override
    public boolean eliminaEntita(String[] chiavi) throws SQLException {...18 lines }

    @Override
    public ResultSet eseguiQuery(String query) throws SQLException {...8 lines }

    @Override
    public Prenotazione ricercaEntita(String[] chiavi) throws SQLException {...25 lines }
```

Classe che implementa l'interfaccia

3 . Command

Per poter invece gestire e parametrizzare le chiamate ai metodi della repository si è deciso di applicare il pattern **Command**. Nel sistema ogni comando è visto come un oggetto separato che può essere passato e invocato (mediante il metodo execute()). L'interfaccia generica istanziata è definita come ICommand.

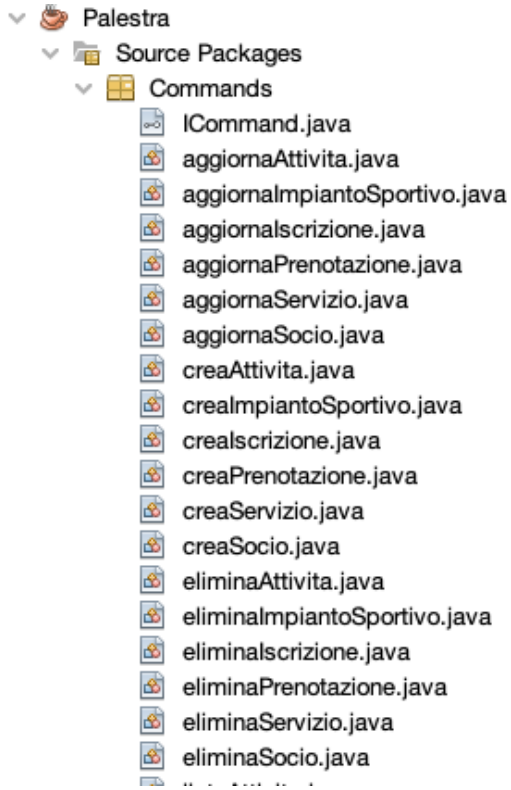
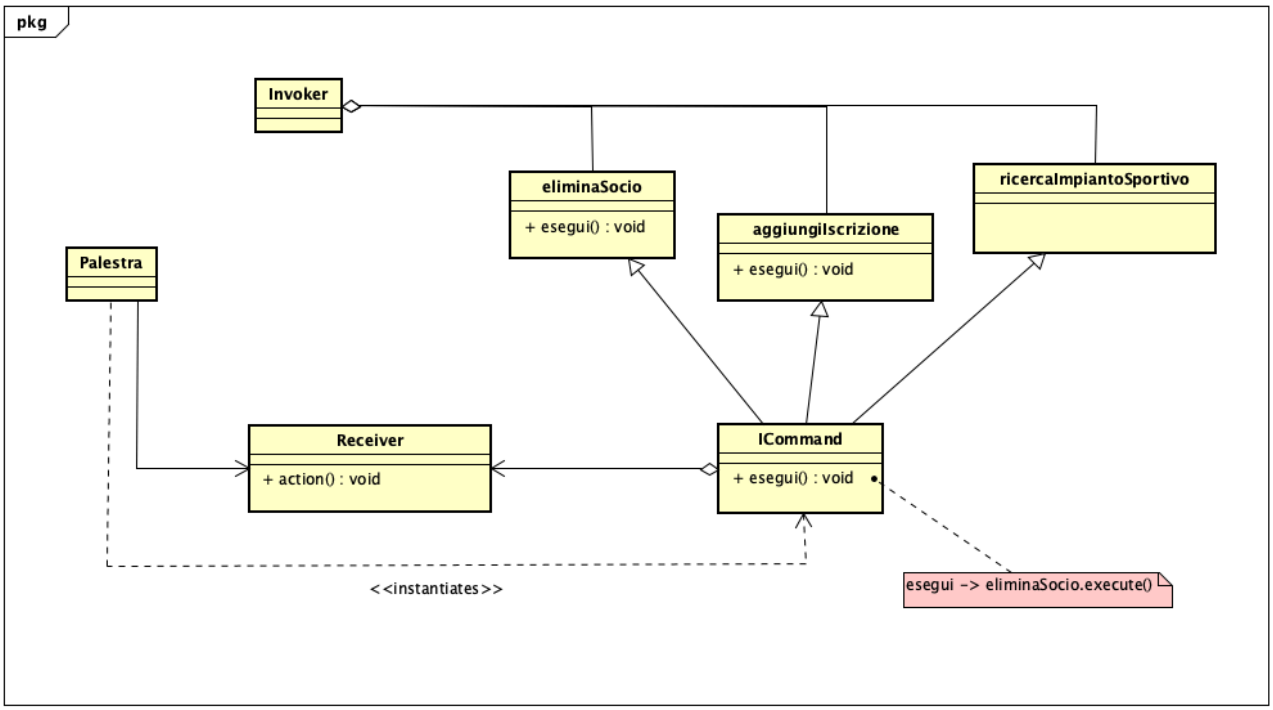


Diagramma delle classi Command (per semplicità si mettono solo alcune classi a scopo illustrativo):



Codice Command:

```
*/
public interface ICommand {

    public static void execute() throws ParseException, SQLException {};

}
```

Interfaccia -> ICommand

```
public class aggiornaAttivita implements ICommand{

    private Attivita attivita;
    ConnectToDatabase conn = new DbConnectionSingleton.Connect().connectToDb();


       static RepositoryPrototype prototypeRepo = new RepositoryPrototype();
       static Scanner sc = new Scanner(System.in);

       public aggiornaAttivita()
       {

       }

   public static void execute() throws ParseException, SQLException
   {
       prototypeRepo.setRepository("Attivita");

           System.out.println("----------------------------------------");
           System.out.println("Modifica Informazioni Attivita ");
           System.out.println("----------------------------------------");
           System.out.println("Inserisci descrizione attivita");
           String descrizione = sc.nextLine();
           System.out.println("Inserisci prezzo attivita");
           double prezzo = new Scanner(System.in).nextDouble();
           prototypeRepo.repo.aggiornaEntita(new Attivita(descrizione, prezzo));

   }

}
```

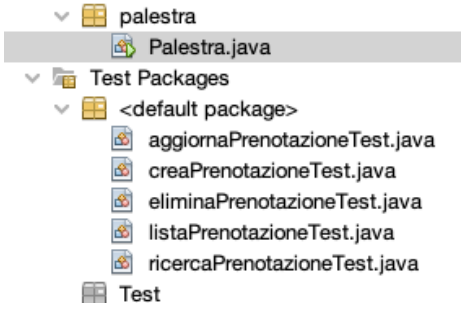
Classe che implementa l'interfaccia

Test in jUnit

Al fine di testare i metodi realizzati si è utilizzato jUnit (tramite librerie importate sul progetto realizzato con l'IDE NetBeans). Per semplicità e tempo si è deciso di effettuare il test dei metodi più significativi, relativi alla "Prenotazione di impianti sportivi".

Nello specifico i test racchiudono i classici metodi quali:

- Ricerca di una prenotazione (per sapere se un impianto è occupato in un giorno specifico)
- Aggiornamento di una prenotazione (cambio data ad esempio)
- Eliminazione della prenotazione (una volta che l'utente ha usufruito dell'impianto sportivo)
- Lista di tutte le prenotazioni (per consentire una ricerca fluida)
- Creazione di una nuova prenotazione relativa ad un impianto sportivo.



I metodi testati si trovano (nel progetto Palestra), sotto la directory Test Package, dove vengono collocati singolarmente alla creazione di un nuovo file di test jUnit.

Si procede dunque all'illustrazione di come sono stati effettuati i test.

1 Test Lista Prenotazione

```
@Test
public void testListaPrenotazione() throws SQLException  {

    RepositoryPrototype prototypeRepo = new RepositoryPrototype();
    Statement stmt;
    ConnectToDatabase conn = new ConnectToDatabase();
    Connection c = conn.connect();
    stmt = c.createStatement();
    stmt.executeUpdate("DELETE FROM prenotazione");

    prototypeRepo.setRepository("Prenotazione");

       ArrayList<Prenotazione> listPrenotazione = new ArrayList<>();
       Prenotazione prenotazione = new Prenotazione("Mario", "Rossi", "Campo Calcio", "19-07-2022");
       Prenotazione prenotazione1 = new Prenotazione("Giuseppe", "Verdi", "Campo Tennis", "20-07-2022");
       listPrenotazione.add(prenotazione);
       listPrenotazione.add(prenotazione1);
       prototypeRepo.repo.creaEntita(prenotazione);
       prototypeRepo.repo.creaEntita(prenotazione1);

       List<Prenotazione> a = prototypeRepo.repo.listaEntita();
       var d = listPrenotazione;

       for(int i = 0; i<a.size() && i<d.size(); i++){
       assertEquals(d.get(i).Nome , a.get(i).Nome);
       assertEquals(d.get(i).Cognome , a.get(i).Cognome);
       assertEquals(d.get(i).Impianto , a.get(i).Impianto);
       assertEquals(d.get(i).Data , a.get(i).Data);
       }
    }
```

2 Test Crea Prenotazione

```
@Test
public void testCreaPrenotazione() throws SQLException  {

  Statement stmt;
  ConnectToDatabase conn = new ConnectToDatabase();
  Connection c = conn.connect();
  stmt = c.createStatement();
  stmt.executeUpdate("DELETE FROM prenotazione");
  RepositoryPrototype prototypeRepo = new RepositoryPrototype();

  Prenotazione prenotazione = new Prenotazione("Francesco", "Maurici", "Campo Calcio", "19-07-2022");

  prototypeRepo.setRepository("Prenotazione");

  prototypeRepo.repo.creaEntita(prenotazione);

  Prenotazione d = new Prenotazione(" ", " ", " ", " ");

  assertFalse(prenotazione.Nome.equals(d.Nome));
}
```

3 Test Ricerca Prenotazione

```
@Test
 public void testRicercaPrenotazione() throws SQLException  {

    RepositoryPrototype prototypeRepo = new RepositoryPrototype();

    Statement stmt;
    ConnectToDatabase conn = new ConnectToDatabase();
    Connection c = conn.connect();
    stmt = c.createStatement();
    stmt.executeUpdate("DELETE FROM prenotazione");

   Prenotazione prenotazione = new Prenotazione("Daniele", "Giusti", "Campo Basket", "25-06-2022");

   prototypeRepo.setRepository("Prenotazione");

   prototypeRepo.repo.creaEntita(prenotazione);
   String[] param = new String[2];
   param[0] = "Daniele";
   param[1] = "Giusti";
   var a = prototypeRepo.repo.ricercaEntita(param);
   var d = (Prenotazione) a;
   assertTrue(prenotazione.Nome.equals(d.Nome));
   assertTrue(prenotazione.Cognome.equals(d.Cognome));
 }
```

4 Test Aggiorna Prenotazione

```
@Test
 public void testAggiornaPrenotazione() throws SQLException  {

   Statement stmt;
   ConnectToDatabase conn = new ConnectToDatabase();
   Connection c = conn.connect();
   stmt = c.createStatement();
   stmt.executeUpdate("DELETE FROM prenotazione");
   RepositoryPrototype prototypeRepo = new RepositoryPrototype();

   prototypeRepo.setRepository("Prenotazione");

   Prenotazione prenotazione = new Prenotazione("Vincenzo", "Zimbone", "Campo Calcio", "19-07-2022");
   prototypeRepo.repo.creaEntita(prenotazione);

   Prenotazione aggiornamento = new Prenotazione("Vincenzo", "Zimbone", "Campo Calcio", "27-07-2022");
   prototypeRepo.repo.aggiornaEntita(aggiornamento);

   assertTrue(prenotazione.Nome.equals(aggiornamento.Nome));
   assertTrue(prenotazione.Cognome.equals(aggiornamento.Cognome));
   assertTrue(prenotazione.Impianto.equals(aggiornamento.Impianto));
   assertFalse(prenotazione.Data.equals(aggiornamento.Data));
 }
```

5 Test Elimina Prenotazione

```
@Test
 public void testEliminaPrenotazione() throws SQLException  {

   Statement stmt;
   ConnectToDatabase conn = new ConnectToDatabase();
   Connection c = conn.connect();
   stmt = c.createStatement();
   stmt.executeUpdate("DELETE FROM prenotazione");

   RepositoryPrototype prototypeRepo = new RepositoryPrototype();

   prototypeRepo.setRepository("Prenotazione");
   Prenotazione prenotazione = new Prenotazione("Mario", "Rossi", "Campo Tennis", "19-07-2022");

   prototypeRepo.repo.creaEntita(prenotazione);
   String[] param = new String[2];
   param[0] = "Campo Calcio";
   param[1] = "19-07-2022";
   boolean a = prototypeRepo.repo.eliminaEntita(param);
   var b = true;
   assertTrue(a = b);

 }
```