Sommario

[1. Introduzione 1](#_Toc127467833)

[1.1 Object Design Goals 1](#_Toc127467834)

[1.2 Linee guida per la documentazione dell’interfaccia 2](#_Toc127467835)

[1.3 Definizioni, Acronimi e Abbreviazioni 2](#_Toc127467836)

[1.4 Riferimenti 2](#_Toc127467837)

[2. Packages 3](#_Toc127467838)

[3. Class Interfaces 10](#_Toc127467839)

[4. Class Diagram 21](#_Toc127467840)

[5. Design Patterns 21](#_Toc127467841)

# 1. Introduzione

HappyFields si propone di semplificare la pianificazione di match sportivi, al fine di incrementare l’interesse delle persone verso questi e di rendere meno lento il processo di organizzazione.

# 1.1 Object Design Goals

**Riusabilità:**

Il sistema deve basarsi sulla riusabilità attraverso l’uso corretto di ereditarietà e design patterns.

**Robustezza:**

Il sistema deve risultare pronto a gestire situazioni impreviste ed errori attraverso la gestione delle eccezioni

**Incapsulamento:**

Il sistema deve garantire l'invisibilità degli aspetti implementativi attraverso l’uso di interfacce, rendendo possibile l’utilizzo delle singole componenti come black-box.

# 

# 1.2 Linee guida per la documentazione dell’interfaccia

Le linee guida includono una lista di regole da rispettare per la corretta costruzione di interfacce grafiche. Per queste si è fatto riferimento alle convenzioni proposte sul sito w3school:

* <https://www.w3schools.com/html/html5_syntax.asp>
* https://checkstyle.sourceforge.io/sun\_style.html

# 1.3 Definizioni, Acronimi e Abbreviazioni

Di seguito alcune definizioni di termini presenti successivamente nel documento

* **Package:** Raggruppamento di classi o interfacce
* **Design Pattern:** Template riutilizzabili di soluzioni a problemi ricorrenti
* **Interfaccia:** Insieme delle firme alle operazioni offerte dalla classe
* **View:** Nel modello MVC rappresenta ciò che l’utente ha a disposizione per interagire con le funzioni applicative
* **lowerCamelCase:** La pratica di scrivere frasi in modo tale che, ad eccezione della prima, ogni parola inizi con lettera maiuscola e non abbia spazi con la precedente
* **UpperCamelCase:** La pratica di scrivere frasi in modo tale che, a partire della prima, ogni parola inizi con lettera maiuscola e non abbia spazi con la precedente

# 1.4 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura

* SDD

# 2. Packages

In questa sezione viene mostrata la suddivisione del sistema in packages, in base a quanto definito nel documento di SDD. Tale suddivisione è determinata dalle scelte architetturali e ricalca la struttura convenzionalmente definita da un progetto Maven.

* **.idea**
* **.mvn**
* **src**
  + **main**
    - **java** -> Contiene le classi Java relative alle componenti Control e Model
    - **webapp** -> Contiene tutti i file relativi alla componente View
  + **test** 
    - **java** -> Contiene tutte le classi di test
* **target** →Tutti i file prodotti da Maven

## 

**Package HappyFields**

Nella seguente sezione si presentano i package creati a partire da ogni sottosistema individuato nel System Design Document.

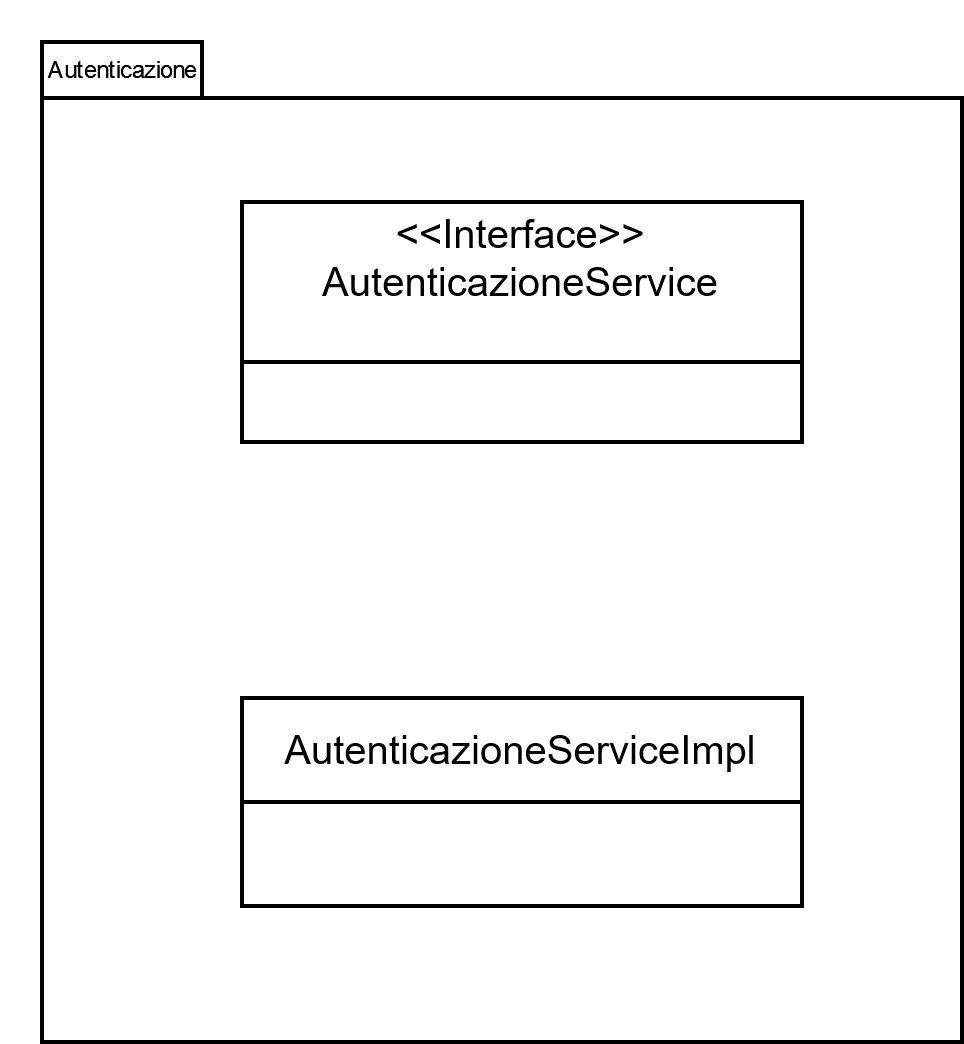
Si è scelto di creare un package separato per il Model contenente tutte le entity e le classi utilità per la connessione al DB.

Un ulteriore package Controller contiene tutte le servlet che implementano la logica applicativa.

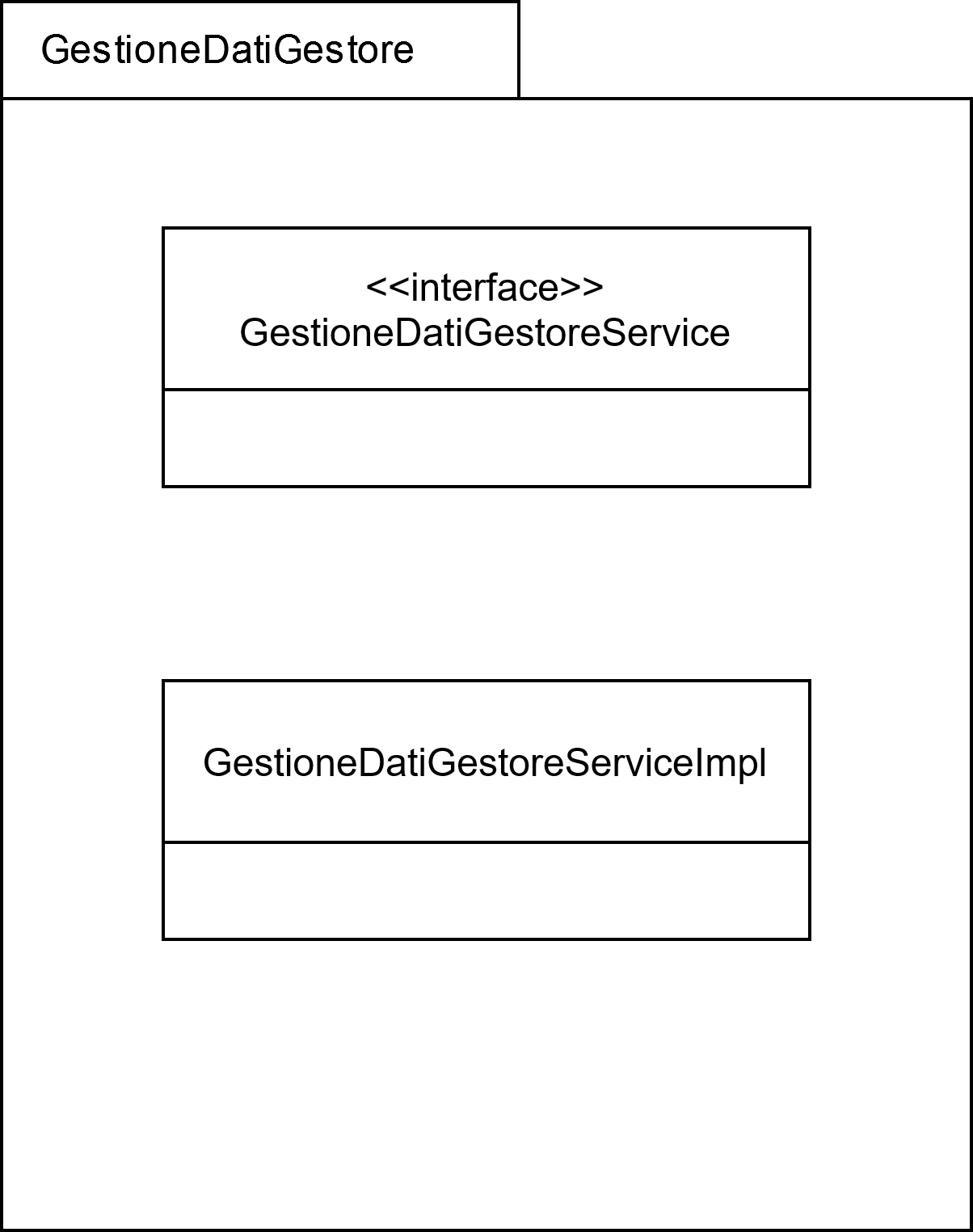
**Package Registrazione**



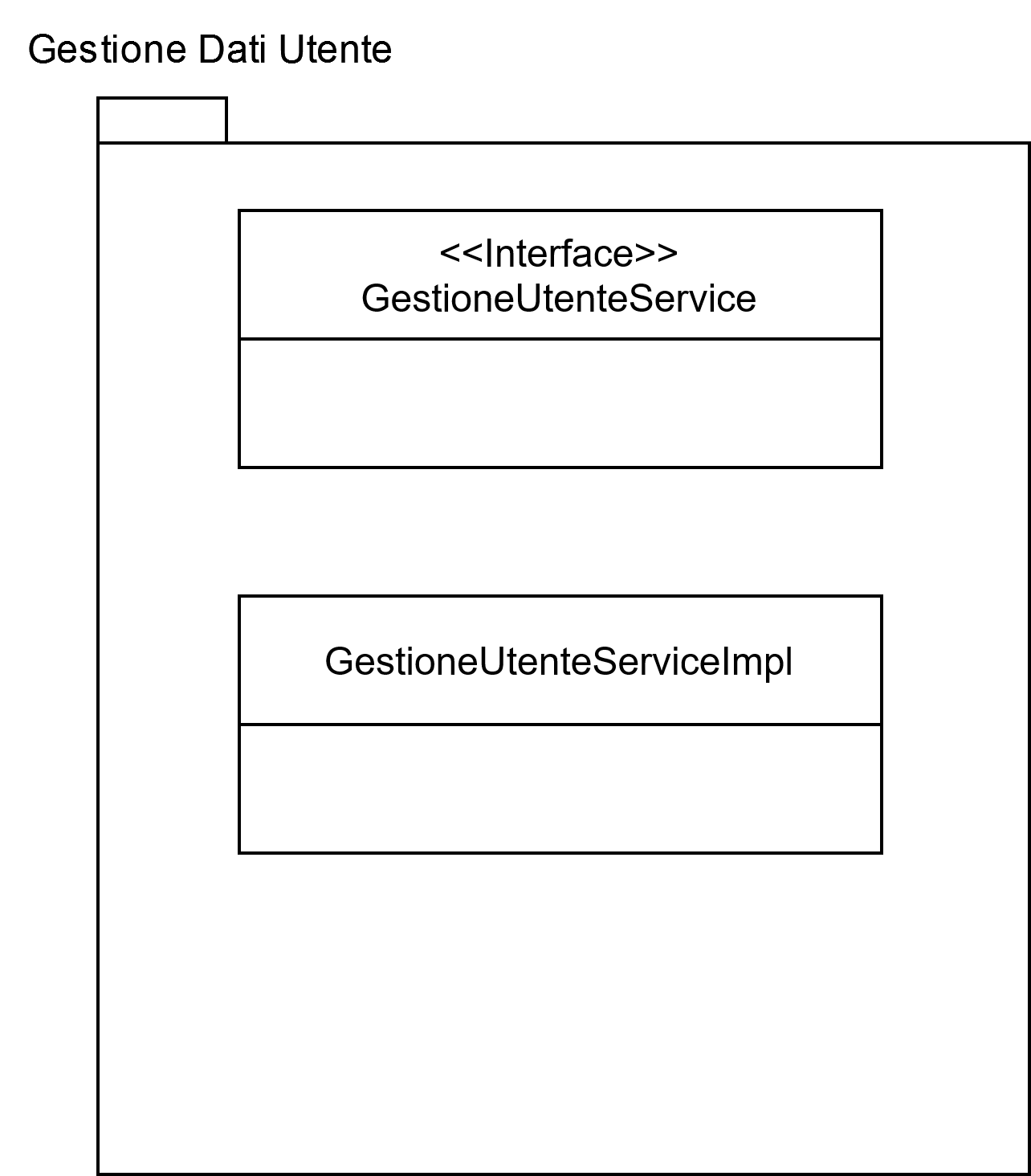
**PackageAutenticazione**



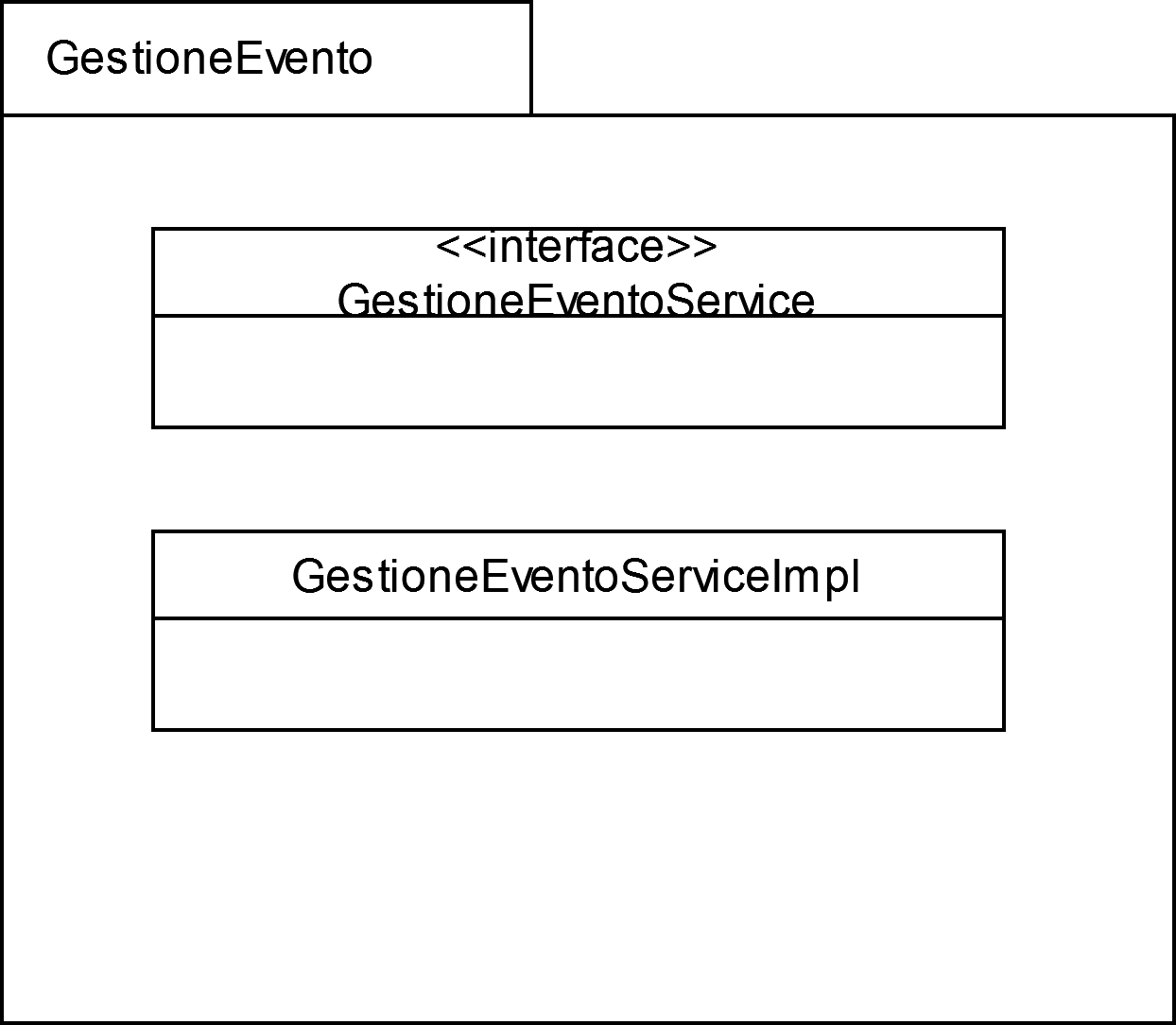
**Package gestioneDatiGestore**



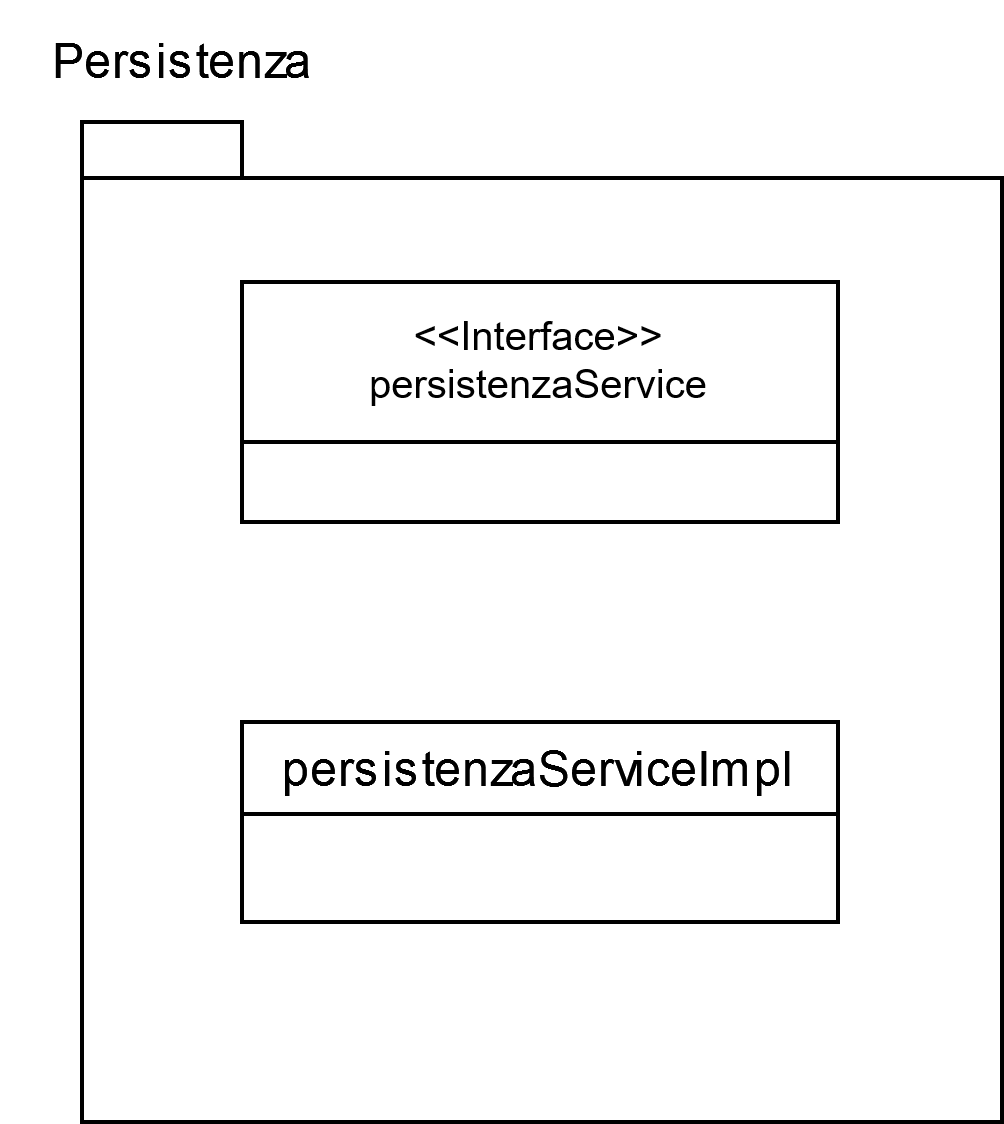
**Package gestioneDatiUtente**



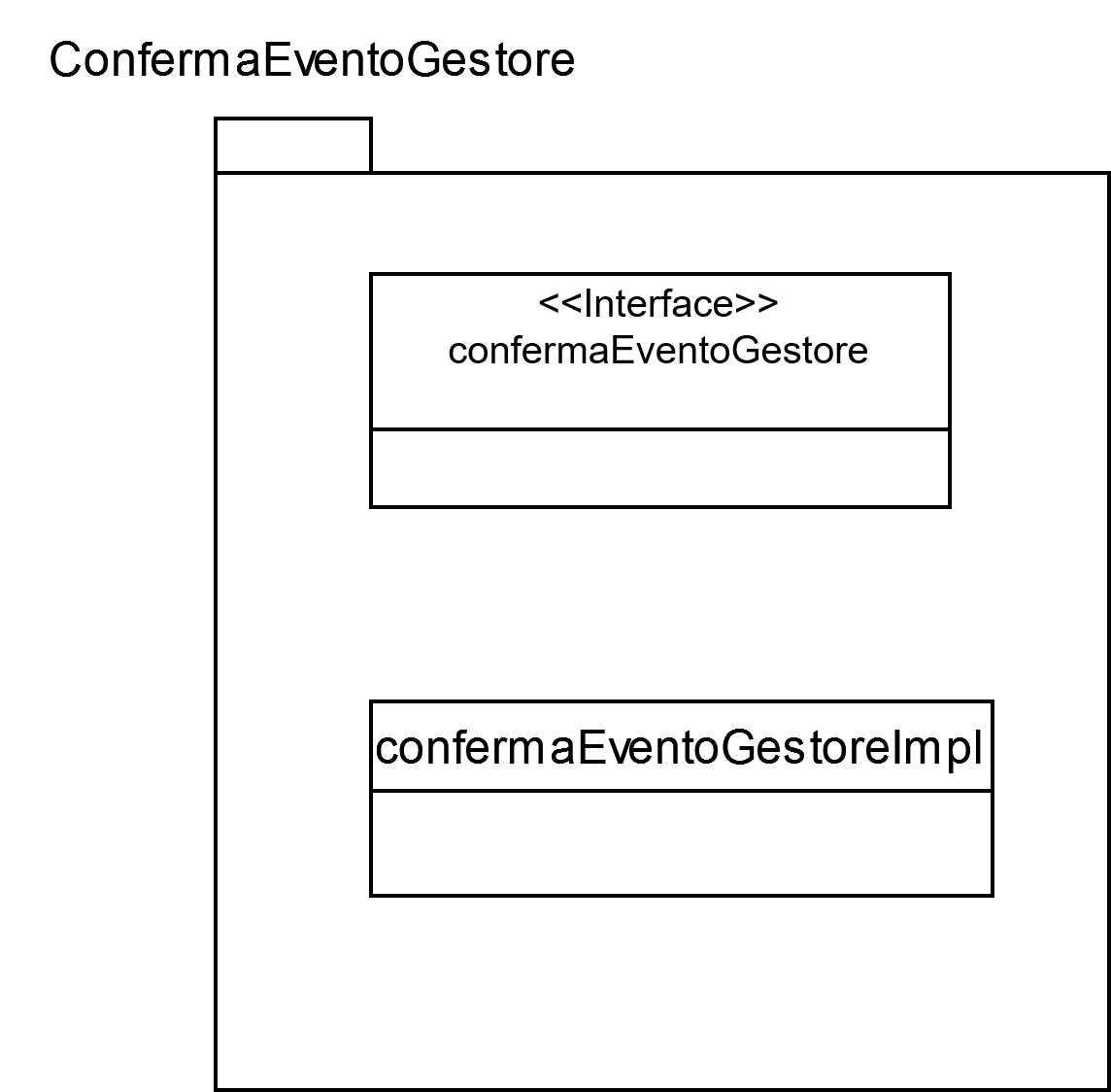
**Package gestioneEventi**



**Package persistenza**



**Package confermaEventoGestore**



# 3. Class Interfaces

Di seguito una lista dei metodi presentati dall’interfaccia di ogni package, alcuni metodi “di supporto” vengono omessi per ragioni di leggibilità.

**Package Autenticazione**

|  |  |
| --- | --- |
| **Nome classe** | **autenticazioneService** |
| **Descrizione** | **Permette la verifica delle credenziali in fase di login o di registrazione.** |
| **Metodi** | * **doCheckUsernameAlredyUsed(Utente username);** * **doCheckPassword(String username, String password);** * **doCheckUsernameManagerAlredyUsed(Gestore username);** * **doCheckManagerPassword(String username, String password);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doCheckUsernameAlredyUsed(Utente username)** |
| **Descrizione** | **Permette di verificare che l’username scelto non sia già occupato.** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **L’username viene registrato ed occupato** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doCheckPassword(String username, String password)** |
| **Descrizione** | **Permette di verificare che la password sia quella corrispondente all’username indicato.** |
| **Pre-condizione** | **L’utente deve essere registrato** |
| **Post-condizione** | **L’utente effettua il login** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doCheckUsernameManagerAlredyUsed(Gestore username)** |
| **Descrizione** | **Permette di verificare che l’username scelto da un gestore non sia già occupato.** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **L’username viene registrato ed occupato** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doCheckManagerPassword(String username, String password)** |
| **Descrizione** | **Permette di verificare che la password sia quella corrispondente all’username gestore indicato.** |
| **Pre-condizione** | **Il gestore deve essere registrato** |
| **Post-condizione** | **Il gestore effettua il login** |

**Package confermaEventoGestore**

|  |  |
| --- | --- |
| **Nome classe** | **confermaEventoGestore** |
| **Descrizione** | **Permette al gestore di visualizzare e di conseguenza accettare oppure rifiutare una prenotazione.** |
| **Metodi** | * **doRetriveSport(String name);** * **doRetriveCampo(String nome);** * **retriveEventoAttesaByName(String titolo);** * **doFetchWaitingMatch(String id);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveSport(String name);** |
| **Descrizione** | **Permette di prelevare uno Sport dal DB cercandolo per nome.** |
| **Pre-condizione** | **Lo sport deve essere precedentemente stato inserito nel DB** |
| **Post-condizione** | **Il metodo ritorno l’oggetto Sport, se trovato** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveCampo(String nome);** |
| **Descrizione** | **Permette di prelevare un campo dal DB cercandolo per nome.** |
| **Pre-condizione** | **Il campo deve essere precedentemente stato inserito** |
| **Post-condizione** | **Il metodo ritorno l’oggetto Campo, se trovato** |

|  |  |
| --- | --- |
| **Nome metodo** | * **retriveEventoAttesaByName(String titolo);** |
| **Descrizione** | **Permette di prelevare un evento tra quelli in attesa cercandolo per nome.** |
| **Pre-condizione** | **Deve essere stata effettuata la prenotazione** |
| **Post-condizione** | **Il metodo ritorno l’oggetto Evento, se trovato** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doFetchWaitingMatch(String id);** |
| **Descrizione** | **Permette di prelevare la lista di tutti gli eventi in attesa per un dato gestore** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **Il metodo ritorno l’oggetto List<Evento>, se trovato** |

**Package gestioneDatiGestore**

|  |  |
| --- | --- |
| **Nome classe** | **gestioneGestoreService** |
| **Descrizione** | **Permette al gestore di modificare i propri dati personali e quelli dei campi da lui gestiti** |
| **Metodi** | * **doChangePassword(String password, String id);** * **doChangeEmail(String email, String id);** * **doChangeNome(String nome, String id);** * **doChangeCognome(String cognome, String id);** * **doChangeProvincia(String provincia, String id);** * **doChangeCitta(String citta, String id);** * **doChangeVia(String via, String id);** * **doChangeTelefono(String telefono, String id);** * **doChangeIban(String Iban, String id);** * **doFetchCampiById(String id);** * **doModificaCampo(String idC, String nome\_c, int numero\_giocatori, String provincia, String citta, String via, double costo, double lunghezza, double larghezza);** * **doAddCampo(Campo c);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doChangePassword(String password, String id);** |
| **Descrizione** | **Permette di modificare la password** |
| **Pre-condizione** | **Il gestore deve essere registrato** |
| **Post-condizione** | **La password viene cambiata** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doChangeEmail(String email, String id);** |
| **Descrizione** | **Permette di modificare l’email** |
| **Pre-condizione** | **Il gestore deve essere registrato** |
| **Post-condizione** | **L’email viene cambiata** |

**Gli altri metodi “change” vengono omessi per motivi di leggibilità, il funzionamento è uguale per tutti.**

|  |  |
| --- | --- |
| **Nome metodo** | * **doFetchCampiById(String id);** |
| **Descrizione** | **Permette di cercare tutti i campi gestiti da un gestore** |
| **Pre-condizione** | **Il gestore deve possedere almeno un campo** |
| **Post-condizione** | **Ritorna la lista dei campi appartenenti ad un gestore** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doModificaCampo(String idC, String nome\_c, int numero\_giocatori, String provincia, String citta, String via, double costo, double lunghezza, double larghezza);** |
| **Descrizione** | **Permette di modificare i dati di un campo** |
| **Pre-condizione** | **Il gestore deve possedere almeno un campo** |
| **Post-condizione** | **Le proprietà del campo vengono modificate** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddCampo(Campo c);** |
| **Descrizione** | **Permette di aggiungere un campo** |
| **Pre-condizione** | **Il gestore deve essere registrato** |
| **Post-condizione** | **Il campo viene registrato** |

**Package gestioneDatiUtente**

|  |  |
| --- | --- |
| **Nome classe** | **gestioneUtenteService** |
| **Descrizione** | **Permette all’utente di modificare i propri dati personali** |
| **Metodi** | * **doChangePassword(String password, String id);** * **doChangeEmail(String email, String id);** * **doChangeProvincia(String provincia, String id);** * **doChangeCitta(String citta, String id);** * **doChangeVia(String via, String id);** * **doChangeTelefono(String telefono, String id);** * **doChangeIban(String Iban, String id);** * **doChangeSport(String sport, String id);** * **doAddCampo(Campo c);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doChangePassword(String password, String id);** |
| **Descrizione** | **Permette di modificare la password** |
| **Pre-condizione** | **L’utente deve essere registrato** |
| **Post-condizione** | **La password viene cambiata** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doChangeEmail(String email, String id);** |
| **Descrizione** | **Permette di modificare l’email** |
| **Pre-condizione** | **L’utente deve essere registrato** |
| **Post-condizione** | **L’email viene cambiata** |

**Gli altri metodi “change” vengono omessi per motivi di leggibilità, il funzionamento è uguale per tutti.**

**Package gestioneEventi**

|  |  |
| --- | --- |
| **Nome classe** | **gestioneEventiService** |
| **Descrizione** | **Permette tutta la gestione degli eventi** |
| **Metodi** | * **doAddUserToEvent(Utente u, Evento e);** * **doRetriveSportsName();** * **doRetriveAllMatch();** * **isPartecipating(String id, String nomeE);** * **doFetchUserMatch(String id);** * **retriveEventoByName(String nomeEvento);** * **doRetriveBySearch(Date date, String provincia);** * **doRetriveBySearch(String provincia);** * **doDropEventoUtente(String id, String nomeE);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddUserToEvent(Utente u, Evento e);** |
| **Descrizione** | **Permette la partecipazione di un utente ad un certo evento** |
| **Pre-condizione** | **L’utente deve essere registrato** |
| **Post-condizione** | **L’utente potrà partecipare all’evento** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveSportsName();** |
| **Descrizione** | **Preleva la lista di tutti gli sport disponibili** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **Restituisce la lista di tutti gli sport disponibili** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveAllMatch();** |
| **Descrizione** | **Preleva la lista di tutti gli eventi esistenti** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **Restituisce la lista con tutti gli eventi esistenti** |

|  |  |
| --- | --- |
| **Nome metodo** | * **isPartecipating(String id, String nomeE);** |
| **Descrizione** | **Permette di verificare se l’utente partecipa già all’evento indicato** |
| **Pre-condizione** | **L’utente deve essere registrato** |
| **Post-condizione** | **true se già partecipa, false altrimenti** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doFetchUserMatch(String id);** |
| **Descrizione** | **Preleva la lista di tutti gli eventi ai quali un utente si è iscritto** |
| **Pre-condizione** | **L’utente deve essere iscritto ad almeno un evento** |
| **Post-condizione** | **Restituisce la lista con tutti gli eventi appartenente all’utente** |

|  |  |
| --- | --- |
| **Nome metodo** | * **retriveEventoByName(String nomeEvento);** |
| **Descrizione** | **Permette di verificare se esiste un evento con il nome indicato** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **restituisce l’evento se esiste** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveBySearch(Date date, String provincia);** |
| **Descrizione** | **Permette di cercare eventi per data e provincia** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **restituisce una lista di eventi** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doRetriveBySearch(String provincia);** |
| **Descrizione** | **Permette di cercare eventi per data** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **restituisce una lista di eventi** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doDropEventoUtente(String id, String nomeE);** |
| **Descrizione** | **Permette ad un utente di cancellare la propria iscrizione ad un evento** |
| **Pre-condizione** | **L’utente deve essere iscritto all’evento** |
| **Post-condizione** | **L’utente viene eliminato dalla lista dei partecipanti** |

**Package persistenza**

|  |  |
| --- | --- |
| **Nome classe** | **persistenzaService** |
| **Descrizione** | **Permette la persistenza dei dati su DB** |
| **Metodi** | * **doAddEvento(Evento e);** * **doAddSport(String s);** * **doDropEvento(String nome);** * **doAddEventoAttesa(Evento e);** * **doDropGestore(String nome);** * **doDropUser(String nome);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddEvento(Evento e);** |
| **Descrizione** | **Permette di aggiungere un evento** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **L’evento viene aggiunto al DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddSport(String s);** |
| **Descrizione** | **Permette di aggiungere uno Sport** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **lo Sport viene aggiunto al DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doDropEvento(String nome);** |
| **Descrizione** | **Permette di rimuovere un certo evento** |
| **Pre-condizione** | **L’evento deve esistere** |
| **Post-condizione** | **L’evento viene rimosso dal DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddEventoAttesa(Evento e);** |
| **Descrizione** | **Permette di aggiungere una prenotazione** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **La prenotazione viene aggiunta al DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doDropGestore(String nome);** |
| **Descrizione** | **Permette ad un gestore di eliminare il proprio account** |
| **Pre-condizione** | **Il gestore deve esistere nel DB** |
| **Post-condizione** | **Il gestore viene rimosso dal DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doDropUser(String nome);** |
| **Descrizione** | **Permette ad un utente di eliminare il proprio account** |
| **Pre-condizione** | **l’utente deve esistere nel DB** |
| **Post-condizione** | **L’utente viene rimosso dal DB** |

**Package registrazione**

|  |  |
| --- | --- |
| **Nome classe** | **registrazioneService** |
| **Descrizione** | **Permette ad utenti e gestori di registrarsi al servizio** |
| **Metodi** | * **doAddUser(Utente utente);** * **doAddManager(Gestore gestore);** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddUser(Utente utente);** |
| **Descrizione** | **Permette ad un utente di creare il proprio account** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **L’utente viene aggiunto al DB** |

|  |  |
| --- | --- |
| **Nome metodo** | * **doAddManager(Gestore gestore);** |
| **Descrizione** | **Permette ad un gestore di creare il proprio account** |
| **Pre-condizione** | **Na** |
| **Post-condizione** | **Il gestore viene aggiunto al DB** |

# 4. Class Diagram

**Vedere SDD**

# 5. Design Patterns

Il sistema è completamente costruito con un Three Tier Design Pattern. Questo ci permette di separare la logica di controllo dalle entità e dal livello di presentazione, in modo da poter lavorare agevolmente su ogni sottosistema.