

Università degli Studi di Salerno
Corso di Ingegneria del Software

**My Fitness Plan
Object Design Document
Versione 2.0**



Data: 14/01/2018

Progetto: MyFitnessPlan	Versione: 2.0
Documento: ObjectDesignDocument	Data: 14/01/2018

Partecipanti:

Nome	Matricola
Andrea Montefusco	0512102336
Marino Iannacchero	0512103434
Simona Santoro	0512103464

Revision History

Data	Versione	Descrizione	Autore
30/12/2017	1.0	Definizione delle linee guida per l'implementazione, documentazione classi.	Andrea Montefusco
30/12/2017	1.0	Descrizione della divisione in packages del sistema.	Simona Santoro
30/12/2017	1.0	Descrizione delle interfacce delle classi.	Marino Iannacchero
5/1/2018	2.0	Modifica delle linee guida per l'implementazione, modifica documentazione classi.	Andrea Montefusco
5/1/2018	2.0	Modifica descrizione delle divisione in packages del sistema.	Simona Santoro
5/1/2018	2.0	Modifica descrizione delle interfacce delle classi.	Marino Iannacchero

Indice

1.LINEE GUIDA PER L'IMPLEMENTAZIONE

- 1.1 Convenzioni per file .html/.jsp
- 1.2 Convenzioni per i file .css
- 1.3 Convenzioni per i file .java
 - 1.3.1 Istruzione if-else
 - 1.3.2 Istruzione for
 - 1.3.3 Istruzione while e do- while
 - 1.3.4 Istruzione switch
- 1.4 Convenzioni per i file .js
 - 1.4.1 Convenzioni per gli oggetti

2. DESIGN PATTERN.

3. PACKAGES.

3.1 Package core.

 3.1.1 Package bean.

 3.1.2 Package control.

 3.1.2.1 Package autenticazione.

 3.1.2.2 Package registrazione.

 3.1.2.3 Package Account Cliente.

 3.1.2.4 Package Account Gestore.

 3.1.3 Package exception.

 3.1.4 Package model.

 3.1.5 Package test.

 3.1.6 Package View.

 3.1.6.1 Package Account Gestore.

 3.1.6.2 Package Account Cliente.

 3.1.2.3 Package Autenticazione.

 3.1.2.4 Package Registrazione.

 3.1.2.5 Package Esito.

1. Linee guida per l'implementazione.

Convenzioni globali

- La lunghezza delle righe di codice deve essere non superiore a 80 caratteri.
- I blocchi di codice deve essere indentati utilizzando 4 spazi vuoti.

1.1 Convenzioni per file .html/.jsp

- I nomi dei file devono essere lettera minuscola.
- I nomi degli elementi devono essere lettera minuscola.
- Tutti gli elementi devono essere chiusi (anche gli elementi per i quali non è necessario).
- I nomi degli attributi devono essere lettera minuscola.
- I valori degli attributi devono iniziare e terminare con i doppi apici.
- Le immagini dovranno avere l'attributo “alt” che specifica, in maniera testuale, il contenuto dell'immagine.
- I blocchi di codice lunghi dovranno essere separati da una riga vuota.
- Indentare il codice utilizzando un solo spazio e solo quando necessario, eventualmente spazi per separare blocchi.
- Non omettere il tag `<head>...</head>`.
- Includere il seguente tag `<meta name="viewport" content="width=device-width, initial-scale=1.0">` in tutte le pagine.

1.2 Convenzioni per i file .css

- I nomi dei file devono essere lettera minuscola.
- Aprire la parentesi dopo il selettore lasciando uno spazio vuoto.
- Inserire gli attributi partendo dalla riga successiva a quella del selettore.
- Utilizzare i doppi apici solo per i valori degli attributi che hanno spazi vuoti.
- Dopo ogni coppia proprietà-attributo inserire un punto e virgola.
- Chiudere le parentesi sulla riga successiva all'ultima riga con proprietà-attributo, senza spazi di indentazione.

1.3 Convenzioni per i file .java

- Tendenzialmente utilizzare per i nomi delle classi dei sostantivi.
- Utilizzare (quando è possibile) aggettivi per i nomi delle interfacce e il suffisso “-able”.
- I nomi dei file devono essere lettera maiuscola.

- I nomi delle servlet devono contenere il suffisso Servlet.
- I nomi di variabili e metodi devono iniziare con una lettera minuscola, le parole successive devono iniziare lettera maiuscola.
- Dichiare ogni variabile su una riga separata, variabili dello stesso tipo devono trovarsi su righe successive.
- Non utilizzare caratteri speciali in variabili e metodi.
- Per variabili costanti o per proprietà statiche, è possibile utilizzare il carattere underscore.
- Per accedere alle variabili di istanza usare metodi del tipo getNomeVariabile(), setNomeVariabile().
- Ogni classe deve contenere all'inizio un commento indicante: Nome della classe, descrizione, autore, versione, utilizzare gli opportuni metodi.
- Per i costruttori e i metodi inserire un commento che li descrive, indicare l'autore, specificare i parametri che intervengono e per i metodi descrivere i valori di ritorno, utilizzare gli opportuni tag.

1.3.1 Istruzione if-else

Deve essere scritta con questa forma:

```
if(Boolean_expression 1) {
    // Executes when the Boolean expression 1 is true
}else if(Boolean_expression 2) {
    // Executes when the Boolean expression 2 is true
}else if(Boolean_expression 3) {
    // Executes when the Boolean expression 3 is true
}else {
    // Executes when the none of the above condition is true.
}
```

-Utilizzare il costrutto if-else in forma abbreviata quando è possibile.

1.3.2 Istruzione for

Deve essere scritta con questa forma:

```
for(initialization; condition ; increment/decrement) {
```

```
    statement(s);  
}
```

1.3.3 Istruzione while e do-while

Devono avere questa forma:

```
while(Boolean_expression) {  
    statements(s);  
}
```

```
do {  
    statements(s);  
} while(Boolean_expression);
```

1.3.4 Istruzione switch

Deve essere scritta con questa forma:

```
switch(expression) {  
    case value :  
        // Statements  
        break; // optional  
  
    case value :  
        // Statements  
        break; // optional  
  
    // You can have any number of case statements.  
    default : // Optional  
        // Statements  
}
```

1.3.5 Istruzione try-catch e try-catch-finally.

Devono avere la seguente forma:

```

try {
    statement(s)
} catch(Exception e) {
    statement(s)
}

try {
    statement(s)
} catch(Exception e) {
    statement(s)
} finally {
    statement(s)
}

```

1.4 Convenzioni per i file .js

- I nomi di variabili e metodi devono iniziare con una lettera minuscola, le parole successive devono iniziare lettera maiuscola.
- Dichiarare ogni variabile su una riga separata.
- Non utilizzare caratteri speciali in variabili e metodi.
- Variabili globali e costanti devono essere scritte lettera grande.
- Terminare ogni dichiarazione con un punto e virgola.
- Per i vari costrutti seguire le convenzioni per i file .java

1.4.1 Convenzioni per gli oggetti

Devono essere definiti in questa maniera

```

var person = {
    firstName: "John",
    lastName: "Doe",
    age: 50,
    eyeColor: "blue"
};

```

oppure così in caso di oggetti con poche proprietà e metodi

```
var person = {firstName:"John", lastName:"Doe", age:50};
```

2.Design pattern.

Singleton Design Pattern: Design pattern che permette di creare una sola istanza di una specifica classe.

Facade Design Pattern: Design pattern che utilizza un oggetto che permette, attraverso un'interfaccia più semplice, l'accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro, nonché a blocchi di codice complessi.

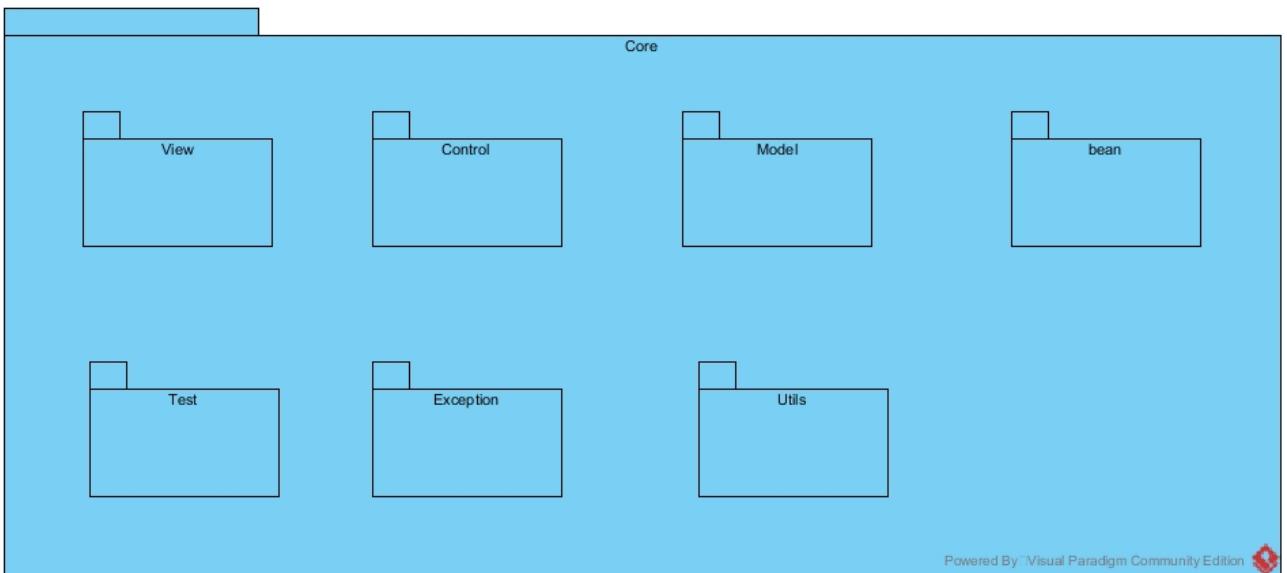
2. Packages

Il sistema segue un modello three-tier, cioè la gestione del nostro sistema viene suddiviso in tre livello:

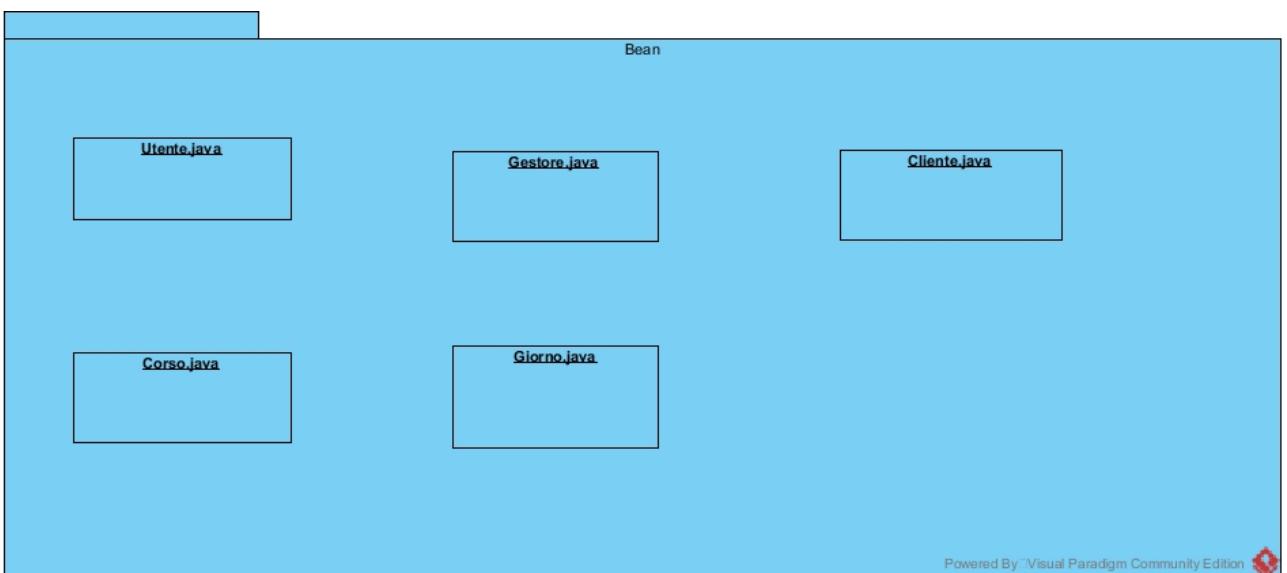
- Interface layer
- Application Logic Layer
- Storage Layer

INTERFACE LAYER	Questo livello rappresenta l'interfaccia con cui l'utente può interagire inviando dati in input e visualizzando dati in output.
APPLICATION LAYER	Questo livello ha il compito di elaborare i dati inviati dal client. Spesso necessita di richieste al database per accedere ai dati persistenti.
STORAGE LAYER	Questo livello si occupa di memorizzare i dati persistenti utilizzando un DBMS.

2.1 Package core

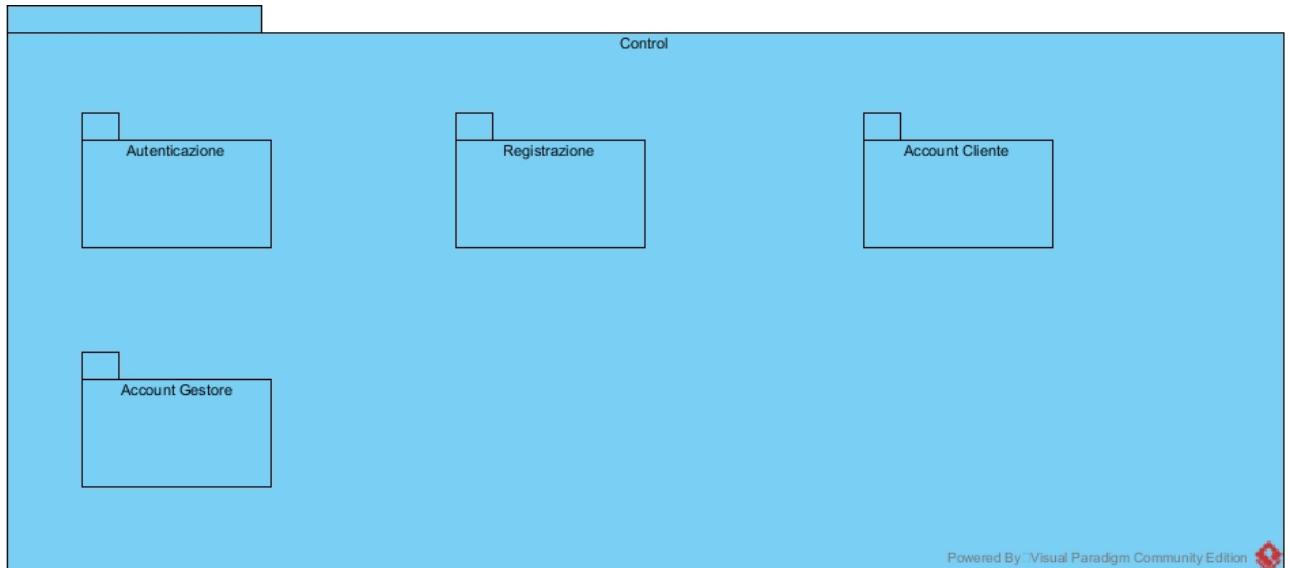


2.1.1 Package Bean

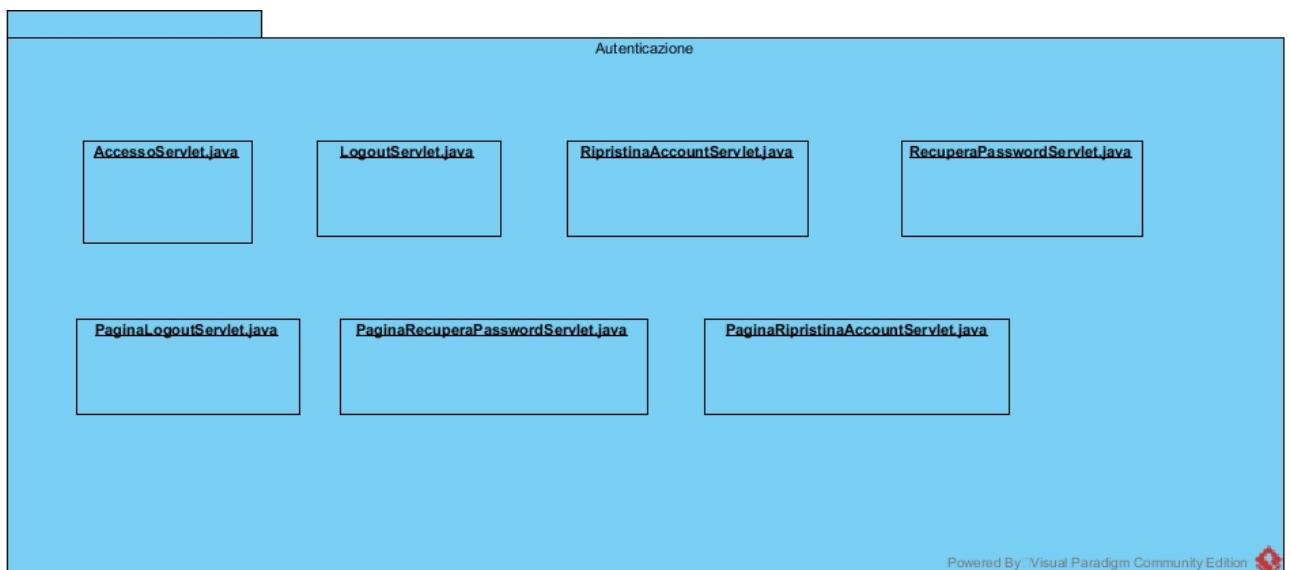


Classe	Descrizione
Utente	Describe l'utente generico che utilizza il sistema
Giorno	Describe i giorni in cui si terrà il corso
Corso	Describe un corso messo a disposizione dalla palestra.
Cliente	Describe il cliente che usufruisce dei servizi della palestra
Gestore	Describe il gestore(unico) della palestra

2.1.2 Package Control



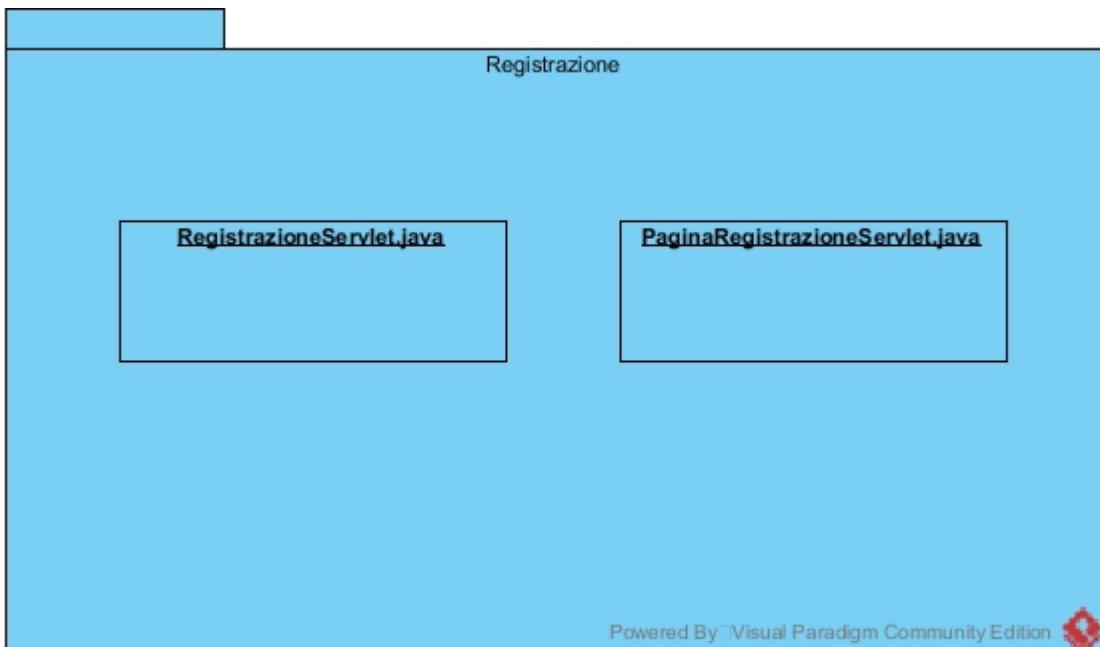
2.1.2.1 Package Autenticazione



AccessoServlet.java	Controller che riguarda l'accesso al sistema
LogoutServlet.java	Controller che riguarda l'uscita dell'utente dal sistema

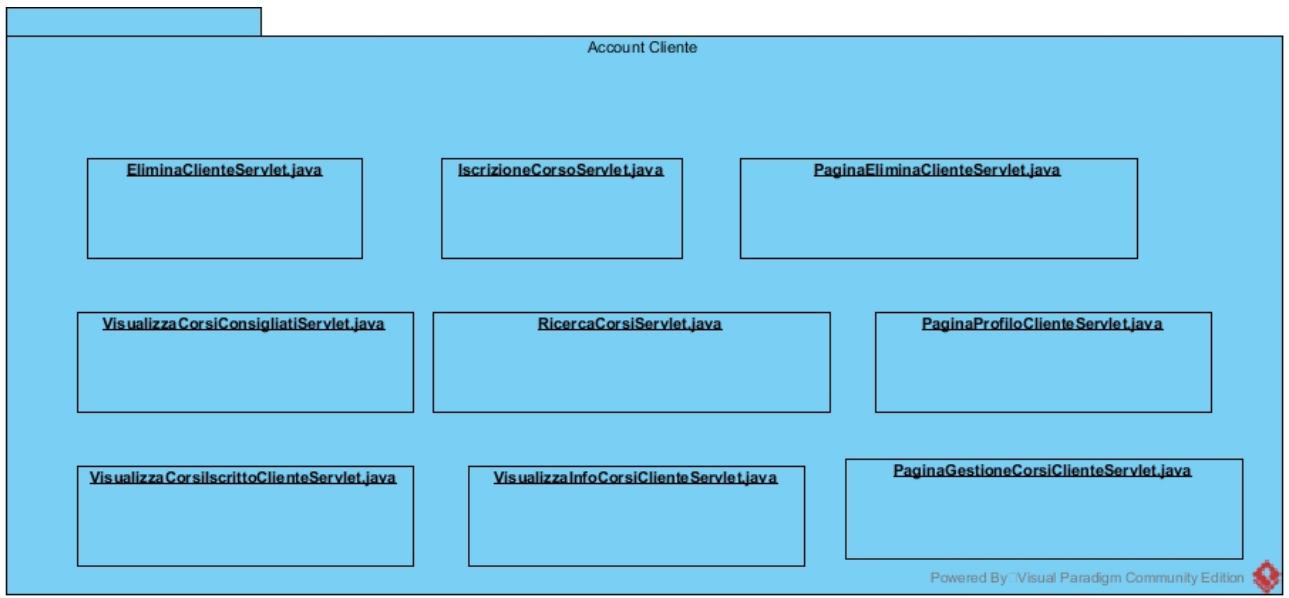
RipristinaAccountServlet.java	Controller che riguarda la riattivazione dell'account del Cliente dopo essersi eliminato dal sistema
RecuperaPasswordServlet.java	Controller che riguarda il recupero della password da parte del Cliente e del Gestore.
PaginaLogoutServlet.java	Controller che reindirizza alla pagina dell'uscita del sistema
PaginaRecuperaPasswordServlet.java	Controller che reindirizza alla pagina di recupero password
PaginaRipristinaAccountServlet.java	Controller che reindirizza alla pagina di ripristina account

2.1.2.2 Package Registrazione



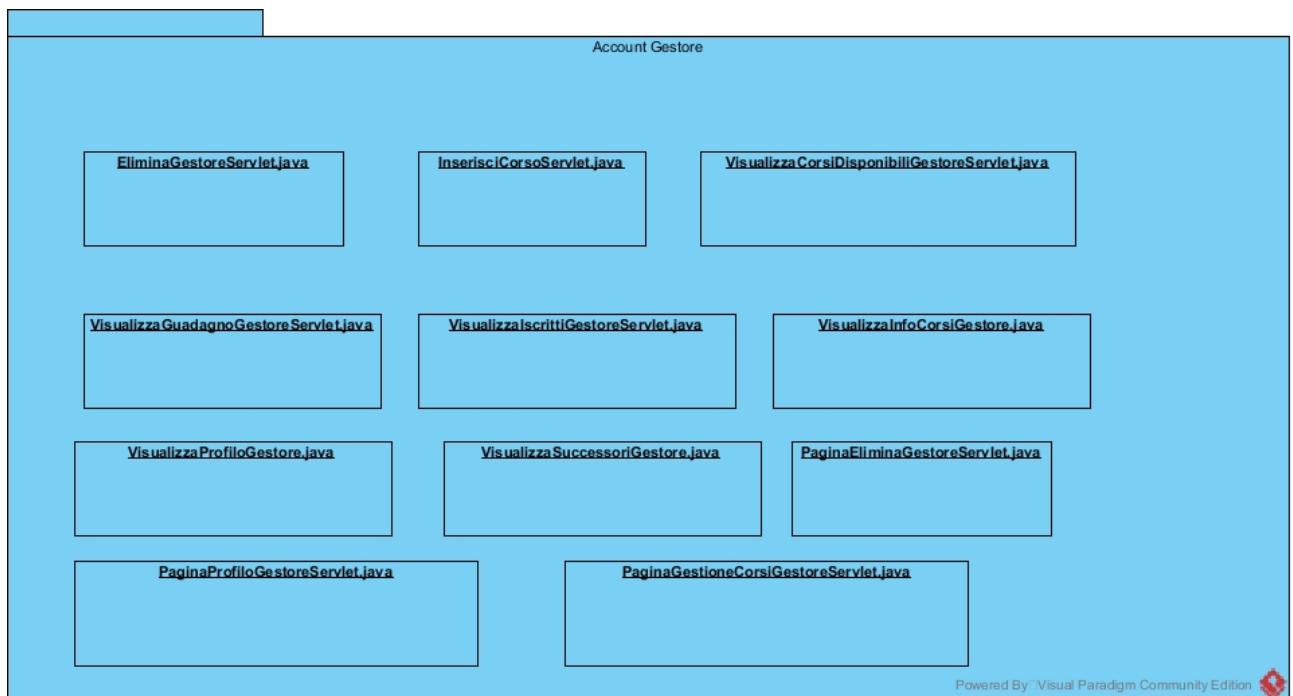
RegistrazioneServlet.java	Controller che riguarda la registrazione dell'utente al sistema.
PaginaRegistrazioneServlet.java	Controller che reindirizza alla pagina di registrazione.

2.1.2.3 Package Account Cliente



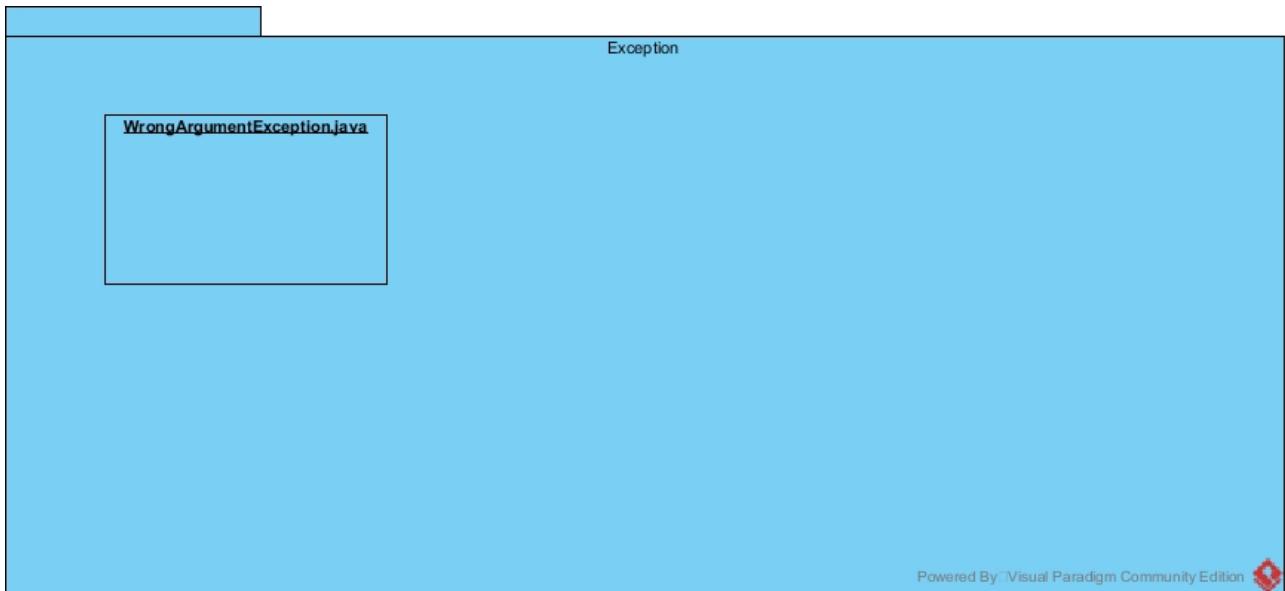
EliminaClienteServlet.java	Controller che permetta al Cliente di eliminare l'account
IscrizioneCorsoServlet.java	Controller che permette al Cliente di iscriversi a un corso messo a disposizione dalla palestra
RicercaCorsiServlet.java	Controller che consente al Cliente di ricercare un corso presente nel sistema
VisualizzaCorsiConsigliatiServlet.java	Controller che consente al Cliente di visualizzare i corsi che il sistema consiglia
VisualizzaCorsiIscrittoClienteServlet.java	Controller che consente al Cliente di visualizzare i corsi a cui è iscritto.
VisualizzaInfoCorsiCliente.java	Controller che consente al Cliente di visualizzare le informazioni del corso da lui scelto
PaginaProfiloClienteServlet.java	Controller che reindirizza alla pagina di profilo del Cliente
PaginaEliminaClienteServlet.java	Controller che reindirizza alla pagina di eliminazione account.
PaginaGestioneCorsiClienteServlet.java	Controller che reindirizza alla pagina di gestione dei corsi del Cliente.

2.1.2.4 Package Account Gestore



InserisciCorsoServlet.java	Controller che permette al Gestore della palestra di inserire un corso.
EliminaGestoreServlet.java	Controller che permette al Gestore della palestra di eliminare il proprio account
VisualizzaCorsiDisponibiliGestoreServlet.java	Controller che permette al Gestore di visualizzare i corsi disponibili, cioè attivi, nel sistema.
VisualizzaGuadagnoGestoreServlet.java	Controller che permette al Gestore della palestra di visualizzare il guadagno totale di un corso non più attivo nel sistema.
VisualizzascrittiGestoreServlet.java	Controller che consente al Gestore di visualizzare gli iscritti a un determinato corso
VisualizzaInfoCorsiGestore.java	Controller che consente al Gestore di visualizzare le informazioni di un determinato corso.
VisualizzaProfiloGestore.java	Controller che consente al Gestore di visualizzare i dati relativi al Gestore.
VisualizzaSuccessoriGestore.java	Controller che consente al Gestore di visualizzare i clienti del sistema.
PaginaEliminaGestoreServlet.java	Controller che reindirizza alla pagina di eliminazione account.
PaginaProfiloGestoreServlet.java	Controller che reindirizza alla pagina di profilo del gestore
PaginaGestioneCorsiGestore.java	Controller che reindirizza alla pagina di gestione dei corsi.

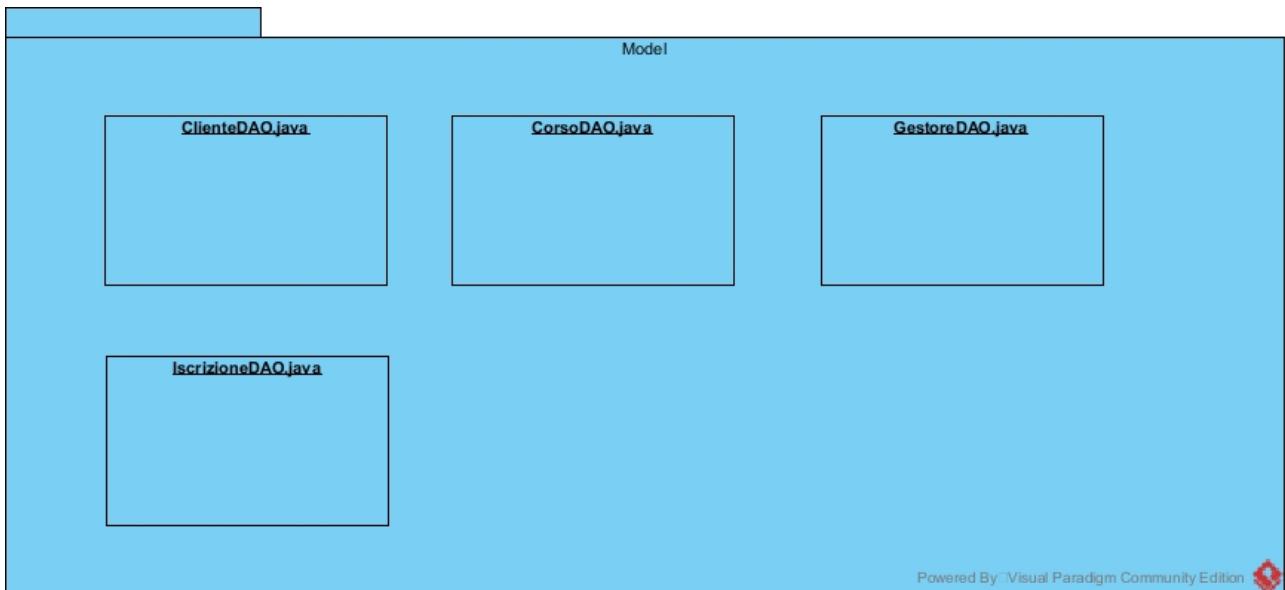
2.1.3 Package exception



WrongArgumentException.java

L'eccezione viene lanciata quando i parametri che vengono passati a un metodo non sono corretti.

2.1.4 Package model

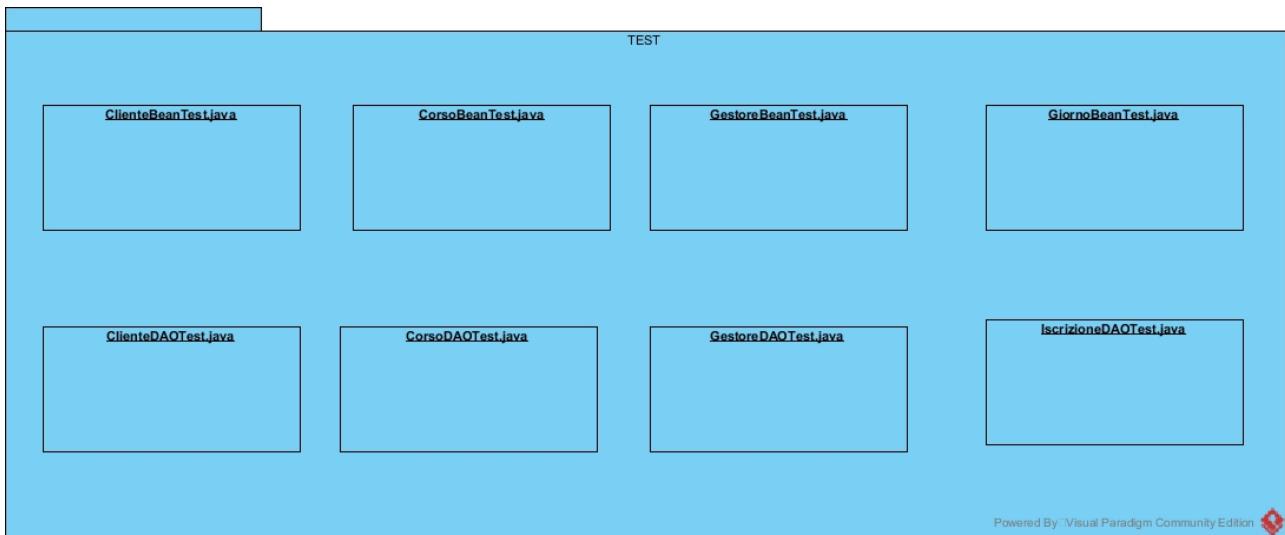


ClienteDAO.java

Questo model effettua le query riguardanti l'account del Cliente

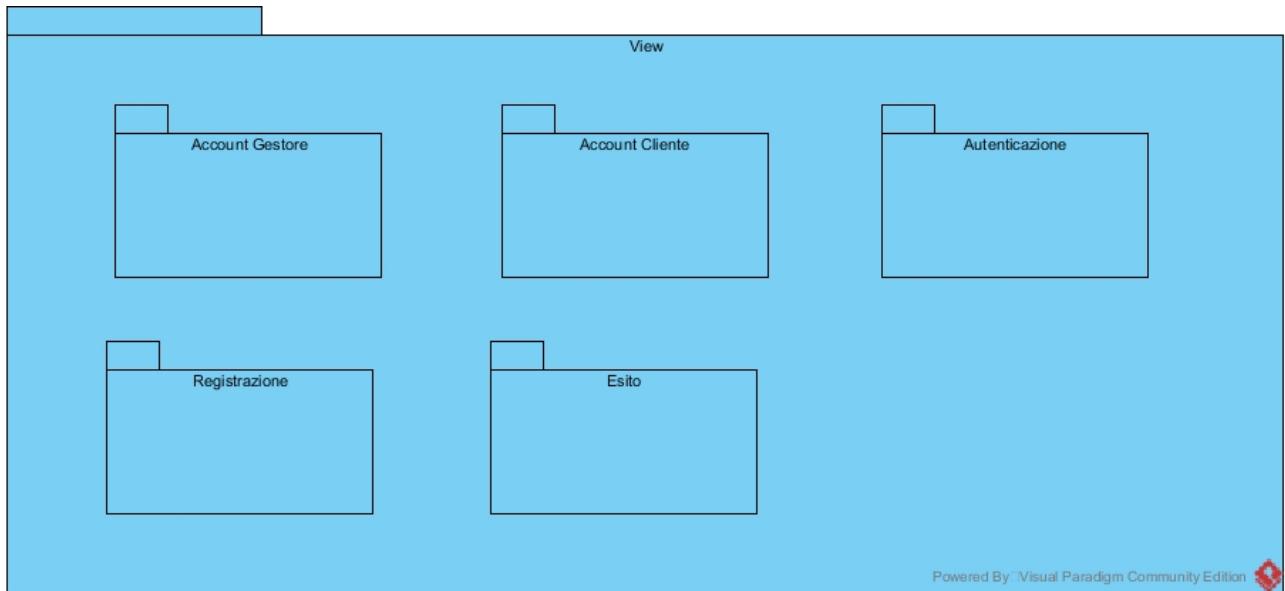
GestoreDAO.java	Model che effettua le query che riguardano la gestione dell'account del Gestore
CorsoDAO.java	Model che effettua tutte le query riguardante a un corso.
IscrizioneDAO.java	Model che effettua tutte le query riguardanti l'iscrizione di un cliente ad un corso.

2.1.5 Package Test

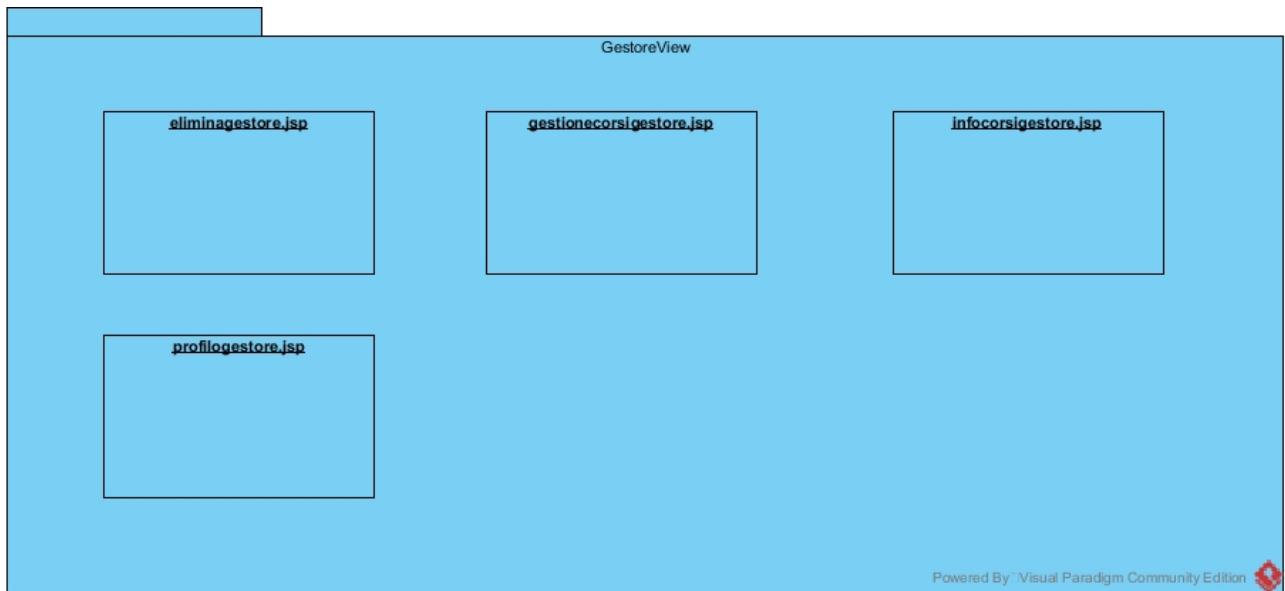


IscrizioneDAOTest.java	Classe che consente di testare i metodi del model Iscrizione
GestoreDAOTest.java	Classe che consente di testare i metodi del model Gestore
CorsoDAOTest.java	Classe che consente di testare i metodi del model Corso
ClienteDAOTest.java	Classe che consente di testare i metodi del model Cliente
ClienteBeanTest.java	Classe che consente di testare i metodi del bean Cliente
CorsoBeanTest.java	Classe che consente di testare i metodi del Bean Corso.
GestoreBeanTest.java	Classe che consente di testare i metodi del bean Gestore.
GiornoBeanTest.java	Classe che consente di testare i metodi del bean Giorno

2.1.6 Package View



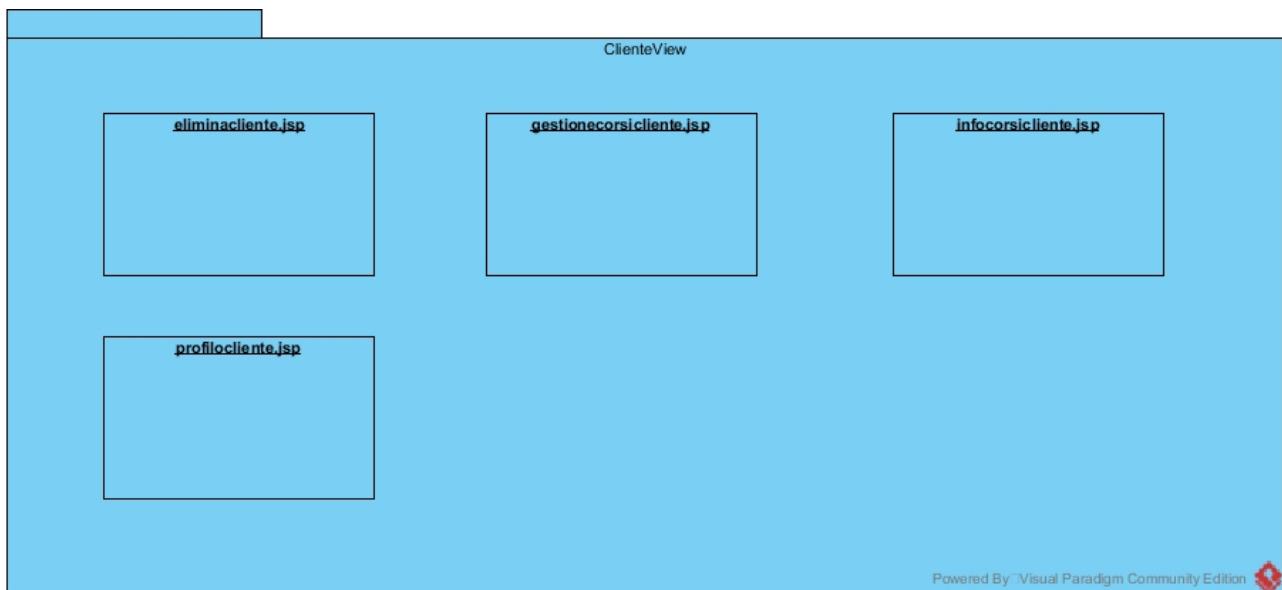
2.1.6.1 Package Account Gestore



profilogestore.jsp	La view che permette al Gestore di visualizzare il profilo dove verranno consultate le seguenti informazioni: 1. Dati Account 2. Storico Guadagno totale per un corso non più attivo
gestionecorsigestore.jsp	La view che permette al Gestore di Gestire i corsi che offre la palestra.

	<ol style="list-style-type: none"> 1. Inserire un Corso 2. Visualizzare i Corsi presenti nel sistema
paginacorso.jsp	<p>La view che permette di visualizzare le informazioni relative a un corso già inserito:</p> <ol style="list-style-type: none"> 1. Dati corso 2. Giorni e orari corso 3. Iscritti al corso
eliminagestore.jsp	<p>La view che permette al Gestore di scegliere il suo successore e confermare l'eliminazione dell'account.</p>

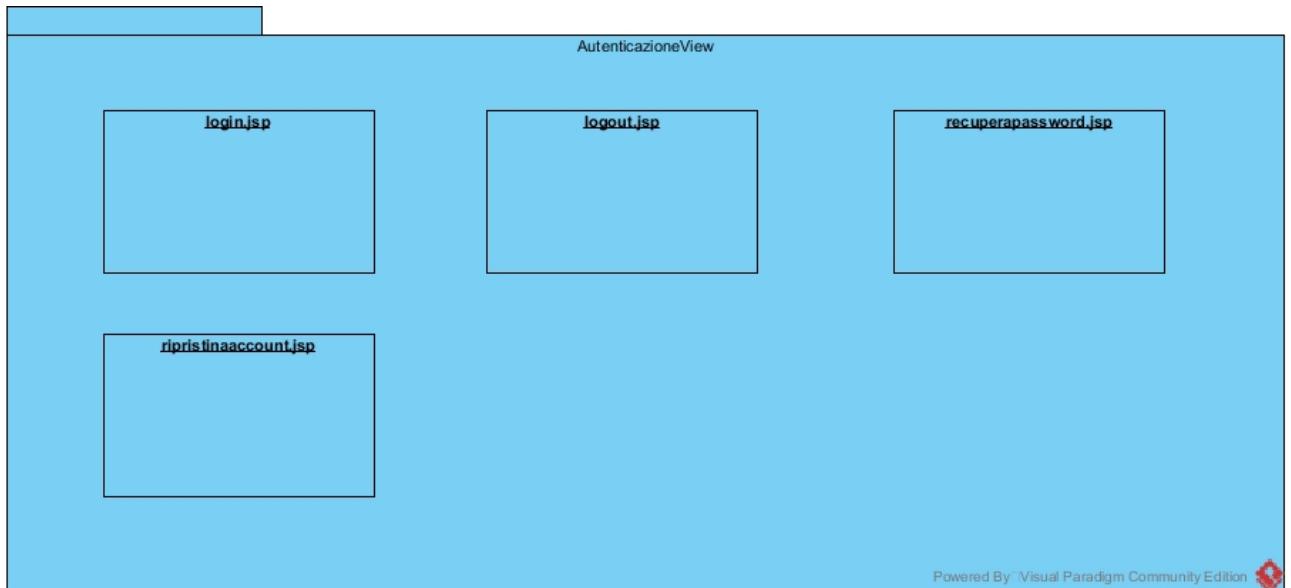
2.1.6.2 Package Cliente



gestionecorsicliente.jsp	<p>Questa View è la prima pagina che il Cliente della Palestra visualizzare dopo aver effettuato l'accesso.</p> <ol style="list-style-type: none"> 1. Visualizzare i Corsi Consigliati 2. Ricerca Un Corso
profilocliente.jsp	<p>Questa view permette al Cliente della Palestra di visualizzare informazioni relativa all'account.</p> <ol style="list-style-type: none"> 1. Visualizzare i suoi dati 2. Visualizzare i Corsi a cui è iscritto
infocorsicliente.jsp	<p>Questa view consente al Cliente della Palestra di visualizzare le informazioni del corso scelto ed eventualmente effettuare l'iscrizione.</p>
eliminacliente.jsp	<p>Questa view consente al cliente di</p>

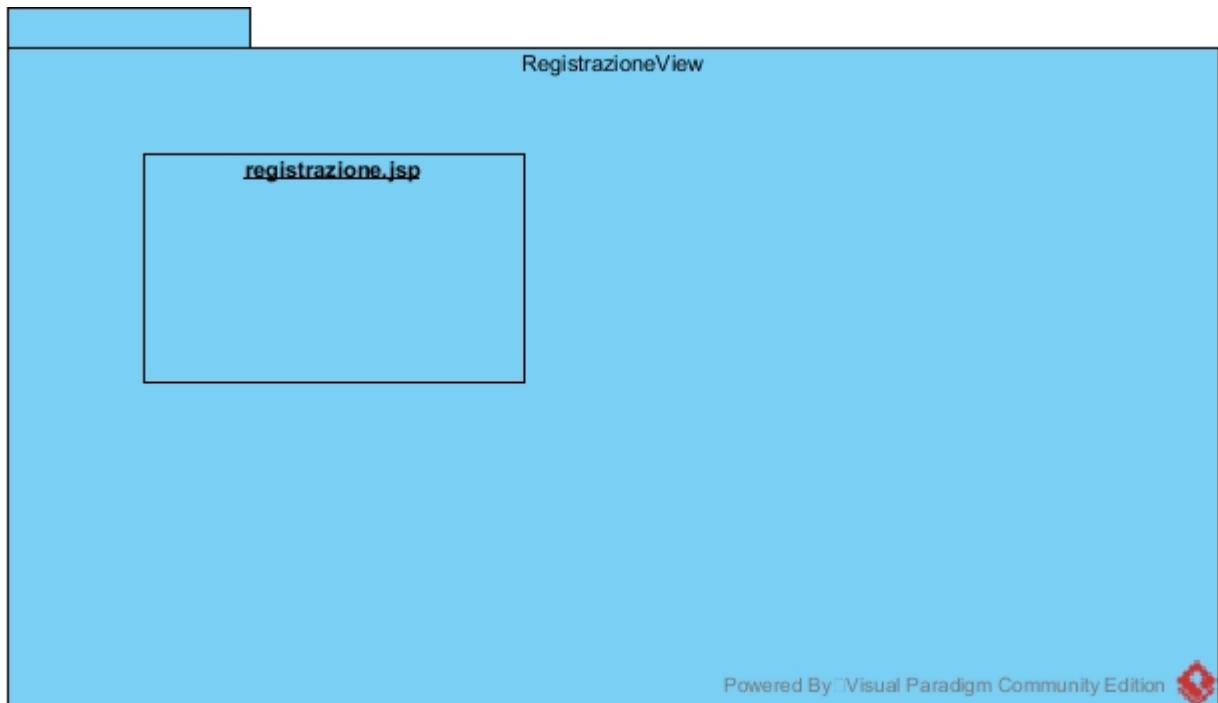
	eliminare il proprio account attraverso l'inserimento della password associata al suo account..
--	---

2.1.6.3 Package Autenticazione



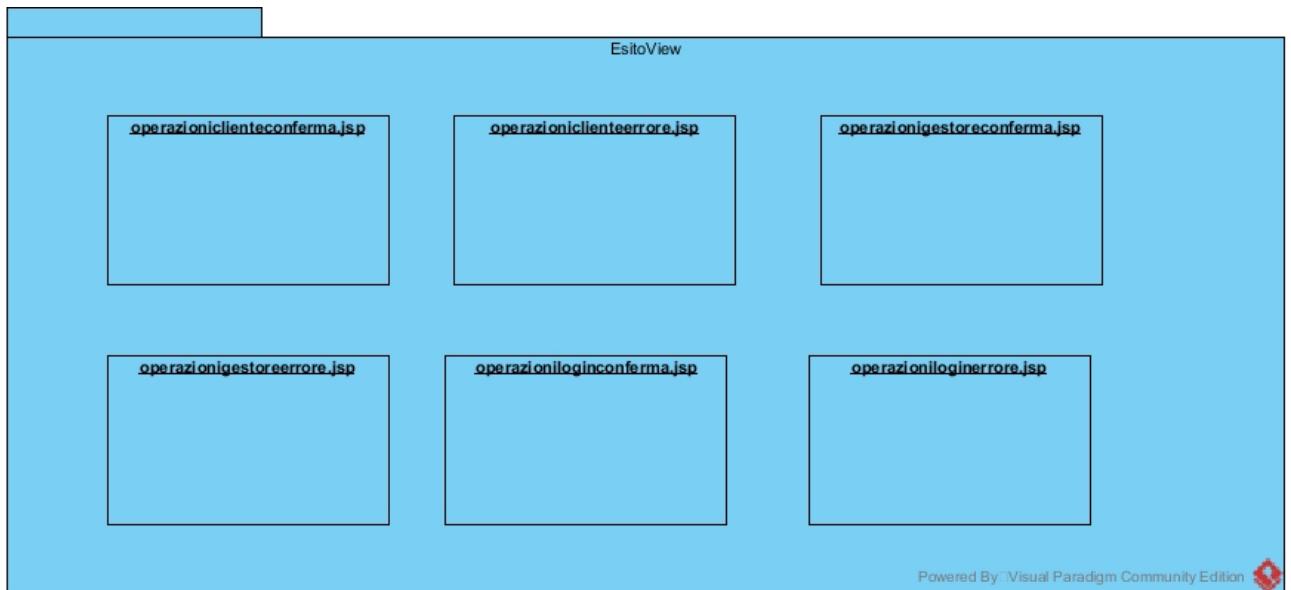
Login.jsp	View che consente all'utente di effettuare l'accesso al sistema
logout.jsp	View che consente all'utente di effettuare l'uscita dal sistema
recuperapassword.jsp	View che consente all'utente di recuperare la password attraverso l'inserimento dell'email.
ripristinaaccount.jsp	View che consente all'utente di riprisinare l'account attraverso l'inserimento dell'email e della password.

2.1.6.4 PACKAGE REGISTRAZIONE



registrazione.jsp	View che consente all'utente l'inserimento dei propri dati necessari al sistema.
-------------------	--

2.1.6.5 PACKAGE ESITO

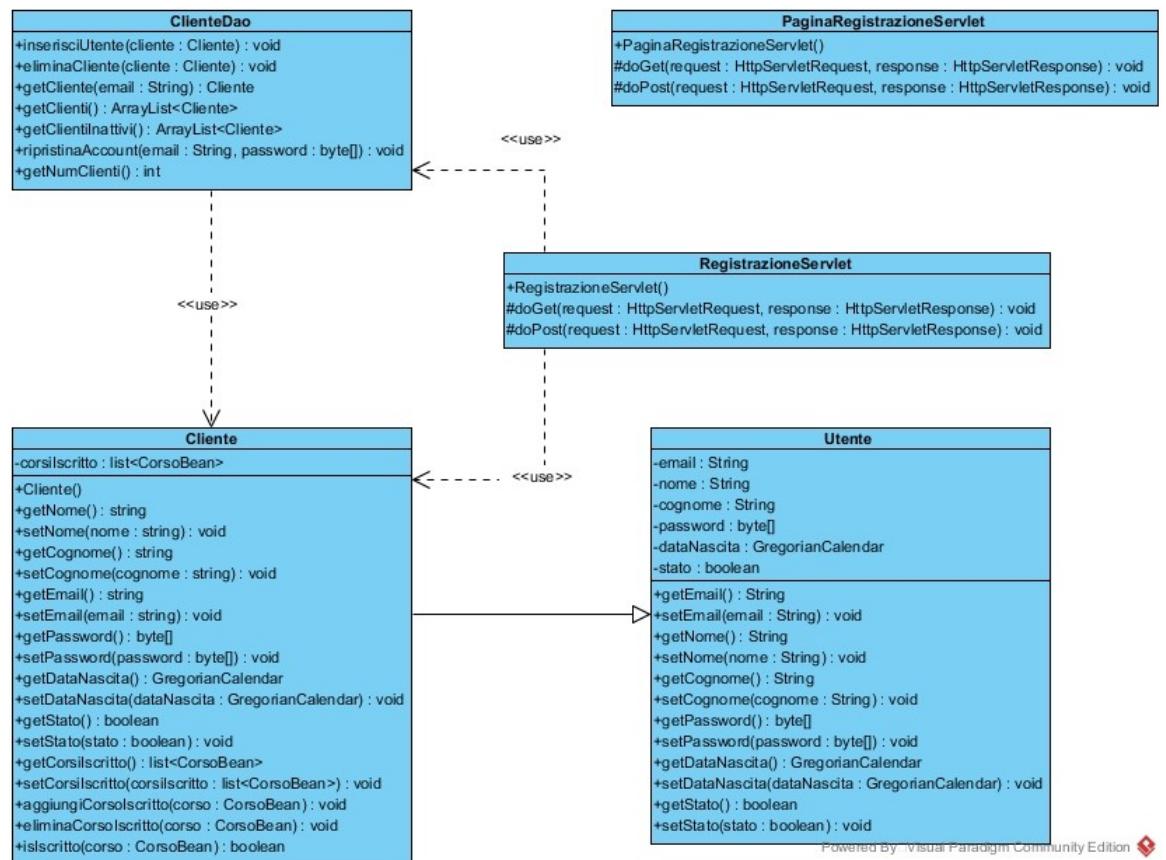


operacionclienteconferma.jsp	View che visualizza la conferma di un operazione effettuata dal cliente
operacionclienteerrore.jsp	View che visualizza l'errore di un operazione effettutata dal cliente
operaciongestoreconferma.jsp	View che visualizza la conferma di un operazione effettuata dal gestore
operaciongestoreerrore.jsp	View che visualizza l'errore di un operazione effettuata dal gestore
operacionloginconferma.jsp	View che visualizza la conferma di un operazione effettuata dall'utente
operacionloginerrore.jsp	View che visualizza l'errore di un operazione effettuata dall'utente

3.Class Interfaces.

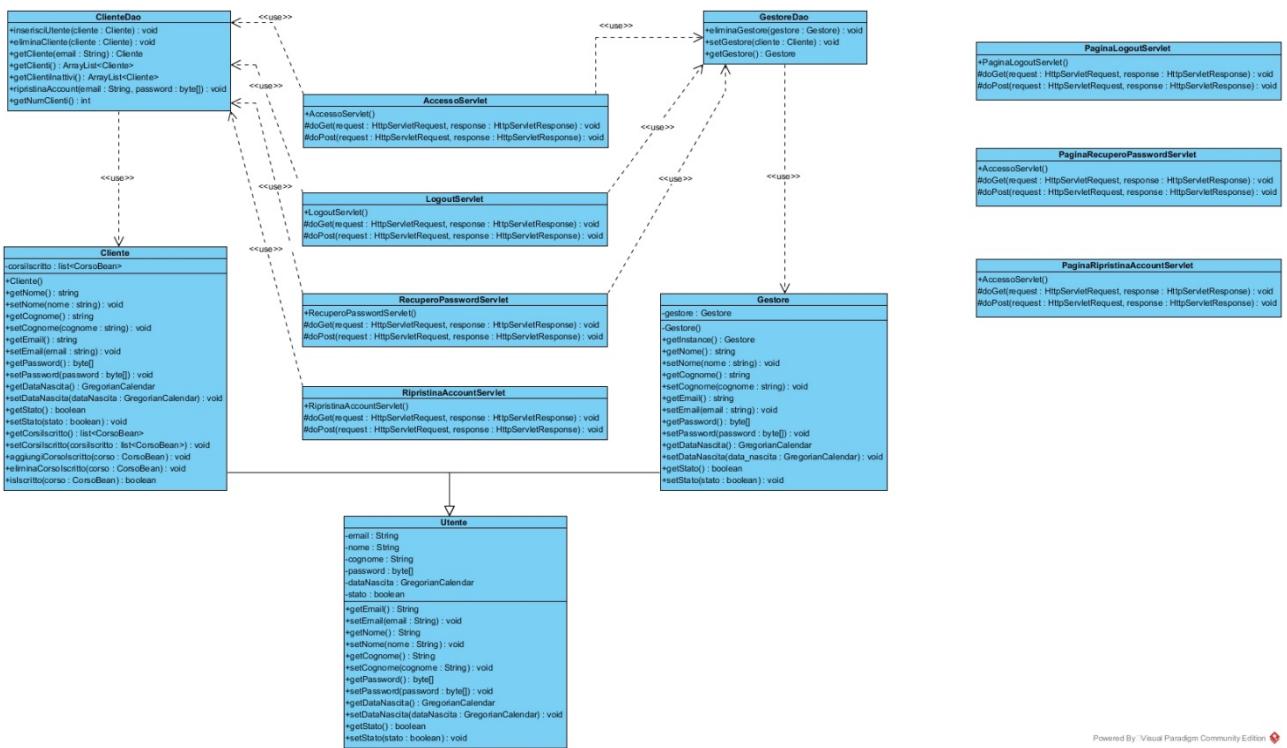
Consultare la i file javaDoc allegati per la documentazione delle interfacce.

3.1 Registrazione.

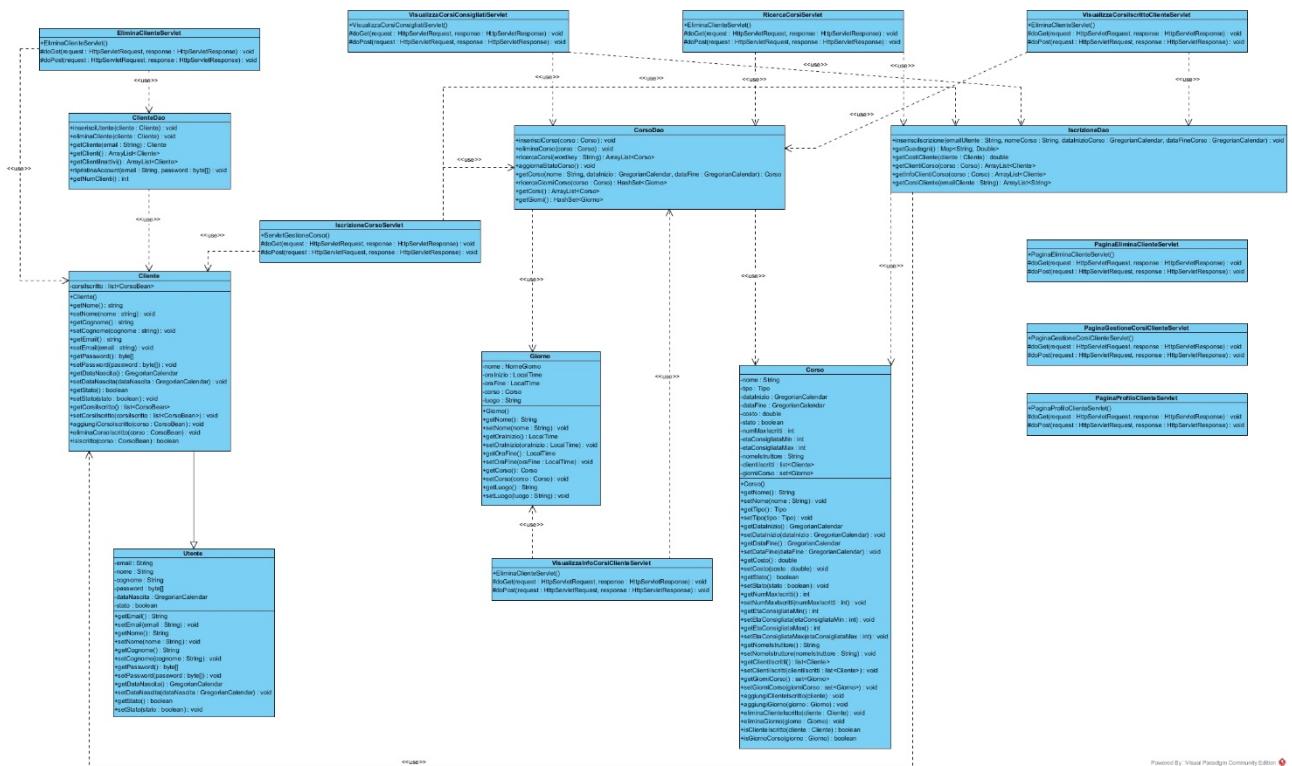


Powered By /Visual Paradigm Community Edition

3.2 Autenticazione.



3.3 Account Cliente



3.4 Account Gestore

