

| Object Design Document  Rojina Review   | Riferimento |  | | --- | --- | | Versione | 2.0 | | Data | 18/01/2023 | | Destinatario | Carmine Gravino, Francesco Casillo (tutor) | | Presentato da | Carmine Iemmino (0512109893), Andrea Vitolo (0512109776), Carmine Pio Nardo (0512110148) | | Approvato da |  | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

# 

# Team members

| **Nome** | **Ruolo** | **Acronimo** |
| --- | --- | --- |
| Carmine Iemmino | Team member | **CI** |
| Andrea Vitolo | Team member | **AV** |
| Carmine Pio Nardo | Team member | **CPN** |

# 

# Revision History

| Data | Versione | Descrizione | Autori |
| --- | --- | --- | --- |
| 28/12/2022 | 0.1 | Prima stesura | **CI, AV, CPN** |
| 07/01/2022 | 0.2 | Inserimento commenti interfacce service | **CI,AV,CPN** |
| 18/01/2022 | 2.0 | Revisione Finale | **CI,CPN,AV** |

**Sommario**

[**Team members**](#_heading=h.f3i0hzxcvcs) **3**

[**Revision History**](#_heading=h.xbbnxm2se52l) **4**

[**1 Introduzione**](#_heading=h.r8j18cpiwyx6) **6**

[1.1 Object design goals](#_heading=h.i63k025szp4q) 6

[1.2 Linee guida per la scrittura del codice](#_heading=h.kst86euf58pd) 7

[1.3 Definizioni, acronimi, e abbreviazioni](#_heading=h.z2lpu2pczw0q) 7

[1.4 Riferimenti](#_heading=h.ct6g1lxf41kf) 7

[**2 Packages**](#_heading=h.l5g58x4mzdye) **8**

[2.1 Package Rojina Review](#_heading=h.k8cpz0iee4nh) 9

[2.2 Package Registrazione](#_heading=h.af7u8uh1b0av) 9

[2.3 Package Autenticazione](#_heading=h.aoithu8b5nie) 9

[2.4 Package Rivista](#_heading=h.vhlfeigxdu5k) 9

[2.5 Package Shop](#_heading=h.lvzfzepg9f6j) 9

[2.6 Package Opinione](#_heading=h.rqpt1eaylwms) 9

[**3 Class Interfaces**](#_heading=h.o4brwfh4802m) **10**

[3.1 Package Registrazione](#_heading=h.htxouoz6s23w) 10

[3.2 Package Autenticazione](#_heading=h.g20xeda3tjxw) 11

[3.3 Package Rivista](#_heading=h.2u83l7k0du3u) 12

[3.4 Package Shop](#_heading=h.sl5krhmvsf7z) 13

[3.5 Package Opinione](#_heading=h.4usycf1g6gj7) 14

[**4 Design Patterns**](#_heading=h.suikzsg7ge6r) **15**

[**5 Glossario**](#_heading=h.m5gf0abpxnj1) **16**

# 

# 1 Introduzione

Rojina Review si propone essere una piattaforma dedicata ai videogiocatori, in cui questi ultimi possono consultare recensioni e notizie su videogiochi, eventualmente inserendo la propria opinione con commenti e voti e acquistando merchandising videoludico.  
In questa prima parte del documento vengono descritti gli object design goals e i relativi trade-offs e vengono stabilite delle convenzioni per la fase di implementazione, riguardanti la documentazione e nomenclatura.

## 1.1 Object design goals

**Robustezza vs Tempo:**  
Il controllo dei dati in input è un aspetto importante dell’applicazione ma farlo nella maniera più completa e corretta possibile aumenterebbe di molto i tempi di sviluppo del sistema. Si è deciso quindi, data una scadenza relativamente vicina, di effettuare meno controlli sull’input.

**Attendibilità vs Tempo:**Il sistema deve garantire l’attendibilità dei suoi servizi, come le transazioni e grazie al tempo guadagnato limitando la robustezza, essa può essere garantita.

**Riusabilità:**  
Il sistema deve basarsi sulla riusabilità, sfruttando i meccanismi di ereditarietà, design patterns ed eventuali componenti off-the-shelf

**Incapsulamento:**  
Il sistema garantisce il mascheramento dei dettagli implementativi delle classi tramite l’utilizzo di interfacce.

## 1.2 Linee guida per la scrittura del codice

Link utili alle convenzioni adottate per l’implementazione:

* **Java Sun**: [Sun's Java Style](https://checkstyle.sourceforge.io/sun_style.html)
* **HTML**: [HTML Style Guide and Coding Conventions](https://www.w3schools.com/html/html5_syntax.asp)

## 1.3 Definizioni, acronimi, e abbreviazioni

**Package:** raggruppamento di classi, interfacce o file correlati **Design pattern:** template di soluzioni a problemi ricorrenti per ottenere riuso e flessibilità.  
**Interfaccia**: insieme di signature delle operazioni offerte dalla classe.  
**Javadoc**: sistema di documentazione offerto da Java, che viene generato sotto forma di interfaccia in modo da rendere la documentazione accessibile e facilmente leggibile.

## 1.4 Riferimenti

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

* Statement Of Work;
* Requirements Analysis Document;
* System Design Document;
* Test Plan;
* Test Case Specification;
* Test Incident Report;
* Test Summary Report;

# 

# 

# 

# 

# 

# 

# 

# 

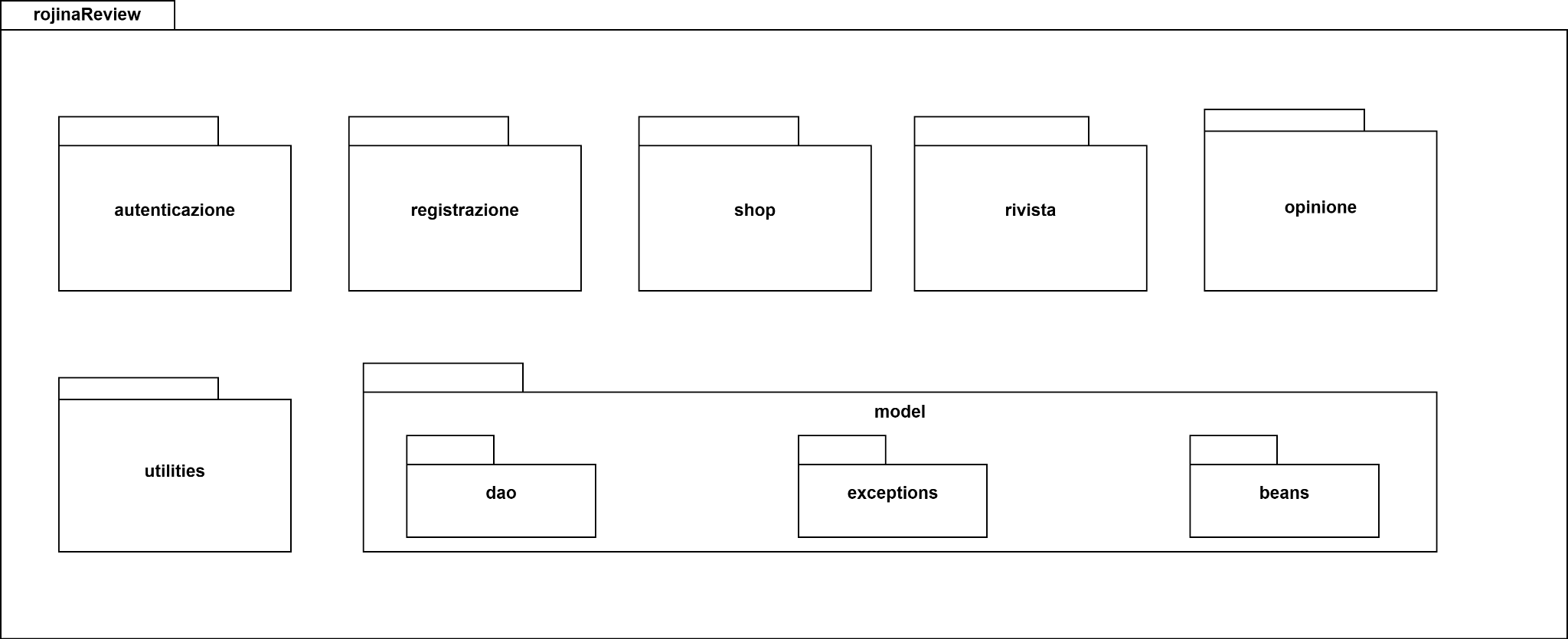
# 2 Packages

In questa sezione è mostrata la suddivisione del sistema in package, in base ai sottosistemi definiti in fase di System Design.

* **.idea**
* **database,** contiene il file per la creazione e il popolamento iniziale del database
* **src**, contiene i file sorgente
  + **main**
    - **java,** contiene le classi Java relative alle componenti Control e Model
    - **webapp**, contiene i file relativi alla View
      * **static,** contiene i file statici del sistema
        + css
        + html
        + images
        + js
      * **WEB-INF**
  + **test,** contiene il necessario per il testing
    - **java,** contiene le classi Java per l’implementazione del testing
* **target**, contiene tutti i file prodotti dal building di Maven

## 2.1 Package Rojina Review

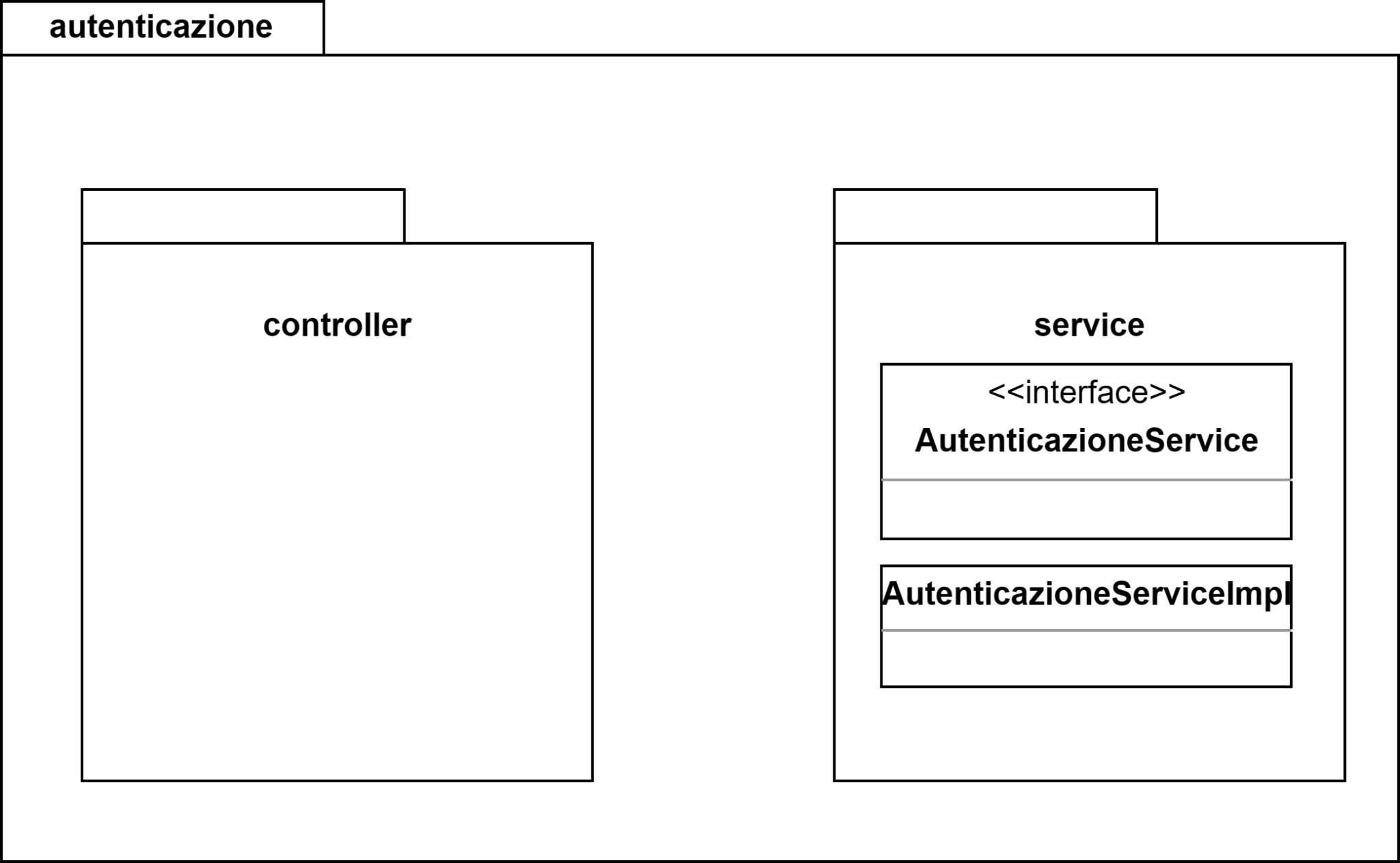
In questa sezione si mostra la struttura del package principale di Rojina Review. La struttura generale è stata costruita tenendo conto della suddivisione in sottosistemi e creando un package separato per le classi del model (entity e DAO) più un package utils per le classi di utilità del sistema.



## 2.2 Package Registrazione

# 

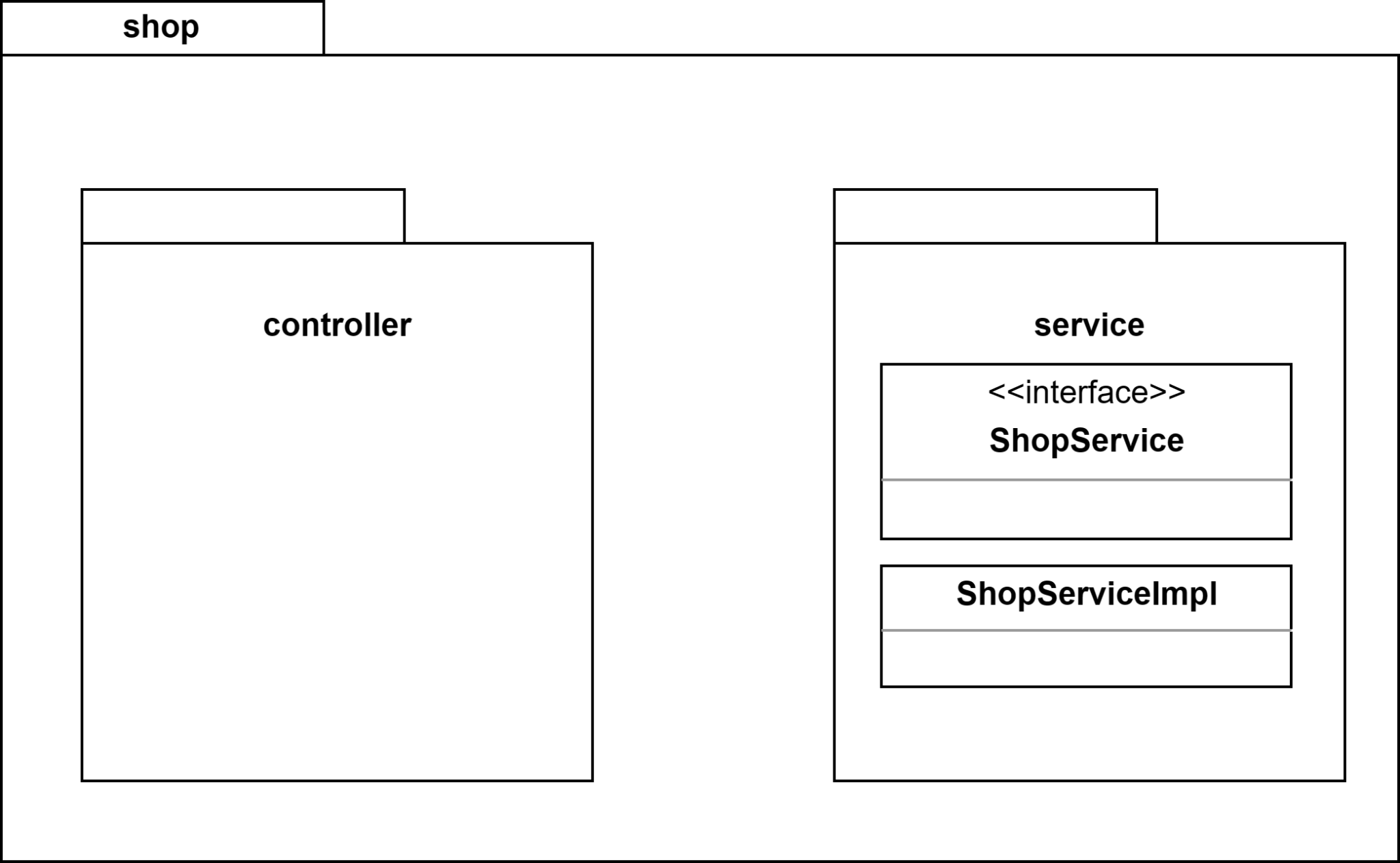
## 2.3 Package Autenticazione



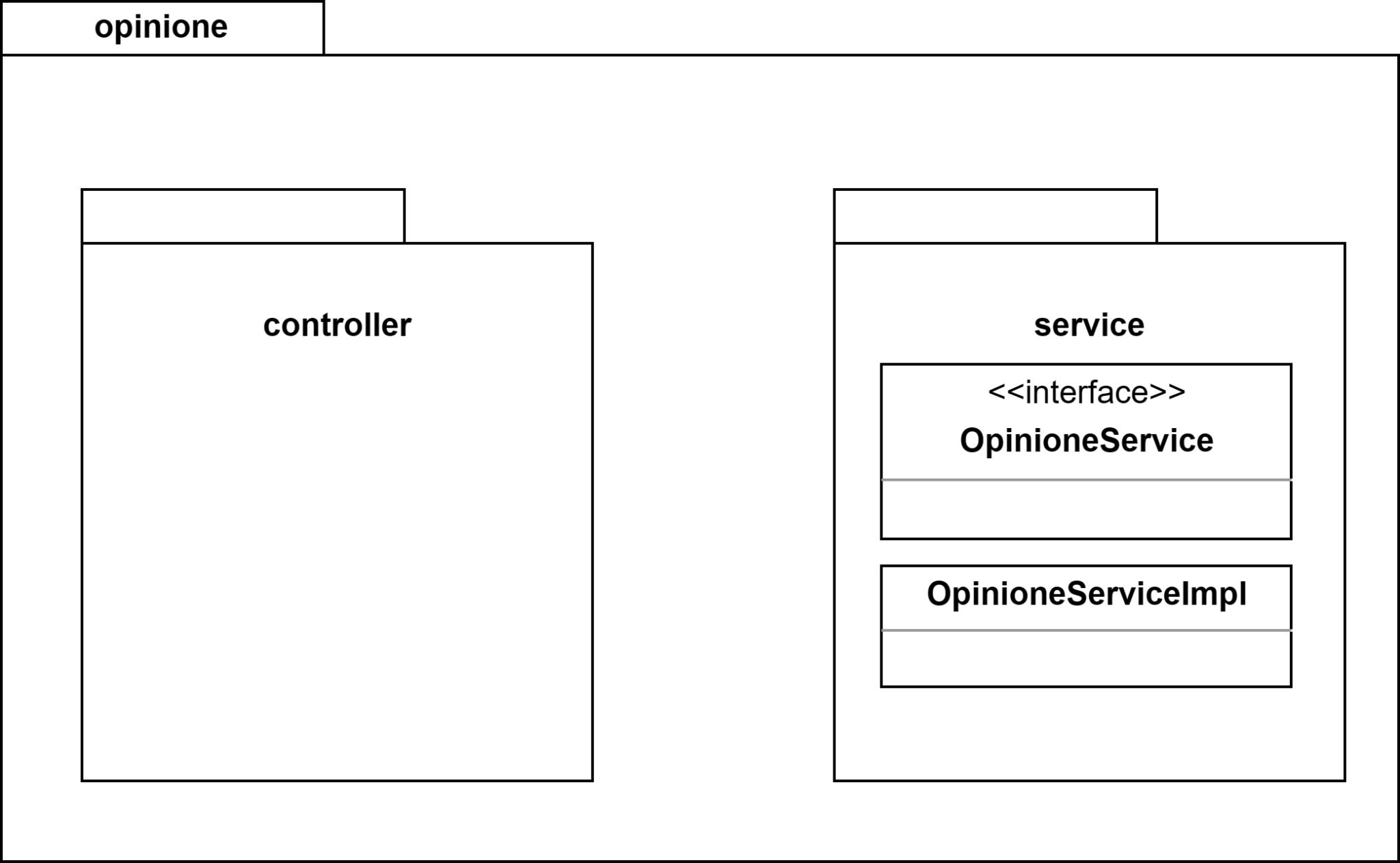
## 2.4 Package Rivista

## 

## 2.5 Package Shop



## 2.6 Package Opinione



# 

# 

# 

# 

# 

# 

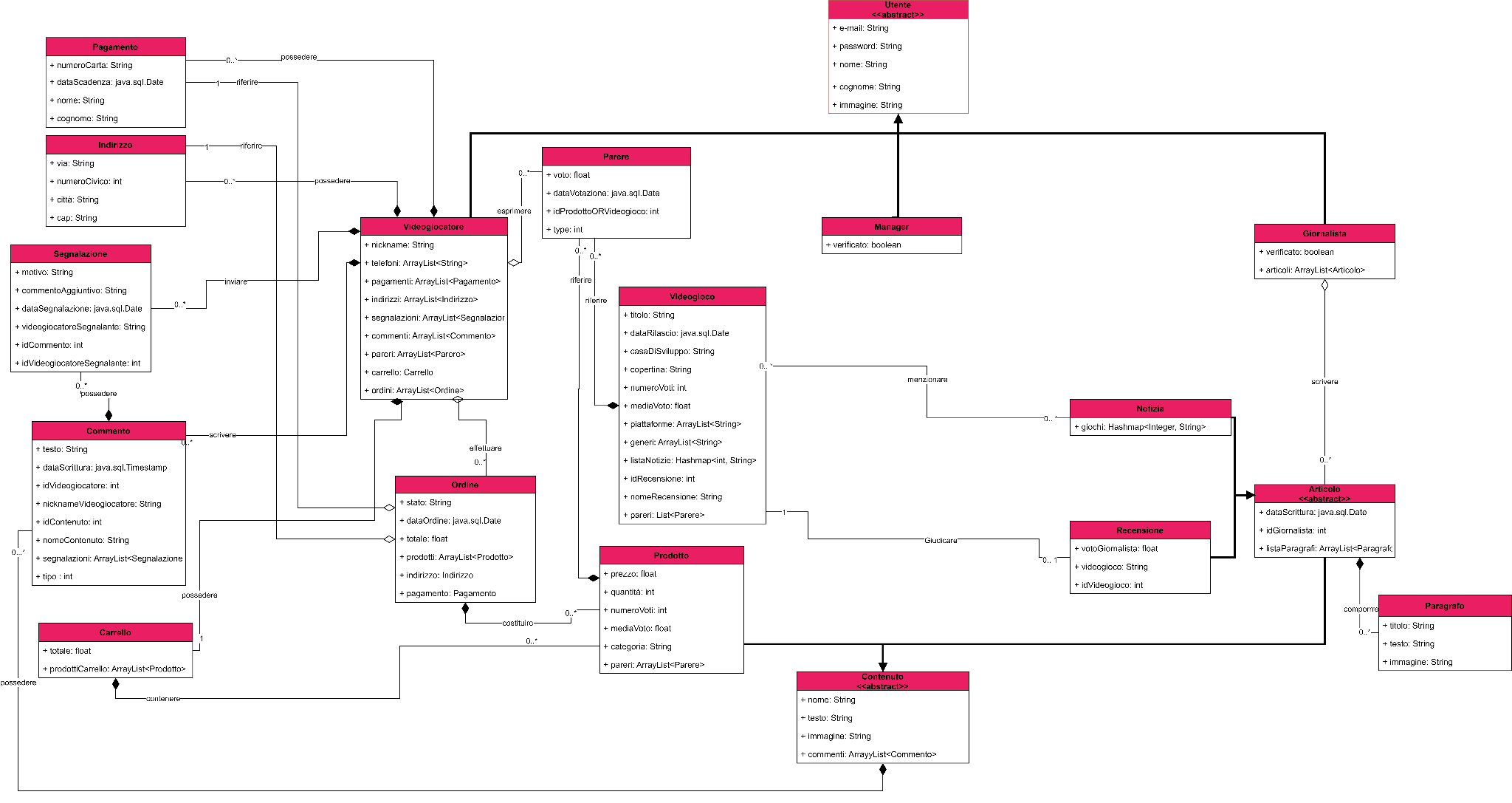
# 

# 3 Class Interfaces

Per quanto riguarda la specifica delle interfacce si farà riferimento alla documentazione JavaDOC generata attraverso i commenti, fornita nel file .**rar** tra i documenti consegnati.

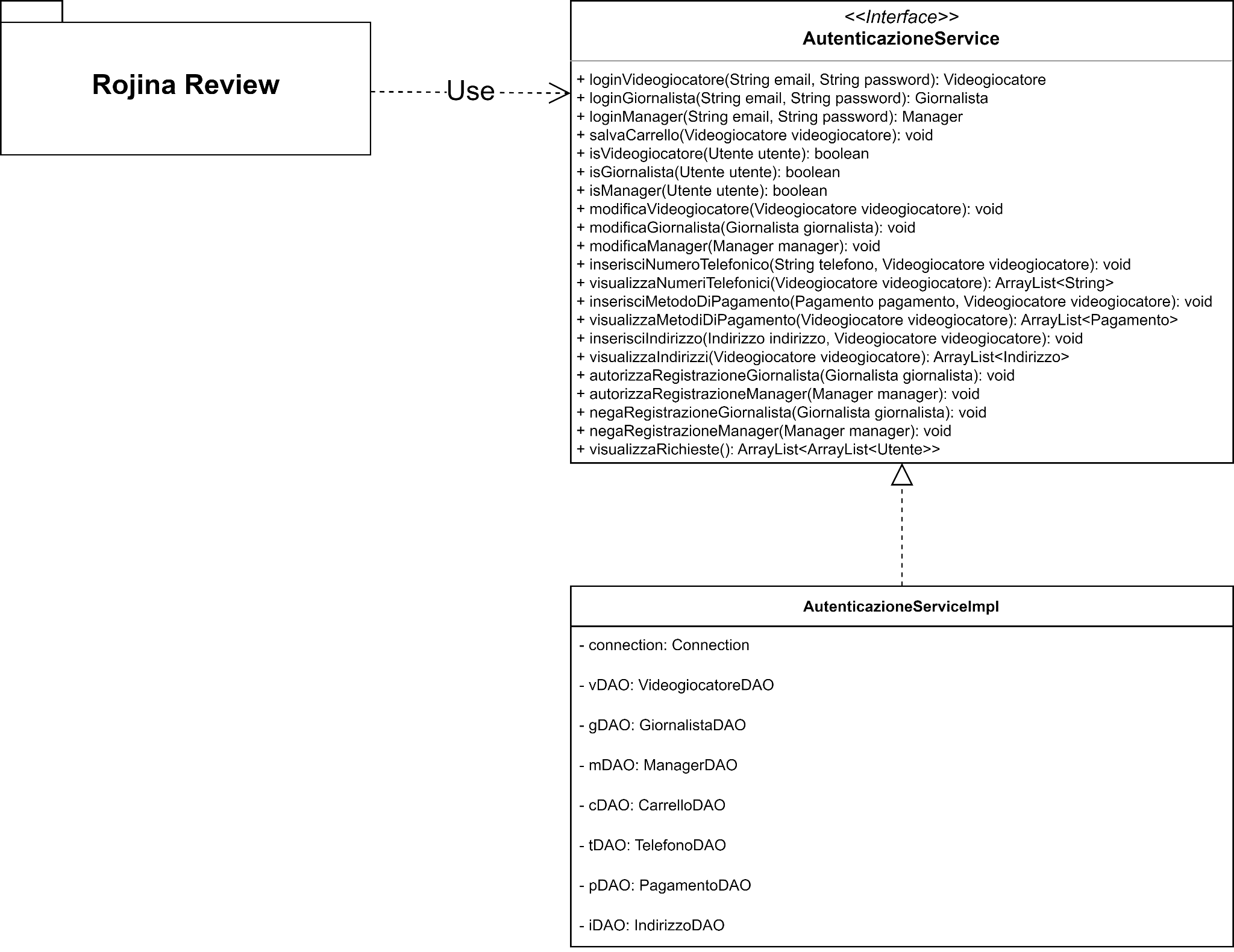
# 4 Class Diagram ristrutturato

* Sono state rimosse le classi Genere e Piattaforma (e le relative classi associative) e trasformate in attributi di Videogioco.
* E’ stata rimossa la classe associativa Videogioco-Notizia e l’associazione è stata realizzata tramite un hashmap in entrambe le entità. Ciò servirà a mostrare i videogiochi menzionati nella pagina notizia e le notizie in cui è menzionato il videogioco nella pagina videogioco (pagina non implementata)
* E’ stata rimossa l’entità Telefono e trasformata in un attributo di Videogiocatore
* In generale le associazioni non sono state realizzate usando un riferimento all’oggetto ma ricopiando alcuni attributi dello stesso. Ciò è stato fatto in quanto non sono necessari tutti gli attributi dell’oggetto da referenziare (Es. Nella pagina di un contenuto in cui vengono mostrati i commenti c’è bisogno solo del nickname del videogiocatore che ha scritto il commento: nell’entità Commento è stato inserito solo l’attributo nickname dell’entità Videogiocatore)



# 5 Elementi di riuso

## 5.1 Design Pattern usati

**Facade**Il facade è un design pattern che permette di accedere alle operazioni di un sottosistema complesso tramite un’unica interfaccia semplificata più ad alto livello. Viene così mascherata la complessità delle diverse classi che si utilizzano per portare a compimento un’operazione e viene ridotto l’accoppiamento.  
Rojina Review è un sistema molto complesso e anche i suoi sottosistemi lo sono, con diversi DAO che si utilizzano a vicenda per implementare la logica di business. Per evitare di gestire tale complessità nella logica di controllo (Servlets), si è deciso di spostare la responsabilità a delle interfacce, una per ogni sottosistema, che offrono dei servizi, attuando di fatto il Facade. ****

## 5.2 Componenti terzi

Il sistema è deployato sul webserver Tomcat.   
Inoltre è stato usato il driver JDBC per la connessione con il database e Maven per il building e compilazione del progetto.

# 6 Glossario

| **Sigla/Termine** | **Definizione** |
| --- | --- |
| **Package** | Raggruppamento di classi ed interfacce |
| **DAO** | Data Access Object, implementazione dell’omonimo pattern architetturale che si occupa di fornire un accesso in modo astratto ai dati persistenti |
| **Controller** | Classe che si occupa di gestire le richieste effettuate dal client |
| **Service** | Classe che implementa la logica di business, viene utilizzata dal controller o da un altro sottosistema |
| **Facade** | Un oggetto che permette, attraverso un’interfaccia più semplice, l’accesso a sottosistemi che espongono interfacce complesse e molto diverse tra loro |