Requirement

Analysis Document

GameSquare

Sommario

[1. Introduzione 3](#_Toc64726799)

[**1.1 Scopo del sistema** 3](#_Toc64726800)

[**1.2 Ambito del sistema** 3](#_Toc64726801)

[**1.3 Obiettivi e criteri di successo** 4](#_Toc64726802)

[**1.4 Definizioni, acronimi e abbreviazioni** 4](#_Toc64726803)

[**1.5 Riferimenti** 4](#_Toc64726804)

[**1.6 Panoramica** 4](#_Toc64726805)

[2. Sistema corrente 5](#_Toc64726806)

[3. Sistema proposto 5](#_Toc64726807)

[**3.1 Panoramica** 5](#_Toc64726808)

[**3.2 Requisiti funzionali** 6](#_Toc64726809)

[**RF\_AUT: AUTENTICAZIONE** 6](#_Toc64726810)

[**RF\_USER: CONTRIBUTI UTENTE** 6](#_Toc64726811)

[**RF\_NAV: NAVIGAZIONE DATABASE** 6](#_Toc64726812)

[**RF\_ADMIN: AMMINISTRAZIONE** 6](#_Toc64726813)

[**3.3 Requisiti non funzionali** 7](#_Toc64726814)

[**3.3.1 Usabilità** 7](#_Toc64726815)

[**3.3.2 Affidabilità** 7](#_Toc64726816)

[**3.3.3 Prestazioni** 7](#_Toc64726817)

[**3.3.4 Manutenibilità** 7](#_Toc64726818)

[**3.3.5 Implementazione** 7](#_Toc64726819)

[**3.3.6 Interfaccia** 7](#_Toc64726820)

[**3.3.7 Packaging** 8](#_Toc64726821)

[**3.3.8 Legali** 8](#_Toc64726822)

[**3.4 Modelli di sistema** 8](#_Toc64726823)

[**3.4.1 Scenari** 8](#_Toc64726824)

[**3.4.2 Use Case Model** 10](#_Toc64726825)

[**3.4.3 Object Model** 14](#_Toc64726826)

[**3.4.4 Dynamic Model** 26](#_Toc64726827)

1. Introduzione

## **1.1 Scopo del sistema**

GameSquare propone una piattaforma centralizzata e pensata per videogiocatori, a prescindere dalla loro esperienza, e sviluppatori.

Gli utenti registrati potranno aggiungere al proprio profilo i videogiochi in loro possesso, catalogandoli come “acquistati”, “in corso di gioco”, “completati”, e opzionalmente aggiungere a un gioco completato l’accezione di “platinato” (completato al 100%), con possibilità di spostarli da una categoria all’altra. Si potranno inoltre recensire videogiochi aventi quantomeno lo status di acquistato, fornendo un voto in punti percentuali e un commento. La recensione dovrà essere significativa in quanto ci sarà la possibilità per gli altri utenti di votare le recensioni per stabilirne l’utilità. Ogni utente potrà proporre al gestore del catalogo di aggiungere un titolo mancante, fornendo della documentazione che prova l’esistenza dello stesso. Infine, i giocatori avranno un punteggio personale, che aumenterà completando o platinando giochi, e terrà conto della difficoltà degli stessi mettendo in relazione la quantità di persone che lo hanno giocato e quanti lo hanno finito per calcolare il punteggio guadagnato per il completamento, e la quantità di persone che lo hanno finito con quanti lo hanno platinato per calcolare il punteggio ottenuto per il conseguimento del platino.

Ogni gioco avrà la sua pagina, in cui sarà possibile visualizzare le informazioni base del titolo, le recensioni degli utenti e quanto sono stati aggiunti a una determinata categoria rispetto alle altre. La pagina di ogni gioco avrà una sezione commenti suddivisa per categorie (FAQs, discussione generale, tornei, etc.).

Una piattaforma del genere è utile all’utente che vuole avere un’idea d’insieme di un titolo prima di acquistarlo, come anche a coloro che vogliono chiedere a giocatori più esperti come completare sezioni difficoltose, o semplicemente chi vuole discutere di un titolo di suo gradimento.

Inoltre può giovare anche agli sviluppatori, in quanto possono rendersi conto di quante persone apprezzino il loro prodotto tanto da completarlo, e ovviamente leggere e scambiare commenti con la community.

## **1.2 Ambito del sistema**

Il mondo dell’intrattenimento videoludico è diventato negli ultimi anni un fenomeno esteso a centinaia di milioni di persone, grazie a servizi e piattaforme che permettono un semplice accesso a cataloghi molto grandi di videogiochi (e. g. Steam, PSN, Nintendo eShop, etc.), e fenomeni mediatici come la comunità di videogiocatori su YouTube, gli eSports e le molte riviste online specializzate.

Con questa crescita, molti usufruitori del prodotto sentono il bisogno di catalogare i propri giochi e analizzarli per avere un’idea di quanti titoli in proprio possesso debbano ancora essere completati (il cosiddetto backlog). Inoltre la ricerca di una community specifica per un determinato gioco, che sia per fare domande riguardo al completamento dello stesso, organizzare tornei in caso di un titolo competitivo, o semplicemente discussione generale, si fa sempre più difficoltosa. Ciò avviene a causa dell’aumento più che lineare della quantità di titoli sul mercato, grazie anche al successo recente di titoli creati da sviluppatori indipendenti e la riscoperta di giochi di generazioni passate (retrogaming).

Un altro aspetto da considerare è la quantità di servizi diversi che offrono qualche tipo di aggregazione per i giocatori, come gruppi Facebook, 4chan, Discord e le miriadi di forum e imageboards che tuttavia non sono centralizzati, e raramente pensati per ospitare questo specifico tipo di utenza.

## **1.3 Obiettivi e criteri di successo**

Obiettivi primari:

* Fornire una piattaforma centralizzata per tenere traccia del proprio catalogo videoludico.
* Compartimentalizzare le discussioni riguardanti uno specifico gioco per rendere più intuitivo lo scambio di opinioni o l’organizzazione di community.
* Offrire agli sviluppatori l’occasione di avere feedback diretto sulle loro opere.

Criteri di successo:

* Il sito deve essere online 24/7.
* Deve avere un tempo di risposta basso.

## **1.4 Definizioni, acronimi e abbreviazioni**

NA

## **1.5 Riferimenti**

NA

## **1.6 Panoramica**

Il seguente documento “GameSquare RAD” è diviso in sezioni ed ha la seguente composizione:

**- Sezione di Introduzione:**

In questa sezione è spiegato in breve qual è l'ambito del progetto e lo scopo del sistema che verrà implementato definendo gli obiettivi e i criteri di successo del sistema.

**-Sezione Sistema Corrente:**

Questa sezione resterà vuota in quanto il sistema proposto verrà creato da zero e non andrà a sostituire alcun servizio pubblico esistente.

**-Sezione SISTEMA PROPOSTO:**

In questa sezione viene inizialmente data una panoramica del sistema che deve essere realizzato e cosa andrà ad implementare.

Si procede con la sottosezione dei Requisiti Funzionali del sistema dove vengono identificate le

funzionalità che Il sistema deve offrire.

I Requisiti Non Funzionali seguono il modello FURPS+, composto da: Usabilità, Affidabilità, Prestazioni, Manutenibilità, Implementazione, Interfaccia, Packaging, Legali. Seguono poi gli scenari, gli use-case e gli use-case diagram.

2. Sistema corrente

NA

3. Sistema proposto

## **3.1 Panoramica**

Il sistema da noi proposto è un sito web che offrirà servizi sia a utenti facenti pare della community videoludica, siano essi sviluppatori, nella forma di discussione con i fruitori del loro prodotto, o i fruitori stessi, nella forma di liste per tenere traccia del proprio catalogo, discussioni con altri utenti e suggerimenti.

L’utente potrà:

* Visionare il proprio profilo con i relativi dati più rilevanti.
* Accedere alle singole liste da lui create e modificarle a piacimento.
* Creare o contribuire a discussioni già esistenti, oltre che crearne di nuove.
* Valutare recensioni di altri utenti, oltre che recensire esso stesso dei prodotti.
* Proporre l’aggiunta o la modifica di un gioco, se dispongono delle prove della sua esistenza.

Gli amministratori potranno:

* Moderare e rimuovere qualsiasi tipo di contenuto (descrizioni, recensioni, etc.) qualora questo sia inopportuno.
* Accettare o rifiutare richieste da parte degli utenti di creazione di discussioni o aggiunta di un gioco al Database.
* Aggiungere essi stessi un gioco al Database, oltre che modificarne le informazioni.

Gli utenti non registrati potranno:

* Navigare il database di giochi tramite ricerca testuale o giochi correlati.
* Visualizzare la pagina di un prodotto e le informazioni relative, comprese recensioni e discussioni a riguardo.

## **3.2 Requisiti funzionali**

### **RF\_AUT: AUTENTICAZIONE**

* **RF\_AUT\_01 – Registrazione** –Priorità Alta

Il sistema deve permettere la registrazione di un utente.

* **RF\_AUT\_02 – Login** –Priorità Alta

Il sistema deve permettere l’accesso di un utente.

* **RF\_AUT\_03 – Log out** –Priorità Alta

Il sistema deve permettere all’utente di terminare la propria sessione.

* **RF\_AUT\_04 – Recupero Password** –Priorità Media

Il sistema deve permettere all’utente di recuperare la propria password.

### **RF\_USER: CONTRIBUTI UTENTE**

* **RF\_USER\_01 – Aggiungi Gioco alla Lista** –Priorità Alta

Il sistema deve permettere all’utente di aggiungere un qualsivoglia gioco a una lista a piacimento.

* **RF\_USER\_02 – Vota Elemento** –Priorità Alta

Il sistema deve permettere all’utente di scrivere una recensione.

* **RF\_USER\_03 – Commenta** –Priorità Alta

Il sistema deve permettere all’utente di contribuire a una discussione tramite un commento.

* **RF\_USER\_04 – Crea Thread** –Priorità Media

Il sistema deve permettere all’utente di richiedere la creazione di una discussione.

* **RF\_USER\_05 – Richiesta Admin** –Priorità Media
* Il sistema deve permettere all’utente di richiedere un’aggiunta al o una modifica del database all’admin.

### **RF\_NAV: NAVIGAZIONE DATABASE**

* **RF\_NAV\_01 – Cerca Elemento** –Priorità Alta

Il sistema permette all’utente di eseguire una ricerca testuale sul database di giochi.

* **RF\_NAV\_02 – Visualizza Elemento** –Priorità Alta

Il sistema deve far visualizzare la pagina dedicata di un gioco con le relative informazioni.

### **RF\_ADMIN: AMMINISTRAZIONE**

* **RF\_ADMIN\_01 – Valuta Richiesta** –Priorità Alta

L’admin deve poter valutare (accettare o rifiutare) richieste di creazione o modifica di contenuti da parte dell’utenza.

* **RF\_ADMIN\_02 – Aggiungi o modifica Elemento** –Priorità Alta

L’admin deve poter modificare o aggiungere giochi al database.

* **RF\_ADMIN\_03 – Rimuovi Elemento** –Priorità Alta

L’admin deve poter rimuovere elementi creati dall’utenza che ritiene non opportuni.

## **3.3 Requisiti non funzionali**

### **3.3.1 Usabilità**

Il sistema deve:

* Permettere di accedere alle funzioni principali (ricerca di un gioco, accesso al profilo utente) in modo intuitivo e dalla homepage.
* Contenere una sezione “Aiuto” che funge da breve guida per l’utente.

### **3.3.2 Affidabilità**

Il servizio, in quanto sito web, dovrà essere online 24/7 con basso rischio di crash del servizio di hosting.

Inoltre, agli utenti non sarà richiesto di fornire dati sensibili in quanto non utili ai fini dell’utilizzo di GameSquare.

### **3.3.3 Prestazioni**

* Il tempo di risposta di una query deve essere basso, in condizioni medie minore o uguale a un secondo.
* Non c’è un limite realistico ai giochi che è possibile aggiungere al database.
* Non c’è un limite realistico ai giochi che l’utente può avere nella sua libreria.
* Il servizio deve funzionare correttamente su dispositivi con diverse risoluzioni e aspect ratio.

### **3.3.4 Manutenibilità**

* In futuro ci sarà, per ogni gioco, una sezione dei giochi consigliati.
* La rimozione dei bug sarà a carico degli sviluppatori.
* Le modifiche e le migliorie saranno a carico di sviluppatori e progettisti, in base alle direttive e alle richieste dell’utenza.

### **3.3.5 Implementazione**

* La parte back-end deve essere programmata in Java.
* Il database deve essere programmato in SQL.
* La parte front-end deve essere strutturata tramite CSS e JavaScript.

### **3.3.6 Interfaccia**

* Il sistema deve interagire con un database di videogiochi importandone il contenuto.
* Il contenuto del database sarà gestito dalle API fornite dallo stesso.
* Il sito si affiderà a un sito di hosting per essere online.

### **3.3.7 Packaging**

Il sistema, in quanto sito web dovrà essere supportato da tutti i browser più usati.

### **3.3.8 Legali**

NA

## **3.4 Modelli di sistema**

### **3.4.1 Scenari**

#### **-Scenario 1:**

***Nome Scenario:*** ricercaTorneo

***Attori Partecipanti:*** Nico:utente

***Descrizione:*** Nico, un giocatore di Tekken 7, vuole partecipare al suo primo torneo competitivo online.

***Flusso di eventi:***

1. Si reca sulla homepage di GameSquare e cerca il gioco interessato.
2. Sulla pagina del gioco in questione e clicca sulla sezione “discussioni”.
3. Filtra le discussioni per argomento selezionando “competitivo”.
4. Nota varie organizzazioni di tornei online, fra cui una per giocatori con poca esperienza in tale ambito.
5. Esprime il suo desiderio di partecipare in quanto orari e modalità gli sono convenienti.
6. L’autore del thread risponde a Nico e si mettono d’accordo sulle modalità di partecipazione.

#### **-Scenario 2:**

***Nome Scenario:*** communityFeedback

***Attori Partecipanti:*** Lorenzo:utente

***Descrizione:*** Lorenzo, uno sviluppatore indipendente, ha pubblicato il suo primo videogioco da pochi giorni e sta cercando del feedback a riguardo.

***Flusso di eventi:***

1. Cerca il proprio gioco sulla piattaforma, visualizzandone la pagina.
2. Nota che l’85% delle persone che lo hanno acquistato lo hanno anche giocato, indice di alta curiosità da parte del pubblico.
3. Nota anche che il 70% delle persone che lo hanno giocato lo hanno anche completato, indice forse di scarsa longevità.
4. Si dirige poi nella sezione recensioni, dove nonostante un generale elogio ha conferma della scarsa difficoltà del titolo.
5. Si dirige infine nella sezione discussioni dove si annuncia come creatore del titolo in questione e richiede consigli e feedback direttamente ai propri acquirenti.

#### **-Scenario 3:**

***Nome Scenario:*** richiestaAggiuntaGioco

***Attori Partecipanti:*** Francesco:utente

***Descrizione:*** Francesco, un appassionato di videogiochi meno conosciuti, non riesce a trovare su GameSquare il titolo “Garage: Bad Dream Adventure”.

***Flusso di eventi:***

1. Si reca sull’apposita pagina contenente il form per proporre l’aggiunta di un gioco.
2. Inserisce i requisiti, ossia il nome del gioco e una fonte attendibile della sua esistenza, in questo caso un riferimento a un video del gameplay caricato per intero su YouTube.
3. Dopo qualche giorno, la sua proposta viene approvata e il gioco aggiunto al database.
4. Francesco aggiunge il gioco al suo catalogo di titoli completati.

#### **-Scenario 4:**

***Nome Scenario:*** eliminazioneContenuto

***Attori Partecipanti:*** Piero:admin

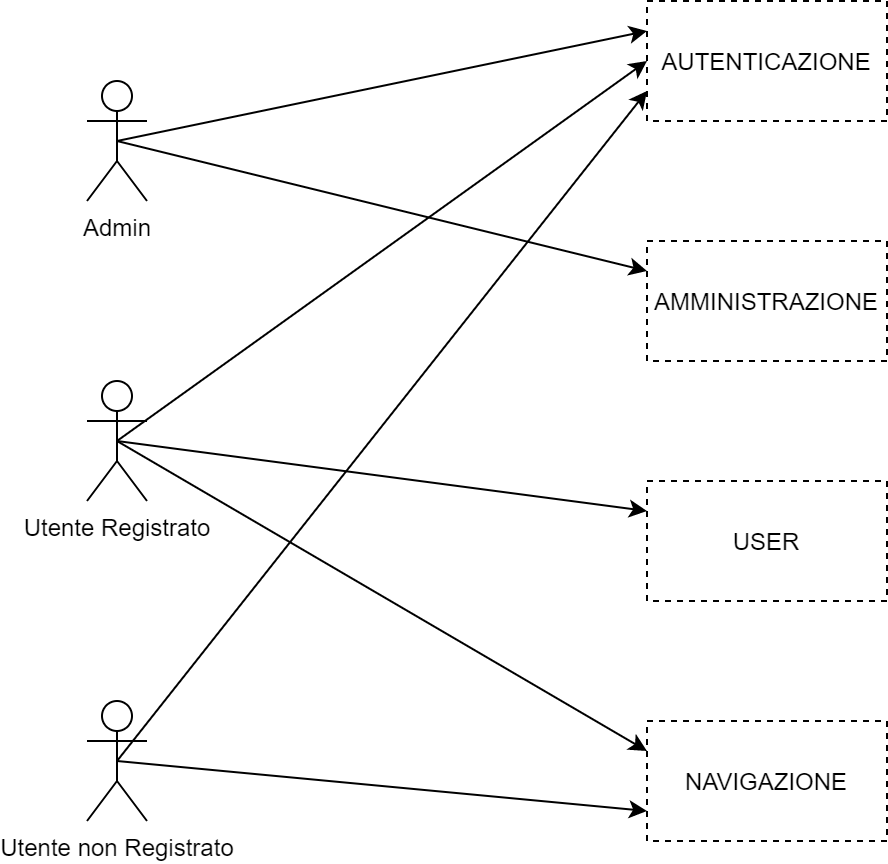
***Descrizione:*** Piero, un admin di GameSquare, nota che un contenuto è inopportuno.

***Flusso di eventi:***

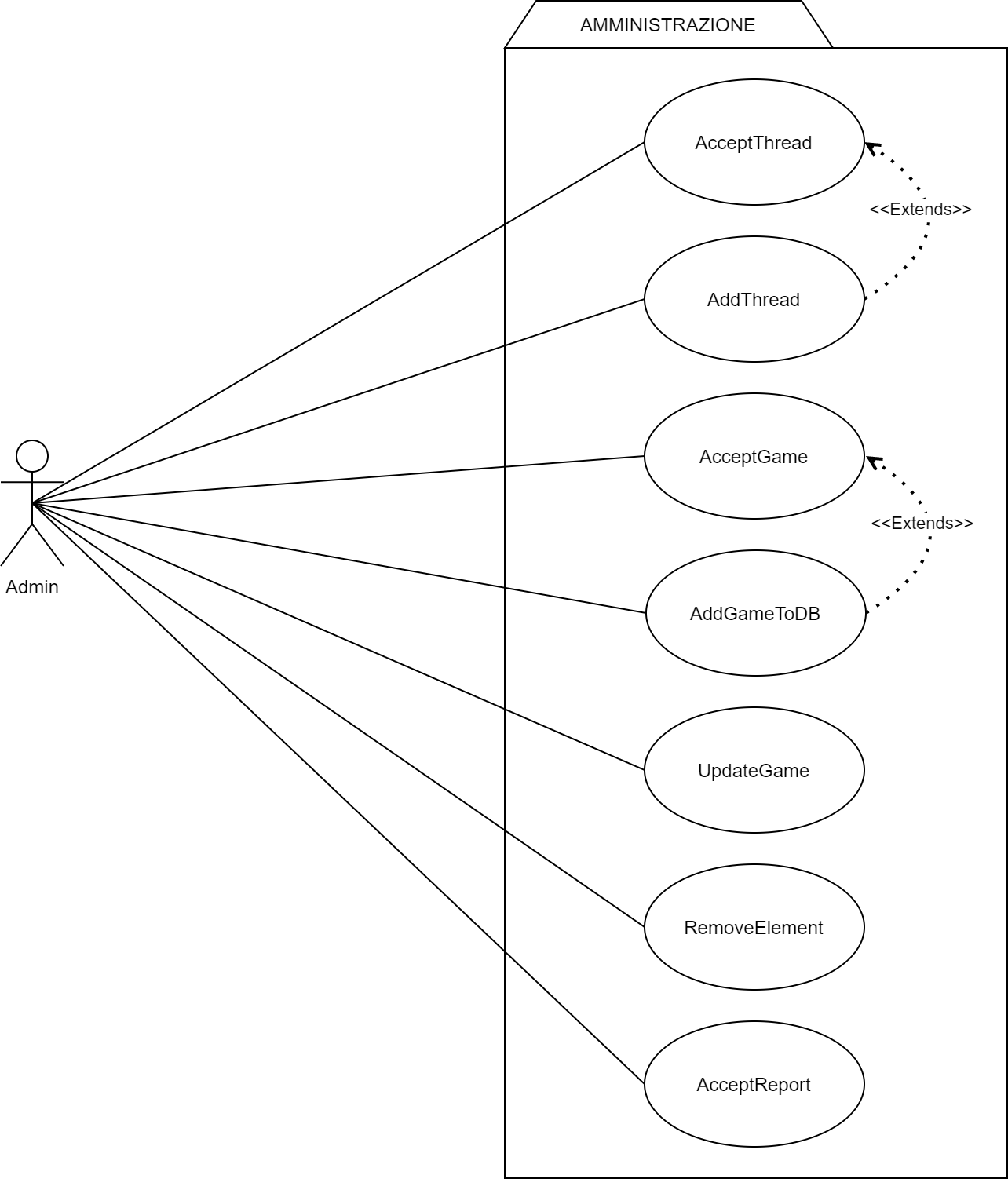
1. Molti utenti hanno segnalato una discussione come non opportuna, a causa della mancanza di contributi significativi alla community e all’eccessiva volgarità.
2. L’admin nota le molte segnalazioni e decide di rimuovere il contenuto nocivo.
3. Tramite un apposito tasto elimina il contenuto inviando una segnalazione privata all’utente che lo ha creato incoraggiandolo a soppesare meglio le proprie parole la prossima volta.

### **3.4.2 Use Case Model**

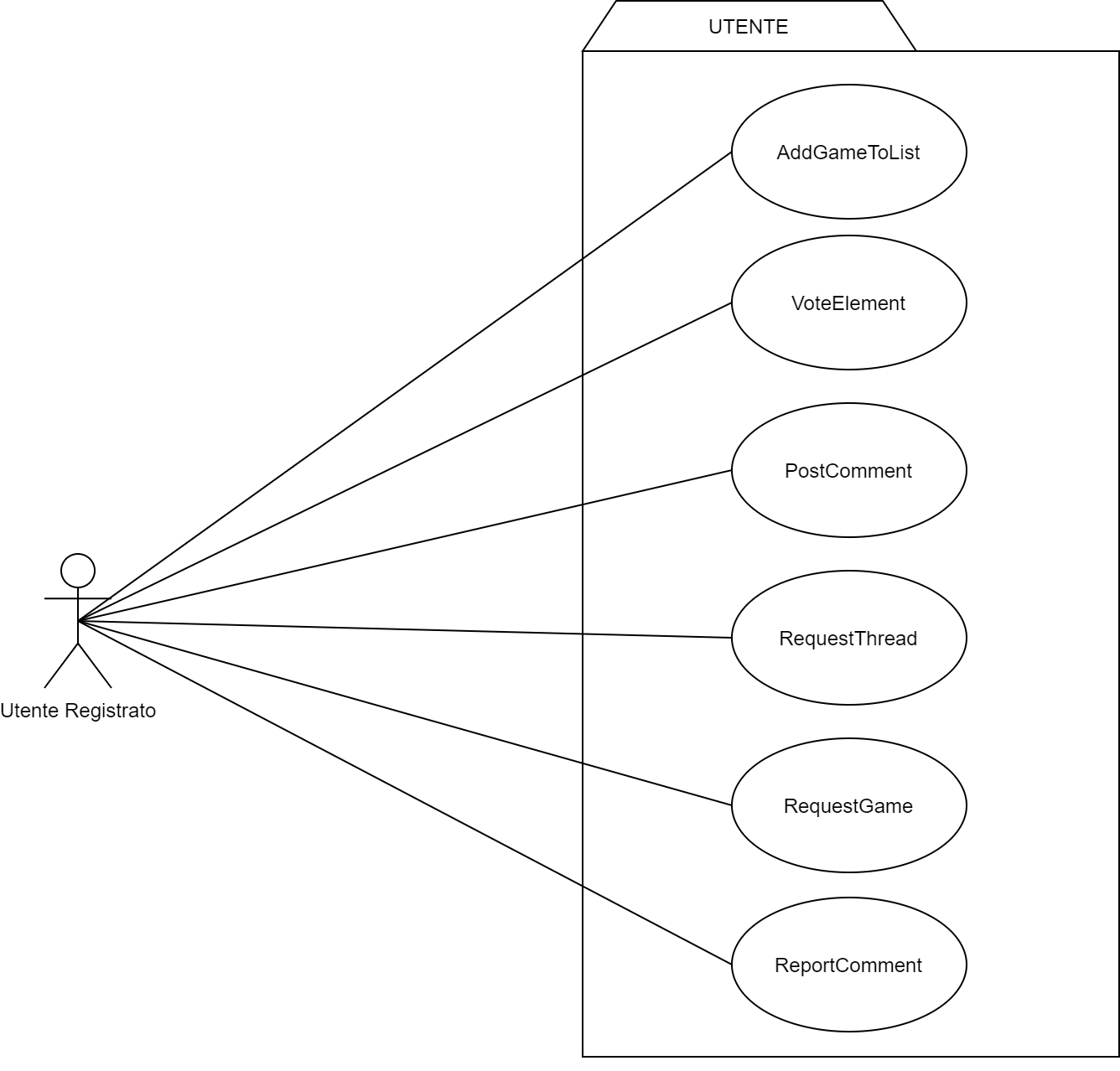
#### **-UCD\_GENERALE**



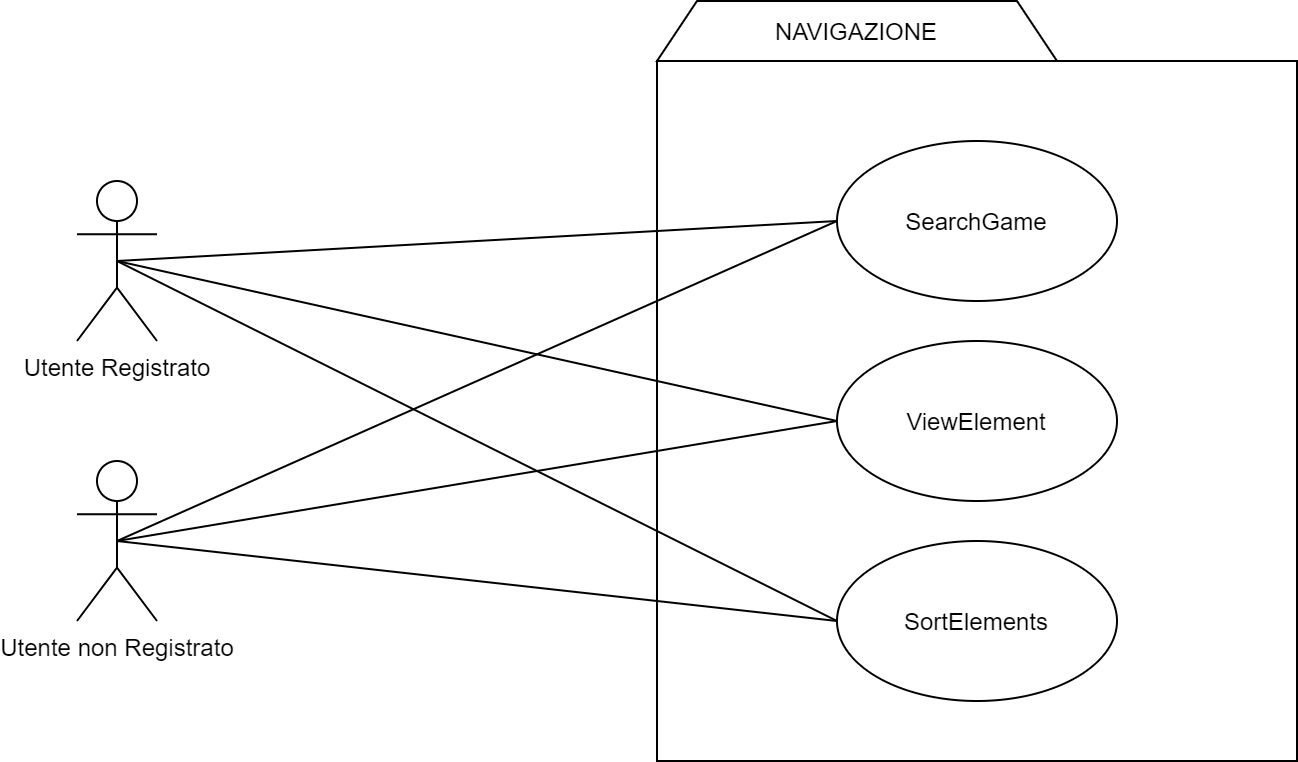
#### **-UCD\_ADMIN**



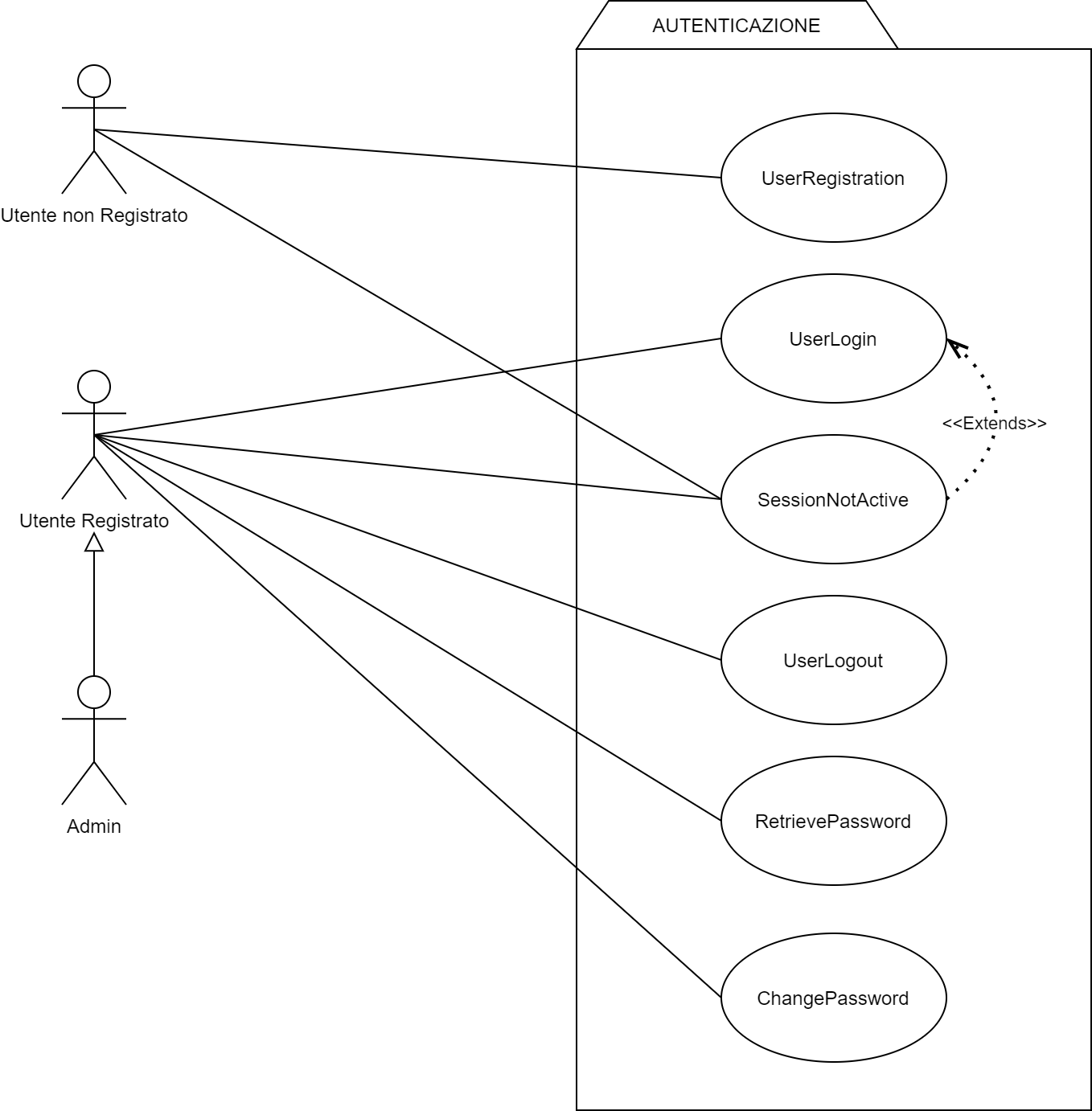
#### **-UCD\_USER**



#### **-UCD\_NAV**



#### **-UCD\_AUT**



### **3.4.3 Object Model**

[aggiungere object diagram corretti]

A seguito, la tabella riassuntiva degli oggetti:

|  |  |  |
| --- | --- | --- |
| –Nome oggetto | Tipologia | Descrizione |
| RegistrationForm | Boundary | Form da compilare per la registrazione, con campi da riempire e submit |
| LoginPage | Boundary | Pagina per effettuare il login da utente registrato o da admin |
| Homepage | Boundary | Pagina principale visualizzata se non si ha una sessione attiva |
| LoggedHomePage | Boundary | Pagina principale a cui si viene reindirizzati una volta attivata la sessione |
| ListAdder | Boundary | Oggetto per aggiungere un titolo a una particolare lista dell’utente |
| VoteField | Boundary | Campo per dare una valutazione ad un elemento, da utente registrato |
| CommentField | Boundary | Form costituito da un TextField (non vuoto) e Submit per inviare un commento (da utente registrato) |
| SearchField | Boundary | Barra di ricerca per effettuare una ricerca per parola chiave |
| SortField | Boundary | Campo per effettuare una ricerca per parametri |
| ResultsPage | Boundary | Pagina che mostra i risultati di una ricerca effettuata nel sito |
| ElementLink | Boundary | Link che reindirizza ad un determinato elemento del sito |
| ElementPage | Boundary | Pagina che mostra tutte le informazioni relative ad un determinato elemento |
| ThreadForm | Boundary | Form da compilare per richiedere una nuova discussione (utente registrato) o crearne una (admin) |
| GameRequestForm | Boundary | Form con cui l’utente registrato può richiedere l’aggiunta di un nuovo gioco |
| CommentReportForm | Boundary | Form con cui l’utente registrato può esplicitare le motivazioni per la segnalazione di un commento inappropriato |
| LogoutButton | Boundary | Button per terminare la propria sessione attiva |
| EmailRetrieveForm | Boundary | Form da compilare per indicare l’indirizzo email presso il quale inviare il link di recupero della password |
| PasswordModifierForm | Boundary | Form da compilare per la modifica della password |
| RequestPage | Boundary | Pagina accessibile agli admin con le richieste di aggiunta di elementi da parte degli utenti |
| AcceptReqButton | Boundary | Consente all’admin di accettare una richiesta di un utente |
| NewGameForm | Boundary | Form che permette all’admin di inserire un nuovo gioco nel database |
| GameUpdaterForm | Boundary | Form che permette all’admin di aggiornare i dati relativi ad un determinato gioco |
| ElementRemoverButton | Boundary | Permette all’admin di rimuovere un determinato elemento dal database |
| DataVerifier | Control | Consente al sistema di verificare la correttezza dei dati e la loro coerenza rispetto a quelli persistenti |
| SessionVerifier | Control | Consente al sistema di verificare lo stato della sessione degli utenti che hanno accesso al sito |
| ListManager | Control | Consente al sistema di organizzare le liste di un determinato utente, verificando la presenza di oggetti ed eventualmente aggiungendoli, togliendoli o spostandoli da una lista all’altra, calcolando conseguentemente il punteggio associato |
| QueryManager | Control | Consente al sistema di verificare i parametri di una ricerca ed effettuare query verso i dati memorizzati su database |
| FieldVerifier | Control | Consente al sistema di verificare la correttezza dei dati inseriti in un form |
| EmailSender | Control | Consente al sistema di inviare una email presso un determinato indirizzo verificato |
| PersistenceManager | Control | Consente al sistema di occuparsi della gestione dei dati persistenti |
| GameData | Entity | Entità che contiene i dati persistenti relativi a un gioco |
| UserData | Entity | Entità che contiene le informazioni persistenti su un determinato utente, compreso il tipo (utente standard, admin) e il punteggio di abilità personale |
| ListData | Entity | Sotto-entità che mostra i dati relativi alle liste dei giochi memorizzati da uno specifico utente |
| VoteData | Entity | Sotto-entità che tiene traccia degli elementi votati da un utente e del rispettivo voto |
| CommentData | Entity | Sotto-entità che mostra i dati relativi ai commenti pubblicati da un utente per specifici giochi |
| ThreadData | Entity | Sotto-entità che contiene i dati relativi a una specifica discussione per un determinato gioco |
| RequestData | Entity | Entità che mostra il tipo di richiesta inviata all’admin per l’aggiunta di giochi o discussioni |

#### **3.4.3.1 Class Diagrams**

##### **- CD\_AUT**



##### **- CD\_NAV**



##### **- CD\_USER**



##### **- CD\_ADMIN**



### **3.4.4 Dynamic Model**