Università degli Studi di Salerno
Dipartimento di Informatica

GameUp

System Design Document

Studenti
Francesco Foglia

Docente
Prof. Andrea De Lucia

Anno Accademico 2020-2021

Cronologia Revisioni

Data	Versione	Descrizione
09/01/2021	1.0	Analisi iniziale del sistema assieme alla sua scomposizione, la descrizione dei servizi di ciascun sottosistema e alla scelta dei trade-off da rispettare, oltre che al mapping hardware/-software e l'analisi delle condizioni limite
20/01/2021	1.1	Conversione in formato TeX dell'intero documento

Indice

Capitolo 1

Introduzione

1.1 Obiettivo del sistema

Il sistema proposto vuole porsi come punto di accesso principale al gaming mobile. Ciò significa rispondere a vari tipi di richieste: in primis bisogna offrire una serie di funzionalità simili ad un e-commerce per permettere l'acquisto di videogiochi mobile, indipendentemente dalla piattaforma usata. Inoltre, il sistema deve fungere da intermediario tra sviluppatori e clienti e deve quindi poter permettere agli sviluppatori di proporre i propri contenuti agli utenti, sia permettendo la vendita e la gestione di essi, sia agevolando la costruzione di una comunità fiorente per i propri progetti, in modo da poter essere in stretto contatto con i propri clienti affezionati tramite discussioni da entrambe le parti con il videogioco come argomento principale. Per ultimo, questo contenuto deve essere gestito dal personale dietro al sistema GameUp, quindi deve offrire delle funzionalità di un gestionale per agevolare la moderazione del contenuto, assieme a modalità di autenticazione per tutti i tipi di utenti in modo da poter tracciare l'origine dei contenuti e poter moderare la piattaforma in modo efficace. Dallo studio fatto in fase di analisi, possiamo dividere il sistema in una serie di sottosistemi:

- Gestione dell'utenza (autenticazione e gestione profili)
- Gestione dei videogiochi (distribuzione, pubblicazione e gestione di essi)
- Gestione dei contenuti (moderazione dei contenuti sui forum dei singoli videogiochi)

1.2 Obiettivi di design e compromessi di progettazione

1.2.1 Obiettivi di design

Il target del sistema proposto è estremamente ampio, oltre ad essere in costante crescita nel tempo grazie a dispositivi sempre più potenti. Deve quindi avere tempi di risposta minimi, per poter soddisfare quanti più utenti possibili, oltre che a gestire efficientemente la memoria, dovendo distribuire gli eseguibili dei videogiochi e svariate immagini associate ad essi. Essendo un nuovo sistema non basato su uno attualmente esistente, deve poter fare una buona impressione iniziale, quindi deve avere il minor numero di bug e di falle di sicurezza per evitare di perdere la fiducia dei propri utenti. Inoltre, le funzionalità principali devono essere implementare nel più breve tempo possibile, lasciando in secondo piano le funzionalità meno importanti e allo stesso tempo permettendo la facile integrazione di idee future atte a migliorare l'esperienza di tutti i tipi di utenti. Il sistema sarà implementato come applicazione Web per poter essere facilmente accessibile da qualsiasi dispositivo, con una eventuale applicazione dedicata per i dispositivi mobile (facilmente integrabile come PWA). Tutto ciò deve essere bilanciato con un uso semplice dell'intero sistema, per poter avere alte possibilità che gli utenti continuino ad usarlo partecipando alle varie comunità dei vari videogiochi, oltre che a generare introiti acquistando videogiochi a pagamento.

Criterio	Descrizione
Tempo di risposta	Il sistema deve poter rispondere velocemente alle varie richieste degli utenti, in particolare quelle dei clienti, senza subire rallentamenti in caso di operazioni onerose quali il download di un videogioco.
Memoria	Il sistema deve gestire efficacemente la mole di dati in modo scalabile, così da garantire la possibilità di inserire quanti più videogiochi possibili e di transazioni da parte di sviluppatori e clienti. In particolare, i dati strutturati verranno gestiti tramite un database relazionale, mentre i contenuti scaricabili, principalmente gli eseguibili dei videogiochi e i vari tipi di immagini, dovranno essere salvate come asset esterni, avendo un riferimento nel database relazionale.
Disponibilità	Il sistema deve essere disponibile 24 ore su 24, 7 giorni su 7, in modo da garantire l'accesso agli utenti di qualsiasi nazionalità e permettendo il deployment sul mercato internazionale.
Sicurezza	Il sistema deve essere quanto più sicuro possibile, usando pratiche conosciute e ben testate per l'autenticazione e l'autorizzazione per i vari servizi, assieme all'uso di un servizio totalmente esterno per gestire i pagamenti.
Estensibilità	Il sistema implementerà le funzionalità ritenute fondamentali per poter entrare velocemente nel mercato, quindi deve poter essere facilmente esteso nel tempo con le funzionalità non urgenti, oltre che ad eventuali funzionalità future scoperte con il tempo, in base a come reagisce il mercato.
Adattabilità	Il sistema deve obbligatoriamente essere disponibile sui principali dispositivi mobile del momento, senza precludere la possibilità di nuovi dispositivi mobile futuri sul quale deve potersi affermare nel minor tempo possibile.
Usabilità	Essendo un nuovo tipo di sistema che si differenzia particolarmente da quelli esistenti, il sistema deve essere semplice da usare per garantire una percentuale alta di utenti registrati e che rimangono attivi all'interno della piattaforma.

1.2.2 Compromessi di progettazione

Tempo di risposta - Memoria Un tempo di risposta basso è fondamentale per garantire un sistema fluido e che renda la navigazione la più fluida possibile. Per tale motivo, si prevede di applicare la ridondanza di dati calcolati all'interno del database, così da rendere le richieste verso di esso più veloci, a discapito di un uso di memoria maggiore.

Performance - Leggibilità Il sistema deve essere veloce abbastanza per soddisfare le richieste di una utenza in costante crescita, ma la leggibilità del codice è considerata come più importante: il sistema, essendo basato su una area di mercato ancora non esplorata completamente, deve poter crescere facilmente con nuove funzionalità e nuovo personale, quindi il codice deve essere semplice da capire e da modificare.

Tempo di consegna - Funzionalità Il sistema deve essere operativo nel minor tempo possibile per soddisfare le richieste del cliente. Per tale motivo, è considerato accettabile il deployment delle sole funzionalità ritenute fondamentali, relegando l'implementazione delle funzionalità rimanenti in aggiornamenti futuri.

Tempo di consegna - Qualità Come detto precedentemente, il sistema deve essere pronto velocemente, ma ciò non deve impattare in modo eccessivo la qualità, fondamentale per questo tipo di sistema. Verrà quindi ritenuta necessaria l'implementazione corretta delle funzionalità dei clienti paganti, permettendo bug minori in quelle dedicate agli sviluppatori e lasciando per ultime quelle dedicate agli amministratori. Il motivo di ciò sta nel fatto che una utenza felice porta ulteriori utenti nella piattaforma, i quali rappresentano ciò che gli sviluppatori cercano e che per tale motivo possono essere disposti a convivere con bug non troppo importanti se possono rilasciare i propri videogiochi sulla piattaforma. Gli amministratori, invece, fanno parte del personale della piattaforma, e come tali avranno più pazienza per eventuali problemi, potendo contattare anche in modo più semplice gli sviluppatori del sistema, esprimendo le loro opinioni.

1.3 Definizioni, acronimi e abbreviazioni

- PWA: Progressive Web App, ovvero un ibrido tra le normali pagine Web e le applicazioni native mobile. Permettono una semplice implementazione di una applicazione nativa che si interfaccia ad un sito web esistente.

1.4 Riferimenti

- RAD_GameUp.pdf
- Steam `https://store.steampowered.com/`

- Android Play Store
- iOS App Store
- Laravel `https://laravel.com/`
- Laravel Sail `https://laravel.com/docs/8.x/sail`
- Docker `https://www.docker.com/`
- Docker Container `https://www.docker.com/resources/what-container`
- WSL `https://docs.microsoft.com/it-it/windows/wsl/about`

1.5 Panoramica

In questo documento verrà analizzata la struttura del sistema e dei sottosistemi relativi alla piattaforma GameUp, assieme alle condizioni limite, alla gestione degli accessi e decidendo lo stack tecnologico da utilizzare per lo sviluppo, oltre che a dettagliare i servizi offerti da ciascun sottosistema.

Capitolo 2

Architettura del Software Corrente

Attualmente non esiste un sistema con lo stesso scopo di quello proposto, bensì il sistema proposto trae spunto dalle implementazioni di sistemi simili ma con obiettivi o target diversi. I principali sistemi da cui il sistema corrente prende spunto sono:

- Steam: Uno dei primi e-commerce dedicati ai videogiochi, avente come target il mondo videoludico su computer. Esso offre dei servizi in forma web molto simili al sistema proposto, ma contiene videogiochi unicamente per computer, escludendo completamente altre piattaforme. Anche con un target ristretto, vanta un guadagno di oltre 4 miliardi di dollari annui.

- Android Play Store / iOS App Store: I punti di accesso principali sulla gran parte dei dispositivi mobile esistenti. Essi permettono il download di applicazioni compatibili per i dispositivi relativi (Dispositivi Apple per iOS App Store, Dispositivi Android di svariati produttori per Android Play Store), senza un determinato target. Da essi è possibile anche il download di videogiochi mobile, ma non è assolutamente lo scopo principale, non sono offerte funzionalità avanzate di ricerca dedicate a questo tipo di target e non esiste alcun tipo di comunità o forum all'interno di tali piattaforme.

Esistono, inoltre, servizi minori per piattaforme attualmente meno usate, come ad esempio Windows Phone, o anche la versione cinese dell'app store, dopo le recenti problematiche tra America e Cina.

Capitolo 3

Architettura del Sistema Proposto

3.1 Panoramica

Il sistema proposto è una applicazione Web, per garantire la portabilità su quanti più dispositivi possibili nel modo più semplice possibile, così da garantire l'obiettivo di design precedentemente elencato. Lo stile architetturale adattato sarà basato sullo stile architetturale three-tier poiché permette una suddivisione delle componenti essenziali dell'applicazione Web: le pagine HTML mostrate all'utente (con eventuali script) rappresenteranno le nostre interfacce, le richieste mandate dai vari tipi di utenti verranno smistate e gestite dai nostri Controller, i quali manipoleranno i dati nel modo necessario attraverso i nostri Model, i quali agiscono da interfaccia verso le nostre fonti di dati (database e file). I tipi di utenti del nostro sistema saranno di tre tipi: Clienti, Sviluppatori e Amministratori, i quali avranno accesso sia a funzionalità comuni tra di loro, sia a funzionalità mirate per i loro ruoli.

3.2 Decomposizione in Sottosistemi

Il sistema proposto seguirà fedelmente la decomposizione suggerita dallo stile architetturale three-tier:

- Interface: Questo strato gestirà la parte di presentazione, contenente le interfacce che verranno mostrate al sistema per presentare i dati richiesti e per permettere di effettuare richieste specifiche;

- Controller: Questo strato gestirà la parte di logica del sistema, ovvero principalmente il modo in cui il sistema reagisce alle richieste dei vari utenti. Esso rappresenterà il flusso di esecuzione del sistema, il quale inizia da una richiesta effettuata verso un determinato percorso smistandola ad un determinato controller;

- Storage: Questo strato gestirà i dati persistenti del nostro sistema, fornendo una interfaccia astratta sul database relazionale, il quale conterrà tutti i dati strutturati trattati dal sistema, oltre che ai dati non strutturati, ovvero immagini e file eseguibili.

Per motivi di manutenibilità e di leggibilità del codice, come specificato dagli obiettivi di design per offrire la miglior esperienza possibile di estensione delle funzionalità del sistema con il passare del tempo, l'architettura adottata sarà di tipo chiuso: Le interfacce non comunicheranno mai direttamente con i model, i vari sottosistemi comunicheranno solo con lo strato immediatamente inferiore al loro. Inoltre, sempre per gli stessi motivi, si cercherà di ridurre quanto possibile l'accoppiamento tra i vari sottosistemi, così da poter permettere modifiche dirette a sottosistemi specifici senza causare un sisma nell'intero sistema. Inoltre, si cercherà di massimizzare la coesione tra le varie componenti dei singoli sottosistemi, così da rispettare il principio di responsabilità singola, come dettato dalla filosofia Unix: ogni componente dovrebbe fare una singola cosa fatta bene.
Da una analisi degli oggetti Boundary individuati nella fase di analisi, possiamo definire tre sottosistemi dedicati alle interfacce:

- InterfacciaProfilo

- InterfacciaVideogiochi

- InterfacciaForum

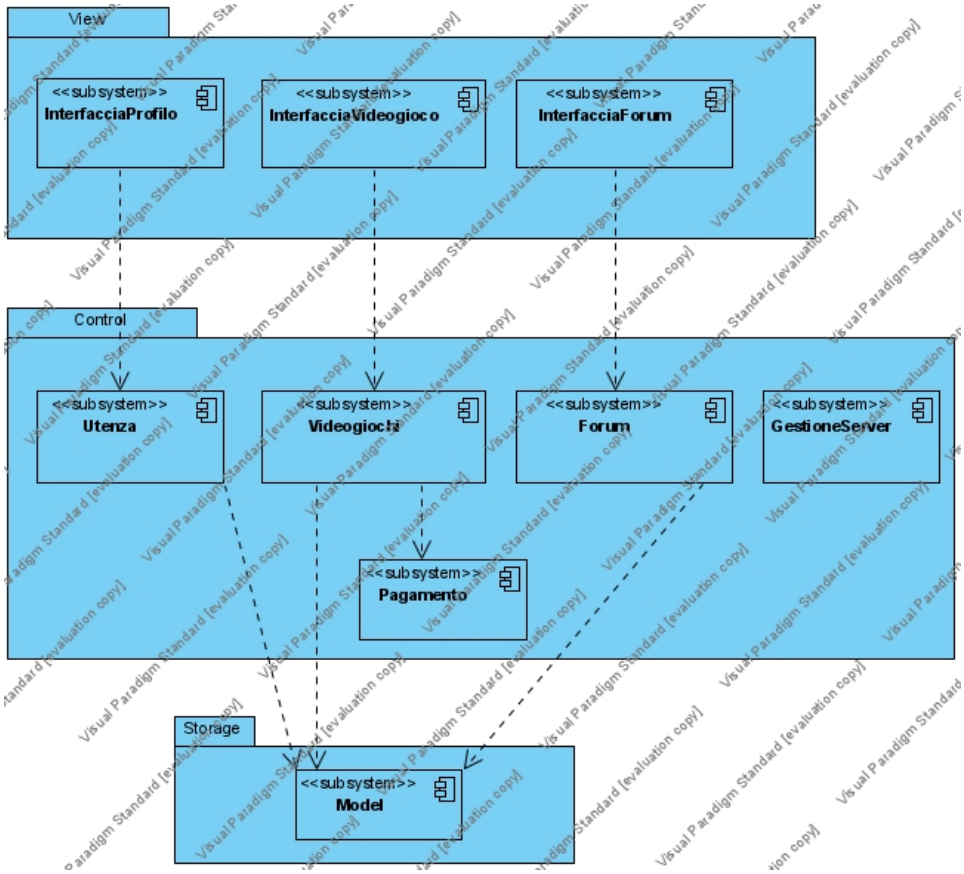
Per gli oggetti Control, possiamo definire uno schema simile individuando quattro sottosistemi:

- Utenza (gestisce la parte di autenticazione e di gestione dei profili utente)

- Videogiochi (gestisce l'intera parte di raccolta, inserimento e aggiornamento di dati relativi ai videogiochi, oltre che ad una minima parte di moderazione, ovvero le funzionalità di modifica diretta e di occultamento di videogiochi con dati ritenuti offensivi)

- Pagamento (rappresenta il sistema esterno che gestisce i pagamenti per gli acquisti dei videogiochi e quelli relativi alle sponsorizzazioni effettuate dagli sviluppatori per i propri videogiochi)
- Forum (gestisce la parte relativa agli spazi offerti ai singoli videogiochi per coltivare la propria comunità tramite l'uso di discussioni e commenti, per permettere l'interazione fra essi e fra utenti e sviluppatori)

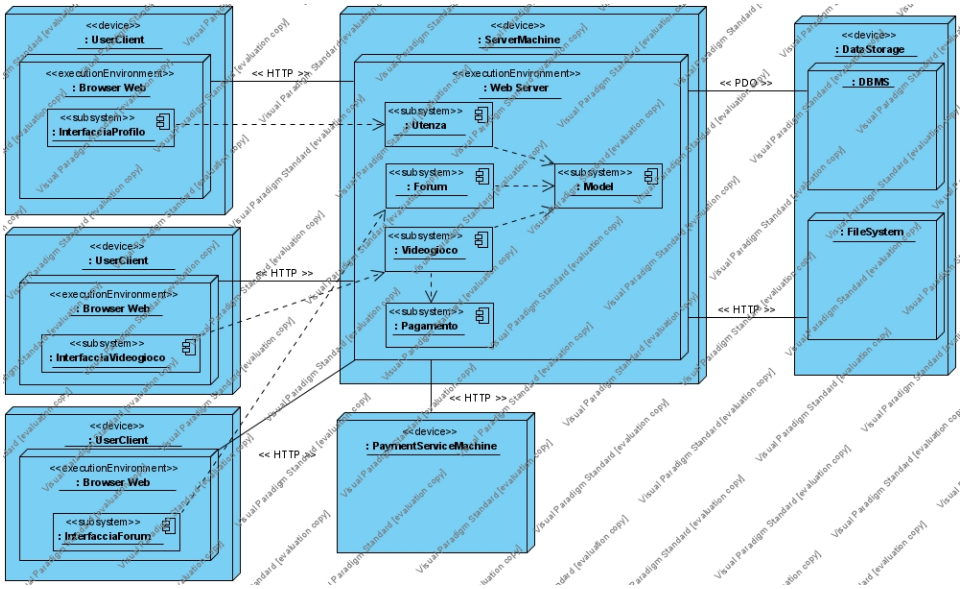
In particolare, per l'entità Validazione, la quale gestisce la validazione dei dati inseriti dagli end-users, verrà usato un servizio già incluso nel framework scelto nella fase di mapping hardware/software, quindi non verrà indicata come sottosistema. Per ultimi, dall'analisi degli oggetti Entity e Manager, definiamo un unico sottosistema che agisce da repository dell'intera mole di dati del sottosistema, Model, come astrazione verso le due fonti di dati principali (database relazionale e file). Durante l'analisi delle condizioni limite, è stato individuato inoltre un sottosistema dedicato alla gestione amministrativa del server, GestioneServer, per le operazioni di inizializzazione e terminazione del sistema.


View
<<subsystem>>
InterfacciaProfilo
<<subsystem>>
InterfacciaVideogioco
<<subsystem>>
InterfacciaForum
Control
<<subsystem>>
Utenza
<<subsystem>>
Videogiochi
<<subsystem>>
Forum
<<subsystem>>
GestioneServer
<<subsystem>>
Pagamento
Storage
<<subsystem>>
Model

3.3 Mapping Hardware/Software

L'architettura della distribuzione del sistema segue il modello client/server, dove il client è rappresentato dal Web browser in esecuzione su una macchina di un determinato utente, il quale permette l'accesso ai servizi forniti dal sistema tramite il protocollo HTTP gestendo la parte di visualizzazione delle interfacce e rendendo possibile effettuare richieste. Il server è rappresentato da una singola macchina accessibile dal Web sul quale il sistema è in esecuzione. L'astrazione della fonte dei dati permette l'esecuzione di un database locale per il deployment iniziale del sistema e per la fase di testing, oltre che a permettere una transizione semplice ad un database eventualmente localizzato su una macchina separata, dovendo solo specificare un indirizzo (ed eventuali credenziali) diverso. Il server sarà in comunicazione con il database relazionale tramite uno strato di astrazione (PDO) basato sul protocollo TCP/IP e sarà anche in comunicazione con un filesystem per la gestione di volumi di dati importanti, rappresentati dagli eseguibili dei videogiochi e dalle immagini. Inoltre, il server sarà in comunicazione con il sistema esterno che gestisce i pagamenti, usando come protocollo HTTP, scegliendo come sistema Stripe, il quale permette di pagare con la maggior parte dei circuiti bancari, offrendo una API semplice da usare. Il sistema stesso sarà basato su Laravel, un popolare framework scritto in PHP per lo sviluppo di applicazioni Web robuste, con una API ben documentata e semplice da usare, con tecniche e pattern come l'Inversion of Control, la Dependency Injection e il testing tramite mockups, assieme ad un particolare plug-in ufficialmente sponsorizzato, Cashier, per astrarre la comunicazione con il sistema Stripe.
Le interfacce mostrate sui dispositivi posseduti dagli utenti verranno implementate usando come tecnologia il sistema di templating "blade", simile alle JSP di Java EE, il quale permette la creazione di una pagina HTML con dati dinamici definiti via codice passati come argomenti al template.
Sulla macchina del server, invece, verranno implementati i Controller di Laravel, i quali permettono di ricevere una richiesta (eventualmente dopo averla passata attraverso vari middleware, pattern architetturali che trattano la richiesta trasformando eventualmente i dati contenuti o adoperando controlli sulle precondizioni da applicare per quella specifica richiesta. Tali Controller avranno un compito simile agli oggetti Control individuati in fase di analisi, ovvero di orchestrare i modelli necessari per rispondere al tipo di richiesta effettuata dall'utente, ritornando la giusta risposta che l'utente si aspetta sotto forma di file HTML, creato appunto tramite il sistema di templating precedentemente nominato.

Oltre ai Controller, verranno anche implementati dei Model, ovvero delle façade di Laravel che fanno parte dell'object relational mapper incluso in Laravel che permettono l'accesso ai dati persistenti del nostro sistema.



3.4 Gestione Dati Persistenti

Il sistema gestisce i dati persistenti in due modi diversi, a seconda del tipo di dati trattato. Tutti i dati, aventi una determinata struttura ed essendo di dimensioni non importanti, verranno resi persistenti attraverso un database relazionale basato su MySQL, per gestire la concorrenza tra i vari tipi di utenti e i modi in cui richiedono accesso (scrittura o lettura) a tali dati. In particolare, i dati rappresentanti immagini o quello che rappresenta l'eseguibile di un determinato videogioco, scaricabile da un utente previo acquisto se dovuto, verranno anch'essi rappresentati nel database relazionale, ma il dato che verrà effettivamente salvato sarà la locazione sul filesystem del contenuto di tale dato. Per questi tipi di dati non strutturati e voluminosi, quindi, si preferirà una persistenza di tipo semplice, anche perché i casi d'uso principali prevedono un singolo inserimento iniziale e conseguenti letture, negando la necessità di gestire la concorrenza per tali dati.
Le classi individuate nel class diagram che verranno rese persistenti saranno:

- Utente
- Videogioco
- VersioneVideogioco
- SponsorizzazioneVideogioco
- Report
- RecensioneVideogioco
- ValutazioneRecensione
- RichiestaVideogioco
- TagVideogioco
- Discussione
- Commento

In particolare, le specializzazioni della classe Utente non riportate come persistenti verranno condensate nella classe Utente, tramite l'aggiunta di un ulteriore attributo il quale rappresenterà il tipo dell'utente (cliente, sviluppatore o amministratore).

3.5 Controllo degli Accessi e Sicurezza

Il sistema deve permettere l'autenticazione degli utenti, in particolare per autorizzarli alle diverse funzionalità in base al loro ruolo. Ciò viene reso possibile offrendo la possibilità di registrarsi e di effettuare il login con le credenziali fornite al momento della registrazione, o eventualmente, nel caso di smarrimento della password, con l'username fornito in fase di registrazione e la password scelta durante il reset. In particolare, si assume l'esistenza a priori di determinati account aventi il ruolo di amministratore.
I dati relativi al controllo degli accessi sono memorizzati nel database relazionale, corrispondendo agli attributi della classe Utente "username", "password" e l'attributo "ruolo" aggiunto per persistere la specializzazione della classe Utente. La password verrà mantenuta rigorosamente nella sua forma hashed, ovvero verrà mantenuto l'hash ottenuto come risultato dell'applicazione della funzione bcrypt, basato sull'algoritmo di hashing Blowfish, usando una chiave privata configurata come variabile di ambiente del sistema. Ciò serve a ridurre i rischi associati ad un eventuale leak dei dati contenuti nel database: per svariati motivi, ciò può succedere, il nostro obiettivo è quello di difendere i dati sensibili dei nostri utenti così da mitigare i danni, in particolare quelli d'immagine, i quali possono essere fatali per un sistema innovativo come GameUp.
La fase di login permetterà all'utente di ottenere una sessione, la quale servirà ad autenticarlo per le richieste successive.
Le operazioni che ogni attore può effettuare sono dettagliate nella seguente lista di controllo degli accessi, scelta rispetto ad una matrice globale degli accessi per migliorare la leggibilità del codice, avendo dentro una singola classe solo le autorizzazioni relative a quella classe, e anche tenendo conto delle performance, poiché l'implementazione consisterà in una semplice mappa per ogni Controller contenente una operazione associata ad una lista di attori, quindi l'accesso alla lista degli attori avente il permesso per una determinata operazione sarà costante, mentre il controllo relativo alla lista di attori, per vedere se contiene l'attore che sta provando ad effettuare l'operazione, sarà lineare. Tale tempo è considerato trascurabile essendo la lista di attori molto piccola, quindi si adotterà questa struttura principalmente per la leggibilità del codice.
Se un utente prova ad accedere ad una operazione senza avere il ruolo richiesto per essa, verrà restituito un errore HTTP 403, basandoci sull'assunzione che l'utente non può fare nulla per poter ottenere l'accesso alla risorsa voluta, poiché il suo ruolo non glielo permette.
Nelle tabelle di seguito riportate si assume che l'attore abbia effettuato

la procedura di autenticazione per effettuare l'operazione relativa, se non specificato altrimenti.
La classe Utente non ha bisogno di un controllo degli accessi basato sul ruolo, poiché tutti gli attori possono effettuare le stesse operazioni rispetto al loro account.
La classe Videogioco, invece, offrirà operazioni diverse in base al ruolo dell'attore, come descritto dalla seguente lista:

Operazione	**Attori**
ottieniDatiVideogioco	Cliente, Sviluppatore, Amministratore (anche non autenticati)
infoEssenzialiUltimiVideogiochi	Cliente, Sviluppatore, Amministratore (anche non autenticati)
ottieniVideogiochiSponsorizzati	Cliente, Sviluppatore, Amministratore (anche non autenticati)
ottieniVideogiochiPiùScaricati	Cliente, Sviluppatore, Amministratore (anche non autenticati)
ottieniUltimiGiochiPubblicati	Cliente, Sviluppatore, Amministratore (anche non autenticati)
ottieniVideogiochiSimili	Cliente, Sviluppatore, Amministratore
aggiornaDatiVideogioco	Amministratore
pubblicaVideogioco	Amministratore
modificaVideogioco	Amministratore
ottieniLinkDownloadVersione	Cliente, Sviluppatore (se possessori del videogioco), Amministratore
nascondi	Amministratore
ottieniVideogiochiCompatibili	Cliente, Sviluppatore, Amministratore (anche non autenticati)

La classe VersioneVideogioco non ha metodi di per se, i dati sono accessibili solo tramite la classe Videogioco e la creazione di una versione deriva dall'aggiornamento o dalla pubblicazione dei dati di un videogioco.
La classe SponsorizzazioneVideogioco avrà i seguenti controlli sugli accessi:

Operazione	**Attori**
verificaDisponibilità	Sviluppatore
creaSponsorizzazione	Sviluppatore (se autore del videogioco)

La classe Report avrà i seguenti controlli sugli accessi:

Operazione	Attori
risolvi	Amministratore
creaReport	Sviluppatore (se autore del videogioco)
ottieniDettagli	Amministratore

La classe RecensioniVideogioco avrà i seguenti controlli sugli accessi:

Operazione	Attori
salvaRecensione	Cliente, Sviluppatore, Amministratore

La classe ValutazioneRecensione avrà i seguenti controlli sugli accessi:

Operazione	Attori
salvaValutazione	Cliente, Sviluppatore, Amministratore

La classe RichiestaVideogioco avrà i seguenti controlli sugli accessi:

Operazione	Attori
risolvi	Amministratore
ottieniSintesiRichieste	Amministratore
ottieniDettagli	Amministratore
nuovaRichiesta	Sviluppatore
finalizzaRichiesta	Sviluppatore

La classe TagVideogioco avrà i seguenti controlli sugli accessi:

Operazione	Attori
aggiungiSuggerimenti	Cliente, Sviluppatore, Amministratore
eliminaTags	Cliente, Sviluppatore, Amministratore

La classe Discussione avrà i seguenti controlli sugli accessi:

Operazione	Attori
chiudiDiscussione	Cliente (se autore della discussione), Sviluppatore (se autore del videogioco relativo alla discussione), Amministratore
poniInRilievo	Sviluppatore (se autore del videogioco relativo alla discussione)
creaDiscussione	Cliente (se possiede il videogioco relativo alla discussione), Sviluppatore, Amministratore
nascondi	Amministratore

La classe Commento avrà i seguenti controlli sugli accessi:

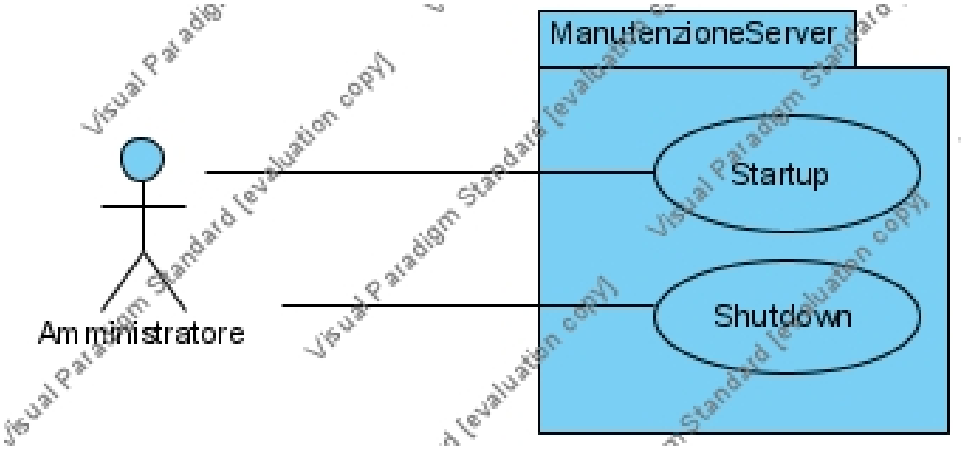
Operazione	**Attori**
creaCommento	Cliente (se possiede il videogioco relativo alla discussione), Sviluppatore, Amministratore
nascondi	Amministratore

3.6 Controllo del flusso globale del sistema

Il controllo del flusso globale del sistema è di tipo *thread-driven*, poiché il web-server metterà in una *thread pool* un insieme di thread disponibili e in attesa di gestire le richieste inviate da client multipli. Ogni richiesta verrà gestita da un singolo thread, il quale verrà liberato dopo l'invio della risposta associata al client, potendo così essere riutilizzato per richieste future.

3.7 Condizioni Limite

Per semplificare il più possibile le procedure di startup e shutdown, sfrutteremo il pacchetto Laravel Sail, il quale, basandosi sulla tecnologia dei container Docker, ci permetterà di inizializzare completamente il nostro sistema in un processo isolato tramite un singolo comando, oltre che a poterlo terminare tramite un ulteriore comando. Inoltre, tale pacchetto ci permetterà di avere una serie di container separati, contenenti il nostro database relazionale e la nostra applicazione con tutte le dipendenze necessarie per il framework Laravel.
Per la configurazione, l'unico dato necessario durante il primo avvio del sistema è un account amministratore, il quale verrà generato automaticamente durante la creazione del database relazionale con delle credenziali di default (con username uguale ad "admin" e password uguale ad "admin"), con il quale l'amministratore può eseguire l'accesso e cambiare la propria password.



L'user task ManutenzioneServer farà parte di un ulteriore sottosistema, GestioneServer.

3.7.1 Startup

<table>
<tr><td>Nome</td><td colspan="2">Startup</td></tr>
<tr><td>Attori</td><td colspan="2">Amministratore</td></tr>
<tr><td>Condizione d'ingresso</td><td colspan="2">L'amministratore è connesso tramite SSH al server ospitante gli artefatti del sistema ed il sistema non è attualmente in fase di esecuzione.</td></tr>
<tr><td rowspan="3">Flusso degli eventi</td><td>Utente</td><td>Sistema</td></tr>
<tr><td>L'amministratore esegue il comando "./vendor/bin/sail up" dalla cartella root del progetto</td><td></td></tr>
<tr><td></td><td>Docker avvia un container contenente l'applicazione Laravel e un container separato come volume contenente l'istanza del database relazionale</td></tr>
<tr><td>Condizione di uscita</td><td colspan="2">Il sistema è completamente inizializzato ed è accessibile tramite un browser Web.</td></tr>
<tr><td>Eccezioni</td><td colspan="2">N/A</td></tr>
<tr><td>Requisiti Speciali</td><td colspan="2">Nel caso il sistema operativo del server sia Windows, è necessaria l'installazione di WSL2.</td></tr>
</table>

3.7.2 Shutdown

<table>
<tr><td>Nome</td><td colspan="2">Shutdown</td></tr>
<tr><td>Attori</td><td colspan="2">Amministratore</td></tr>
<tr><td>Condizione d'ingresso</td><td colspan="2">L'amministratore è connesso tramite SSH al server ospitante gli artefatti del sistema ed il sistema è attualmente in fase di esecuzione.</td></tr>
<tr><td rowspan="3">Flusso degli eventi</td><td>Utente</td><td>Sistema</td></tr>
<tr><td>L'amministratore esegue il comando "./vendor/bin/sail down" dalla cartella root del progetto</td><td></td></tr>
<tr><td></td><td>Docker ferma tutti i container relativi al sistema</td></tr>
<tr><td>Condizione di uscita</td><td colspan="2">Il sistema non è più in esecuzione e non è più raggiungibile tramite browser Web.</td></tr>
<tr><td>Eccezioni</td><td colspan="2">N/A</td></tr>
<tr><td>Requisiti Speciali</td><td colspan="2">N/A</td></tr>
</table>

3.7.3 Fallimenti

I principali tipi di fallimento che ci si aspetta in fase di esecuzione sono due:

- Hardware: Nel caso di un fallimento di tipo hardware sul server ospitante l'applicazione Laravel, non si prevede alcuna strategia di fallback per garantire l'operabilità del sistema. Nel caso il fallimento accada sulla macchina ospitante, verranno effettuati backup settimanali dei dati del database relazionale, mentre per i file eseguibili e per le immagini non verranno adottate strategie di backup.
- Software: Nel caso di un problema con l'implementazione del sistema a runtime con una richiesta effettuata da un client, verrà mostrata una pagina generica per comunicare l'avvenuto errore all'utente. La traccia di esecuzione verrà salvata in un file log automaticamente dal framework Laravel, per effettuare il debugging in un secondo momento.

3.8 Servizi dei Sottosistemi

3.8.1 Utenza

Sottosistema	Descrizione
Utenza	Racchiude tutte le operazioni relative al proprio profilo, alla sua gestione e al suo utilizzo.

Servizio	Descrizione
login	Consente ad un utente di autenticarsi al sistema tramite l'username e la password relative al suo account salvato nella piattaforma
logout	Consente ad un utente di invalidare la propria sessione autenticata
registrazione	Consente ad un visitatore di creare un account per potersi autenticare ed effettuare operazioni sensibili
tentaRecuperoPassword	Permette ad un utente di iniziare la procedura di recupero password, ottenendo per email un link per finalizzarla
resetPassword	Permette ad un utente di finalizzare la procedura di recupero password, effettuando il reset di essa
visualizzaProfilo	Permette ad un utente di ottenere i dati associati al proprio profilo, indicati al momento di registrazione o aggiornati in seguito a tale momento
modificaProfilo	Permette ad un utente di avviare la procedura di modifica dei dati associati al proprio profilo
modificaDatiProfilo	Permette ad un utente di finalizzare la procedura di modifica dei dati associati al proprio profilo

3.8.2 Videogioco

Sottosistema	Descrizione
Videogioco	Racchiude tutte le operazioni relative ai videogiochi della piattaforma, per la creazione e la gestione di essi

Servizio	Descrizione
ottieniDatiVideogioco	Permette ad un utente di ottenere le informazioni di dettaglio di un particolare videogioco
getListaVideogiochi	Permette ad un utente di ottenere una lista di videogiochi paginata, dove per ogni videogioco vengono riportate le informazioni ritenute essenziali in fase di analisi
applicaCriteri	Permette di ottenere una lista filtrata di videogiochi paginata, con le stesse informazioni di getListaVideogiochi
ottieniVideogiochiSponsorizzati	Permette di ottenere una lista dei videogiochi classificati come sponsorizzati nella data passata come parametro (normalmente la data odierna)
ottieniVideogiochiPiùScaricati	Permette di ottenere una lista dei videogiochi più scaricati della piattaforma
ottieniUltimiGiochiPubblicati	Permette di ottenere una lista ordinata di videogiochi, in ordine da quello pubblicato più recentemente a quello meno
ottieniVideogiochiSimili	Permette di ottenere una lista ordinata di videogiochi, filtrati secondo una analisi di compatibilità con il profilo passato come argomento analizzando le tag dei giochi acquistati precedentemente dall'utente
avviaModifica	Permette di avviare la procedura di modifica diretta dei dettagli di un videogioco
aggiornaDatiVideogioco	Permette di aggiornare i dati di un videogioco immediatamente
ottieniSintesiRichieste	Permette di ottenere una lista di richieste, con i dati ritenuti essenziali in fase di analisi, di pubblicazione e modifica di un videogioco
ottieniDettagliRichiesta	Permette di ottenere i dati relativi alla richiesta specificata
risolvi	Permettere di risolvere una determinata richiesta
modificaVideogioco	Permette di avviare la procedura di creazione di una richiesta di modifica di un videogioco

Servizio	Descrizione
richiediPubblicazione	Permette la creazione di una richiesta di pubblicazione di un videogioco
finalizzaRichiesta	Permette la finalizzazione di una richiesta di pubblicazione, specificando l'eseguibile della prima versione da caricare del videogioco
iniziaSponsorizzazione	Permette di iniziare la procedura di sponsorizzazione di un videogioco
selezionaSettimana	Permette di verificare la disponibilità di una settimana, per la sponsorizzazione di un videogioco
procediPagamento	Permette la creazione di una sponsorizzazione attraverso il pagamento della somma dovuta
acquistaVideogioco	Permette l'aggiunta alla propria libreria di un determinato videogioco, previo pagamento se dovuto
downloadVideogioco	Permette il download di una determinata versione di un videogioco
avviaProceduraSuggerimentoTags	Permette di avviare la procedura per la gestione dei suggerimenti delle tags di un determinato videogioco
seleziona	Permette di selezionare una determinata tag per poterla suggerire
consiglia	Permette di suggerire effettivamente una tag per un videogioco
rimuoviSuggerimento	Permette di iniziare la procedura di rimozione di un suggerimento tag per un videogioco
confermaRimozione	Permette di rimuovere effettivamente una tag
valutaRecensione	Permette all'utente di valutare positivamente o negativamente una recensione
valutaVideogioco	Permette all'utente di lasciare una propria recensione ad un videogioco acquistato

3.8.3 Forum

Sottosistema	Descrizione
Forum	Racchiude tutte le operazioni relative agli spazi dedicati alle comunità di ogni singolo videogioco e le operazioni relative ai report di contenuti offensivi

Servizio	Descrizione
chiudiDiscussione	Permette la chiusura di una discussione, per renderla di sola lettura
creaNuovaDiscussione	Permette di avviare la procedura di creazione di una discussione
creaDiscussione	Permette la creazione di una discussione
commenta	Permette all'utente di lasciare un commento in una discussione
iniziaReport	Permette di avviare la procedura di reporting per contenuti offensivi
creaReport	Permette la creazione di un report associato ad un determinato contenuot ritenuto offensivo
poniInRilievo	Permette di porre una discussione in rilievo, mostrandola nella griglia delle discussioni di un forum prima delle discussioni non in rilievo
nascondi	Permette di nascondere un contenuto, rendendolo non più visualizzabile da utenti non amministratori
visualizzaDettagliReport	Permette di ottenere i dati relativi ad un report
risolvi	Permette di marcare un report come risolto

3.8.4 Gestione Server

Sottosistema	Descrizione
GestioneServer	Racchiude le operazioni relative alle procedure di startup e shutdown del sistema da parte di un amministratore

Servizio	Descrizione
startup	Permette l'inizializzazione del sistema, procedendo alla creazione di servizi ausiliari nel caso si tratti del primo startup
shutdown	Permette di terminare l'esecuzione del sistema in modo sicuro, senza perdite di dati

3.8.5 Pagamento

Sottosistema	Descrizione
Pagamento	Rappresenta il sistema esterno ausiliario atto alla gestione dei pagamenti per videogiochi e sponsorizzazioni

Servizio	Descrizione
avviaPagamento	Permette di pagare la somma dovuta per l'acquisto di un videogioco
procediPagamento	Permette di pagare la somma dovuta per la sponsorizzazione di un videogioco