



Università degli Studi di Salerno
Dipartimento di Informatica

GameUp

Object Design Document

Studenti
Francesco Foglia

Docente
Prof. Andrea De Lucia

Anno Accademico 2020-2021

Cronologia Revisioni

Data	Versione	Descrizione
20/01/2021	0.1	Conversione in formato TeX dell'intero documento e release iniziale
24/01/2021	1.0	Completamento interfacce e class diagram completi

Indice

1	Introduzione	1
1.1	Compromessi di Object Design	1
1.1.1	Componenti off-the-shelf	1
1.1.2	Design Patterns	2
1.1.3	Linee guida per la documentazione delle interfacce . .	3
1.1.4	Definizioni, Acronimi e Abbreviazioni	4
1.1.5	Riferimenti	5
2	Packages	6
3	Interfacce delle Classi	8
3.1	Classi Control	9
3.1.1	UtenteControl	9
3.1.2	VideogiocoControl	11
3.1.3	ForumControl	16
3.2	Classi Service	18
3.2.1	Utenza Service	18
3.2.2	Videogioco Service	20
3.2.3	Forum Service	25
3.2.4	Pagamento Service	27
3.3	Classi Repository	28
3.3.1	Utenza Repository	28
3.3.2	Videogioco Repository	29
3.3.3	Forum Repository	30
3.3.4	Class Diagrams	31

Capitolo 1

Introduzione

1.1 Compromessi di Object Design

Il compromesso principale di object design è dato dal poco tempo disponibile per lo sviluppo dell'implementazione core del sistema: per avere una base da cui partire, si sceglierà di usare componenti off-the-shelf laddove possibile, considerando anche l'assenza di un budget dedicato al sistema, sia per la parte relativa al back-end e allo sviluppo della logica di business, sia per lo sviluppo del front-end con le relative interfacce che verranno servite ai vari utenti del sistema.

1.1.1 Componenti off-the-shelf

Per la progettazione del back end del sistema useremo Laravel, un framework dell'ecosistema del linguaggio di programmazione PHP il quale offre una base potente, fluida ed estremamente comoda nella quale sviluppare una applicazione Web, assieme a svariati componenti per semplificare qualsiasi particolarità si voglia implementare o gestire, come ad esempio Laravel Sail per la gestione delle condizioni limite del sistema (startup e terminazione), Laravel Socialite per la gestione dell'autenticazione di utenti tramite OAuth e molto altro.

In particolare, per l'implementazione del sistema verrà usata l'ultima attuale versione del framework stabile, ovvero la 8.5.8. I componenti che verranno usati saranno:

- Laravel Sail, per la gestione dello startup e della terminazione del sistema;
- Eloquent, come ORM per la comunicazione con il database;

- Laravel-Admin, per l'implementazione del pannello di amministrazione;
- Laravel Blade, templating engine principale di Laravel per l'implementazione del front end, ovvero di tutte le interfacce proposte dal sistema;
- Laravel Cashier, per interfacciarsi con il servizio esterno di gestione di pagamenti Stripe.

In particolare, Eloquent è stato scelto poiché è già integrato nel framework ed è proposto come ORM principale e per motivi simili sono stati scelti anche Laravel Blade e Laravel Cashier. Invece, Laravel-Admin è stato scelto poiché il componente principale suggerito dagli sviluppatori del framework prevede un costo non irrisorio (\$99 per progetto), si è quindi preferito scegliere una alternativa *open-source* e gratis, sulla quale si è già maturata una certa esperienza tramite ulteriori progetti e lavori. Le altre componenti e il framework stesso sono *open-source* e completamente gratis, quindi i requisiti di costo sono soddisfatti.

1.1.2 Design Patterns

Service Layer Pattern

Nel framework Laravel, sorge il problema di dove collocare la logica di business, relativa ai servizi individuati durante la fase di System Design. Porre tale logica all'interno dei Controller violerebbe il principio di responsabilità singola: i Controller dovrebbero preoccuparsi solamente di ricevere una richiesta e di fornire una risposta, non di come i dati della risposta vengono ottenuti. Per la risoluzione di tale problematica, verrà applicato il Service Layer pattern, il quale includerà la logica di business del sistema, permettendo ai vari Controller di poter richiedere un particolare tipo di servizio offerto da un sottosistema senza dover comunicare direttamente, ad esempio, con i Model.

Per l'implementazione di tale pattern, verrà creato un package contenente i servizi del nostro sistema e le classi contenute in tale package potranno essere incluse nei Controller tramite la tecnica della *dependency injection*, così da ridurre quanto più possibile l'accoppiamento tra Controller e servizi, così da permettere eventualmente l'uso di mock di servizi per il testing.

Repository Pattern

Come specificato nel documento di System Design, il nostro sistema avrà fonti di dati multiple. La principale consiste nel database relazionale, il quale sarà in comunicazione con il nostro sistema attraverso Eloquent. Ma sono presenti anche due ulteriori fonti di dati:

- I file contenuti nel filesystem, ovvero immagini ed eseguibili dei videogiochi;
- Eventuali servizi esterni futuri, o anche servizi interni per operazioni di caching (es. Redis);

Per tale motivo, si ritiene opportuno astrarre l'accesso alle fonti di dati con un layer composto da implementazioni di una interfaccia Repository, la quale esporrà i metodi per le operazioni principali relative alla persistenza dei dati (creazione, aggiornamento, ricerca e cancellazione). In tal modo, verrà ridotto l'accoppiamento tra il layer dei servizi e quello della persistenza dei dati. Per tale motivo, si ritiene necessaria la creazione di semplici classi PHP per ogni classe del nostro sistema (es. Utente, Videogioco) che verrà usato come tipo di ritorno dei metodi delle nostre Repository e che saranno semplici contenitori di dati con getter, setter e costruttori che permettono l'istanziamento partendo dagli oggetti ottenuti dalle nostre fonti di dati (es. Modelli).

1.1.3 Linee guida per la documentazione delle interfacce

Agli sviluppatori viene richiesto di seguire le regole di seguito enunciate per mantenere il codice il più consistente possibile, così da mantenere alta la qualità del progetto ed eventualmente favorire l'integrazione di sviluppatori ulteriori in futuro.

Regole Globali

- Tutto il codice PHP scritto deve rispettare le regole standard di PHP, ovvero le PSR. In particolare, considerato l'uso del framework Laravel di ultima versione, è necessario l'uso delle PSR-12 (<https://www.php-fig.org/psr/psr-12/>), così da ridurre al minimo l'impatto cognitivo del progetto per un eventuale nuovo sviluppatore;
- I nomi dei file contenenti delle sottoclassi dei controller di Laravel (e le sottoclassi stesse) devono essere singolari e devono essere seguiti dalla

parola “Controller” (esempio: “VideogameController.php” contenente la classe “VideogameController”). Stessa cosa deve avvenire con i servizi e con le repository: nel primo caso, il termine da usare come suffisso sarà “Service”, nel secondo, “Repository”;

- I nomi dei file contenenti delle sottoclassi dei model di Eloquent (e le sottoclassi stesse) devono essere singolari (esempio: “Videogame.php” contenente la classe “Videogame”);
- Usare sempre oggetti *Carbon*, forniti dal framework Laravel, per rappresentare internamente date e orari;

Organizzazione dei file

- Tutti i file devono seguire la gerarchia di cartelle proposta da Laravel, in particolare Controllers, Models e Views.

1.1.4 Definizioni, Acronimi e Abbreviazioni

- Off-the-shelf: Componenti oppure framework pronti all’utilizzo per risolvere una determinata problematica;
- Back end: La parte del sistema contenente la logica applicativa;
- Front end: La parte del sistema contenente le interfacce proposte all’utente per interagire con il sistema;
- Laravel: Web Application Framework per la creazione di sistemi sul Web;
- Laravel Sail: Componente di Laravel per la gestione dello startup e della terminazione del sistema;
- ORM: Object relational mapping, una tecnica che permette di interfacciarsi ad un sistema software tramite il paradigma della programmazione orientata agli oggetti;
- Eloquent: ORM principale del framework Laravel;
- Laravel-Admin: Componente di Laravel per l’implementazione di pannelli di amministrazione;
- Laravel Cashier: Componente di Laravel per gestire la comunicazione con servizi di pagamento;

- Stripe: Infrastruttura di pagamenti online;
- PSR: PHP Standard Recommendations;

1.1.5 Riferimenti

- Laravel: <https://laravel.com/>
- Laravel Sail: <https://laravel.com/docs/8.x/sail>
- Eloquent: <https://laravel.com/docs/8.x/eloquent>
- Laravel-Admin: <https://github.com/z-song/laravel-admin>
- Laravel Cashier: <https://laravel.com/docs/8.x/billing>
- Stripe: <https://stripe.com/it>
- PSR-12: <https://www.php-fig.org/psr/psr-12/>

Capitolo 2

Packages

La divisione del sistema in packages verrà realizzata tramite una esatta gerarchia del filesystem del progetto. La struttura di base usata è quella proposta dal framework Laravel, dove in particolare:

- La cartella *app* conterrà tutto il codice per gestire le richieste dei clienti, dall'arrivo della richiesta ad un controller fino all'invio di una risposta;
- La cartella *database* conterrà il codice relativo alla generazione della nostra fonte di dati relazionale, ovvero le migrazioni contenenti gli schemi delle nostre tabelle (così da avere una traccia dei cambiamenti applicati nel tempo al database) ed eventuali classi seeder per il riempimento di dati mock per il testing;
- La cartella *resources* conterrà tutta la parte di front end del nostro sistema, categorizzato in tre sottocartelle:
 - *css* per i fogli di stile;
 - *js* per il codice JavaScript client-side;
 - *views* per i documenti rappresentanti le interfacce proposte agli utenti;

In particolare, la cartella *app* sarà quella dove verrà collocata la gran parte del nostro sistema. La sua struttura è come segue:

- *Exceptions*, contenente le definizioni delle eccezioni lanciate dal nostro sistema;
- *Http*, contenente una cartella *Controllers* dove verranno realizzati i nostri Controller per ogni pagina del sistema, assieme ad una cartella *Middleware* per eventuali middleware customizzati;

- Models, contenente i nostri modelli Eloquent per interfacciarsi con il database relazionale;

Inoltre, verranno create due cartelle all'interno della cartella *app* per i layer ulteriori individuati durante la fase di individuazione di design patterns:

- Services, contenente i servizi offerti dal nostro sistema, raggruppati in cartelle che rappresentano i sottosistemi ai quali fanno parte;
- Repositories, contenente l'interfaccia Repository e le repositories per l'accesso ai dati persistenti gestiti dal sistema;

Capitolo 3

Interfacce delle Classi

Le classi che comporranno il nostro back end saranno di cinque tipi: Model, Controller, Services, Repositories e le classi PHP rappresentanti gli oggetti trattati dalla logica di business. I model e gli oggetti della logica di business non avranno bisogno di una descrizione della loro interfaccia, poiché non avranno implementazioni relative alla logica di business: il loro obiettivo è di fornire un punto di accesso tramite il paradigma ad oggetti al database relazionale per i primi, mentre i secondi servono come rappresentazione orientata ad oggetti dei dati trattati dal sistema; i servizi useranno questi dati per costruire risposte da fornire ai controller. Il front end sarà composto semplicemente da dei template riempiti con i dati forniti dai controller. Il flusso di esecuzione generale sarà sempre composto da un oggetto Control che chiama un metodo di un oggetto Service per eseguire un servizio, senza mai accedere direttamente agli oggetti Repository; gli oggetti Service accederanno agli oggetti Repository necessari per ottenere i dati necessari; gli oggetti Repository comunicheranno con gli oggetti per l'accesso dei dati, come ad esempio i Model di Laravel.

3.1 Classi Control

3.1.1 UtenteControl

Nome	UtenteControl
Descrizione	Questo control riceve le richieste relative al sottosistema Utente, invocando i servizi necessari per eseguire operazioni di autenticazione e di gestione del profilo.
Attributi	- utenzaService: UtenzaService
Firme Metodi	+ login(Request request): RedirectResponse + logout(): RedirectResponse + registrazione(): View + effettuaRegistrazione(Request request): RedirectResponse + visualizzaProfilo(): View + modificaProfilo(): View + modificaDatiProfilo(Request request): RedirectResponse + getAvatar(): Response
Pre-condizioni	<ul style="list-style-type: none">• context UtenteControl::login(request) pre: request.has(['username', 'password']) and !utenzaService.isAuthenticated()• context UtenteControl::logout() pre: utenzaService.isAuthenticated()• context UtenteControl::effettuaRegistrazione(request) pre: request.has(['username', 'email', 'password', 'confermaPassword'])• context UtenteControl::visualizzaProfilo() pre: utenzaService.isAuthenticated()• context UtenteControl::modificaProfilo() pre: utenzaService.isAuthenticated()• context UtenteControl::modificaDatiProfilo(request) pre: request.hasAny(['username', 'email', 'is-Sviluppatore', 'avatar', 'nuovaPassword']) and utenzaService.isAuthenticated() and request.has('passwordAttuale')• context UtenteControl::getAvatar() pre: utenzaService.isAuthenticated()

UtenteControl

Post-condizioni	<ul style="list-style-type: none"> • context UtenteControl::login(request) post: utenzaService.isAuthenticated() • context UtenteControl::logout() post: !utenzaService.isAuthenticated() • context UtenteControl::registrazione(request) post: utenzaService.isAuthenticated() and utenzaService.usernameExists(request.input('username')) and utenzaService.emailExists(request.input('email')) • context UtenteControl::modificaDatiProfilo(request) post: utenzaService.getUtenteAutenticato().username == request.input('username') and utenzaService.getUtenteAutenticato().email == request.input('email') and utenzaService.getUtenteAutenticato().isSviluppatore == request.input('isSviluppatore') and utenzaService.getUtenteAutenticato().avatar == request.input('avatar')
Invarianti	<ul style="list-style-type: none"> • context UtenteControl inv: utenzaService <> null

3.1.2 VideogiocoControl

Nome	VideogiocoControl
Descrizione	Questo control riceve le richieste relative al sottosistema Videogioco, invocando i servizi necessari per eseguire tutte le operazioni relative ai videogiochi contenuti nel sistema.
Attributi	<ul style="list-style-type: none"> - videogiocoService: VideogiocoService - utenzaService: UtenzaService
Firme Metodi	<ul style="list-style-type: none"> + ottieniDatiVideogioco(Request request): View + paginaIniziale(): View + catalogo(Request request): View + getLogo(int idVideogioco): Response + avviaModifica(int idVideogioco): View + aggiornaDatiVideogioco(Request request): RedirectResponse + visualizzaRichieste(): Response + visualizzaDettagliRichiesta(int idRichiesta): View + risolviRichiesta(Request request): RedirectResponse + richiediModificaVideogioco(): View + modificaDatiVideogioco(Request request): RedirectResponse + avviaPubblicazioneVideogioco(): View + richiediPubblicazioneVideogioco(Request request): RedirectResponse + iniziaSponsorizzazione(): View + verificaDisponibilitàSettimana(Request request): JsonResponse + procediPagamentoSponsorizzazione(): Response + acquistaVideogioco(Request request): Response + downloadVideogioco(Request request): Response + avviaProceduraSuggerimentoTags(): View + suggerisciTags(Request request): JsonResponse + rimuoviSuggerimento(Request request): JsonResponse + salvaValutazione(Request request): JsonResponse + salvaRecensione(Request request): JsonResponse

VideogiocoControl

Pre-condizioni	<ul style="list-style-type: none"> • context VideogiocoControl::ottieniDatiVideogioco(request) pre: request.has('idVideogioco') • context VideogiocoControl::getLogo(idVideogioco) pre: idVideogioco <> null • context VideogiocoControl::avviaModifica(idVideogioco) pre: request.has('idVideogioco') • context VideogiocoControl::aggiornaDatiVideogioco(request) pre: utenzaservice.isAuthenticated() and utenzaService.isAdmin() and request.has('idVideogioco') and request.hasAny(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo']) • context VideogiocoControl::visualizzaRichieste() pre: utenzaService.isAdmin() • context VideogiocoControl::visualizzaDettagliRichiesta(idRichiesta) pre: utenzaService.isAdmin() and idRichiesta <> null • context VideogiocoControl::risolviRichiesta(request) pre: utenzaservice.isAuthenticated() and utenzaService.isAdmin() and request.has(['idRichiesta', 'esito', 'commento']) • context VideogiocoControl::modificaDatiVideogioco(request) pre: utenzaService.isSviluppatore() and request.has('idVideogioco') and request.hasAny(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo']) • context VideogiocoControl::avviaPubblicazioneVideogioco(request) pre: utenzaService.isSviluppatore() • context VideogiocoControl::richiediPubblicazioneVideogioco(request) pre: utenzaService.isSviluppatore() and request.has(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo', 'eseguibile']) • context VideogiocoControl::iniziaSponsorizzazione() pre: utenzaService.isSviluppatore() • context VideogiocoControl::verificaDisponibilitàSettimana(request) pre: utenzaService.isSviluppatore() and request.has('settimana') • context VideogiocoControl::procediPagamentoSponsorizzazione(request) pre: utenzaService.isSviluppatore() and request.has(['idVideogioco', 'settimane']) and videogiocoService.settimaneDisponibili(request.input('settimane')) • context VideogiocoControl::acquistaVideogioco(request) pre: utenzaService.isAuthenticated() and request.has('idVideogioco')
----------------	---

VideogiocoControl

Pre-condizioni	<ul style="list-style-type: none"> • context VideogiocoControl::downloadVideogioco(request) pre: utenzaService.isAuthenticated() and request.hasAll(['idVideogioco', 'versione']) and videogiocoService.getVideogiochiAcquistati()-> includes(videogiocoService .getVideogioco(request.input('idVideogioco'))) • context VideogiocoControl::avviaProceduraSuggerimentoTags() pre: utenzaService.isAuthenticated() and request.has('idVideogioco') and videogiocoService.getVideogiochiAcquistati()-> includes(videogiocoService .getVideogioco(request.input('idVideogioco'))) • context VideogiocoControl::suggerisciTags(request) pre: utenzaService.isAuthenticated() and request.hasAll(['idVideogioco', 'tags']) and videogiocoService.getVideogiochiAcquistati()->includes(videogiocoService .getVideogioco(request.input('idVideogioco'))) • context VideogiocoControl::rimuoviSuggerimento(request) pre: utenzaService.isAuthenticated() and request.hasAll(['idVideogioco', 'tags']) and videogiocoService.getTagsSuggerite(request.input('idVideogioco')) <> 0 • context VideogiocoControl::salvaValutazione(request) pre: utenzaService.isAuthenticated() and request.has('idRecensione') and videogiocoService.getValutazioneRecensione(request.input('idRecensione')) == null • context VideogiocoControl::salvaRecensione(request) pre: utenzaService.isAuthenticated() and request.has('idVideogioco') and videogiocoService.getRecensione(request.input('idVideogioco')) == null
----------------	--

VideogiocoControl

Post-condizioni	<ul style="list-style-type: none"> • context VideogiocoControl::aggiornaDatiVideogioco(request) post: request.all()->forAll(k: string, v: string videogiocoService.getVideogioco(request.input('id'))[k] == v) • context VideogiocoControl::risolviRichiesta(request) post: videogiocoService.getRichiesta(request.input('idRichiesta')).esito == request.input('esito') and videogiocoService.getRichiesta(request.input('idRichiesta')).commento == request.input('commento') • context VideogiocoControl::modificaDatiVideogioco(request) post: videogiocoService.getVideogioco(request.input('idVideogioco')).getRichieste()->size == videogiocoService.getVideogioco(request.input('idVideogioco')).@pre.getRichieste()->size + 1 • context VideogiocoControl ::richiediPubblicazioneVideogioco(request) post: utenzaService.getRichieste(utenzaService.getUtenzaAutenticato())->size == utenzaService@pre.getRichieste(utenzaService.getUtenzaAutenticato())->size + 1 • context VideogiocoControl::suggerisciTags(request) post: videogiocoService.getTags(request.input('idVideogioco'))->intersection(request.input('tags')) == request.input('tags')
-----------------	---

VideogiocoControl

Post-condizioni	<ul style="list-style-type: none"> • context VideogiocoControl::rimuoviSuggerimento(request) post: videogiocoService.getTagsSuggeste(request.input('idVideogioco'))->intersection(request.input('tags'))->size == 0 • context VideogiocoControl::salvaRecensione(request) post: videogiocoService.getRecensione(request.input('idVideogioco')) <> null
Invarianti	<ul style="list-style-type: none"> • context VideogiocoControl inv: videogiocoService <> null and utenzaService <> null

3.1.3 ForumControl

Nome	ForumControl
Descrizione	Questo control riceve le richieste relative al sottosistema Forum, invocando i servizi necessari per eseguire tutte le operazioni relative alle discussioni e ai commenti contenuti nel sistema.
Attributi	<ul style="list-style-type: none"> - videogiocoService: VideogiocoService - utenzaService: UtenzaService - forumService: ForumService
Firme Metodi	<ul style="list-style-type: none"> + chiudiDiscussione(Request request): RedirectResponse + creaNuovaDiscussione(Request request): View + creaDiscussione(Request request): RedirectResponse + commenta(Request request): JsonResponse + poniInRilievo(Request request): JsonResponse
Pre-condizioni	<ul style="list-style-type: none"> • context ForumControl::chiudiDiscussione(request) pre: utenzaservice.isAuthenticated() and request.has('idDiscussione') and forumService .checkPermessiDiscussione(request.input('idDiscussione')) and forumService .getDiscussione(request.input('idDiscussione')).chiusa == false • context ForumControl::creaNuovaDiscussione(request) pre: utenzaservice.isAuthenticated() and request.has('idVideogioco') and videogiocoService.getVideogiochiAcquistati() ->includes(videogiocoService.getVideogioco(request.input('idVideogioco'))) • context ForumControl::creaDiscussione(request) pre: utenzaservice.isAuthenticated() and request.hasAll(['idVideogioco', 'titolo', 'corpo']) and videogiocoService.getVideogiochiAcquistati() ->includes(videogiocoService.getVideogioco(request.input('idVideogioco')))

ForumControl

Pre-condizioni	<ul style="list-style-type: none"> • context ForumControl::commenta(request) pre: utenzaservice.isAuthenticated() and request.hasAll(['idDiscussione', 'corpo']) and videogiocoService.getVideogiochiAcquistati() ->includes(videogiocoService.getVideogioco(request.input('idVideogioco'))) • context ForumControl::poniInRilievo(request) pre: utenzaservice.isAuthenticated() and request.has('idDiscussione') and forumService.checkPermessiDiscussione(request.input('idDiscussione')) and forumService.getDiscussione(request.input('idDiscussione')).in_rilievo == false
Post-condizioni	<ul style="list-style-type: none"> • context ForumControl::chiudiDiscussione(request) post: forumService.getDiscussione(request.input('idDiscussione')).chiusa == true • context ForumControl::creaDiscussione(request) post: forumService.getNumDiscussioni(request.input('idVideogioco')) == forumService@pre.getNumDiscussioni(request.input('idVideogioco')) + 1 • context ForumControl::commenta(request) post: forumService.getNumCommenti(request.input('idDiscussione')) == forumService@pre.getNumCommenti(request.input('idDiscussione')) + 1 • context ForumControl::poniInRilievo(request) post: forumService.getDiscussione(request.input('idDiscussione')).in_rilievo == true
Invarianti	<ul style="list-style-type: none"> • context ForumControl inv: forumService <> null and videogiocoService <> null and utenzaService <> null

3.2 Classi Service

3.2.1 Utanza Service

Nome	UtenzaService
Descrizione	Questo service rende disponibili tutte le funzionalità relative all'autenticazione, all'autorizzazione e quelle relative ai singoli utenti come la gestione del proprio profilo.
Attributi	- utanzaRepository: UtenzaRepository
Firme Metodi	+ login(string username, string password): boolean + logout(): void + registraUtente(string username, string password, string email, UploadedFile avatar = null): boolean + getUtenteAutenticato(): ?Utenza + modificaDatiProfilo(string passwordAttuale, string username, string nuovaPassword, string email, boolean isSviluppatore, UploadedFile avatar) + usernameExists(string username): boolean + emailExists(string email): boolean + isAdmin(): boolean + isSviluppatore(): boolean + getAvatar(): string + getRichieste(): array
Pre-condizioni	<ul style="list-style-type: none">• context UtenzaService::login(username, password) pre: utanzaRepository.allUsers()-> exists(u: Utente u.username == username and u.password == Hash::make(password))• context UtenzaService::registraUtente(username, password, email, avatar) pre: !usernameExists(username) and !emailExists(email)• context UtenzaService::modificaDatiProfilo(passwordAttuale, username, email, avatar, isSviluppatore) pre: utanzaRepository.checkPassword(Auth::id(), passwordAttuale) and !emailExists(email) and !usernameExists(username)• context UtenzaService::getRichieste() pre: isSviluppatore()

Utenza Service

Post-condizioni	<ul style="list-style-type: none"> • context UtenzaService::login(username, password) post: getUtenzaAutenticato().username == username • context UtenzaService::logout() post: getUtenzaAutenticato() == null • context UtenzaService::registraUtenza(passwordAttuale, username, password, email, avatar) post: getUtenzaAutenticato().username == username and getUtenzaAutenticato().password == Hash::make(password) and getUtenzaAutenticato().email == email and Hash::make(getUtenzaAutenticato().avatar) == Hash::make(avatar) • context UtenzaService::modificaDatiProfilo(username, email, avatar, isSviluppatore) post: (username == null or getUtenzaAutenticato().username == username) and (email == null or getUtenzaAutenticato().email == email) and (avatar == null or Hash::make(getUtenzaAutenticato().avatar) == Hash::make(avatar)) and (isSviluppatore == null or ((isSviluppatore() and isSviluppatore) or (!isSviluppatore() and !isSviluppatore)))
Invarianti	<ul style="list-style-type: none"> • context UtenzaService inv: utenzaRepository <> null

3.2.2 Videogioco Service

Nome	VideogiocoService
Descrizione	Questo service rende disponibili tutte le funzionalità relative alla gestione dei videogiochi e oggetti correlati, come le sponsorizzazioni, le richieste e le versioni di un videogioco.
Attributi	<ul style="list-style-type: none"> - videogiocoRepository: VideogiocoRepository - utenzaService: UtenzaService - pagamentoService: PagamentoService
Firme Metodi	<ul style="list-style-type: none"> + ottieniDatiVideogioco(int idVideogioco): ?Videogioco + applicaCriteri(string titolo = null, float prezzo = null, array tagsObbligatorie = null, array tagsOpzionali = null, boolean acquistati = false, string ordine = 'DESC'): Collection + ottieniVideogiochiSponsorizzati(Carbon data = null): Collection + ottieniVideogiochiPiùScaricati(): Collection + ottieniUltimiVideogiochiPubblicati(): Collection + ottieniVideogiochiSimili(): Collection + aggiornaDatiVideogioco(int idVideogioco, UploadedFile logo = null, string titolo = null, array immagini = null, string descrizione = null, float prezzo = null): void + ottieniSintesiRichieste(): array + ottieniDettagliRichiesta(int idRichiesta): Richiesta + risolviRichiesta(int idRichiesta, boolean esito, string commento): void + modificaDatiVideogioco(int idVideogioco, UploadedFile logo = null, string titolo = null, array immagini = null, string descrizione = null, float prezzo = null): void + richiediPubblicazioneVideogioco(UploadedFile logo, string titolo, array immagini, string descrizione, float prezzo, UploadedFile eseguibile): void + verificaDisponibilitàSettimana(Carbon settimana): boolean + procediPagamentoSponsorizzazione(int idVideogioco, array settimane): void + acquistaVideogioco(int idVideogioco): void + getEseguibileVideogioco(int idVideogioco, string versione): File + suggerisciTags(int idVideogioco, array tags): void

Videogioco Service

Firme Metodi	<ul style="list-style-type: none"> + rimuoviSuggerimento(int idVideogioco, array tags): void + salvaValutazione(int idRecensione, boolean giudizio): void + getTagsSuggerite(int idVideogioco): array + getValutazioneRecensione(int idRecensione): Valutazione-Recensione + salvaRecensione(int idVideogioco, boolean giudizio, string commento): void + getRecensione(int idVideogioco): Recensione
Pre-condizioni	<ul style="list-style-type: none"> • context VideogiocoService::applicaCriteri(titolo, prezzo, tagsObbligatorie, tagsOpzionali, acquistati, ordine) pre: (titolo <> null or prezzo <> null or tagsObbligatorie <> null or tagsOpzionali <> null) and (ordine == 'DESC' or ordine == 'ASC') • context VideogiocoService::ottieniVideogiochiSponsorizzati(data) pre: data <> null • context VideogiocoService::ottieniVideogiochiSimili() pre: utenzaService.isAuthenticated() • context VideogiocoService::aggiornaDatiVideogioco(idVideogioco, logo, titolo, immagini, descrizione, prezzo) pre: videogiocoRepository.getVideogioco(idVideogioco) <> null and (logo <> null or titolo <> null or immagini <> null or immagini->size <> 0 or descrizione <> null or prezzo <> null) • context VideogiocoService::ottieniDettagliRichiesta(idRichiesta) pre: videogiocoRepository.getRichiesta(idRichiesta) <> null

Videogioco Service

Pre-condizioni	<ul style="list-style-type: none"> • context VideogiocoService::risolviRichiesta(idRichiesta, esito, commento) pre: videogiocoRepository.getRichiesta(idRichiesta) <> null and utenzaService.isAdmin() and esito <> null • context VideogiocoService::modificaDatiVideogioco(idVideogioco, logo, titolo, immagini, descrizione, prezzo) pre: videogiocoRepository.getVideogioco(idVideogioco) <> null and (logo <> null or titolo <> null or immagini <> null or immagini->size > 0 or descrizione <> null or prezzo <> null) • context VideogiocoService::richiediPubblicazioneVideogioco(logo, titolo, immagini, descrizione, prezzo, eseguibile) pre: utenzaService.isSviluppatore() • context VideogiocoService::procediPagamentoSponsorizzazione(idVideogioco, settimane) pre: videogiocoRepository.getVideogioco(idVideogioco) <> null and settimane->size <> 0 • context VideogiocoService::acquistaVideogioco(idVideogioco) pre: videogiocoRepository.getVideogioco(idVideogioco) <> null • context VideogiocoService::getEseguibileVideogioco(idVideogioco, versione) pre: videogiocoRepository.getVideogioco(idVideogioco) <> null and videogiocoRepository.getVersioniVideogioco(idVideogioco)->includes(versione) • context VideogiocoService::suggerisciTags(idVideogioco, tags) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null and videogiocoRepository.getCompratori(idVideogioco)->includes(utenzaService.getUtenteAutenticato()) and tags->size <> 0
----------------	--

Videogioco Service

Pre-condizioni	<ul style="list-style-type: none"> context VideogiocoService::rimuoviSuggerimenti(idVideogioco, tags) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null and videogiocoRepository.getCompratori(idVideogioco)->includes(utenzaService.getUtenteAutenticato()) and tags <> null and tags->size <> 0 context VideogiocoService::salvaValutazione(idRecensione, giudizio) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null and videogiocoRepository.getCompratori(idVideogioco)->includes(utenzaService.getUtenteAutenticato()) and tags <> null and tags->size <> 0 and giudizio <> null context VideogiocoService::getTagsSuggeste(idVideogioco) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null context VideogiocoService::getValutazioneRecensione(idRecensione) pre: utenzaService.isAuthenticated() and videogiocoRepository.getRecensione(idRecensione) <> null context VideogiocoService::salvaRecensione(idVideogioco, giudizio, commento) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null context VideogiocoService::getRecensione(idVideogioco) pre: utenzaService.isAuthenticated() and videogiocoRepository.getVideogioco(idVideogioco) <> null
----------------	---

Videogioco Service

Post-condizioni	<ul style="list-style-type: none"> • context VideogiocoService::ottieniVideogiochiPiùScaricati() post: result->sortedBy(videogioco -videogioco.acquisti->size) • context VideogiocoService ::ottieniUltimiVideogiochiPubblicati() post: result->sortedBy(videogioco -videogioco.created_at.timestamp) • context VideogiocoService::ottieniSintesiRichieste() post: result->forAll(richiesta richiesta.titolo <> null and richiesta.descrizione <> null and richiesta.tipo <> null) • context VideogiocoService::risolviRichiesta(idRichiesta, esito, commento) post: videogiocoRepository.getRichiesta(idRichiesta).esito == esito and videogiocoRepository.getRichiesta(idRichiesta).commento == commento • context VideogiocoService ::verificaDisponibilitàSettimana(settimana) post: videogiocoRepository.getSponsorizzazioni ->forAll(sponsorizzazione sponsorizzazione.data_inizio.startOfWeek() <> settimana.startOfWeek() and sponsorizzazione.data_fine.endOfWeek() <> settimana.endOfWeek()) and result == true • context VideogiocoService::suggerisciTags(idVideogioco, tags) post: videogiocoRepository .getTagsVideogioco(idVideogioco)->intersection(tags) == tags • context VideogiocoService::rimuoviSuggerimenti(idVideogioco, tags) post: videogiocoRepository .getTagsVideogioco(idVideogioco)->intersection(tags) ->size == 0
Invarianti	<ul style="list-style-type: none"> • context VideogiocoService inv: videogiocoRepository <> null and utenzaService <> null and pagamentoService <> null

3.2.3 Forum Service

Nome	ForumService
Descrizione	Questo service rende disponibili tutte le funzionalità relative alla gestione dei forum relativi ai singoli videogiochi, con le relative discussioni e commenti associati.
Attributi	<ul style="list-style-type: none"> - forumRepository: ForumRepository - videogiocoService: VideogiocoService - utenzaService: UtenzaService
Firme Metodi	<ul style="list-style-type: none"> + chiudiDiscussione(int idDiscussione): void + creaDiscussione(int videogiocoId, string titolo, string corpo): Discussione + commenta(int idDiscussione, string corpo): Commento + poniInRilievo(int idDiscussione): void
Pre-condizioni	<ul style="list-style-type: none"> • context ForumService::chiudiDiscussione(idDiscussione) pre: utenzaService.isAuthenticated() and forumRepository.getDiscussione(idDiscussione) <> null and videogiocoService. .ottieniDatiVideogioco(forumRepository.getDiscussione(idDiscussione).idVideogioco).autore == utenzaService.getUtenzaAutenticato() • context ForumService::creaDiscussione(idVideogioco, titolo, corpo) pre: videogiocoService. .ottieniDatiVideogioco(idVideogioco) <> null and videogiocoRepository. .getCompratori(idVideogioco)->includes(utenzaService.getUtenzaAutenticato()) • context ForumService::commenta(idDiscussione, corpo) pre: forumRepository.getDiscussione(idDiscussione) <> null and videogiocoRepository. .getCompratori(forumRepository.getDiscussione(idDiscussione).idVideogioco)->includes(utenzaService.getUtenzaAutenticato()) • context ForumService::poniInRilievo(idDiscussione) pre: forumRepository.getDiscussione(idDiscussione) <> null and forumRepository.getDiscussione(idDiscussione).in_rilievo == false

Forum Service

Post-condizioni	<ul style="list-style-type: none"> • context ForumService::chiudiDiscussione(idDiscussione) post: forumRepository. getDiscussione(idDiscussione).chiusa == true • context ForumService::commenta(idDiscussione, corpo) post: forumRepository.getCommenti(idDiscussione)->size == forumRepository @pre.getCommenti(idDiscussione)->size + 1 • context ForumService::poniInRilievo(idDiscussione) post: forumRepository .getDiscussione(idDiscussione).in_rilievo == true
Invarianti	<ul style="list-style-type: none"> • context ForumService inv: forumRepository <> null and videogiocoService <> null and utenzaService <> null

3.2.4 Pagamento Service

Nome	PagamentoService
Descrizione	Questo service gestisce i pagamenti relativi all'acquisto di videogiochi e di sponsorizzazioni, esponendo un metodo ciascuno per astrarre la comunicazione con il provider esterno del servizio di pagamento online.
Attributi	- videogiocoService: VideogiocoService
Firme Metodi	+ avviaPagamento(int idVideogioco): void + procediPagamento(int idVideogioco): void + callbackStripeAcquisto(int idUtente, int idVideogioco): void + callbackStripeSponsorizzazione(int idUtente, int idVideogioco): void
Pre-condizioni	<ul style="list-style-type: none"> • context PagamentoService::avviaPagamento(idVideogioco) pre: videogiocoService.getVideogioco(idVideogioco) <> null • context PagamentoService::procediPagamento(idVideogioco) pre: videogiocoService.getVideogioco(idVideogioco) <> null
Post-condizioni	<ul style="list-style-type: none"> • context PagamentoService::callbackStripeAcquisto(idUtente, idVideogioco) post: videogiocoService .getVideogiochiAcquistati(idUtente)->size == videogiocoService @pre.getVideogiochiAcquistati(idUtente)->size + 1 • context PagamentoService::callbackStripeSponsorizzazione(idVideogioco) post: videogiocoService .getSponsorizzazioni(idVideogioco)->size == videogiocoService @pre.getSponsorizzazioni(idVideogioco)->size + 1
Invarianti	<ul style="list-style-type: none"> • context PagamentoService inv: videogiocoService <> null

3.3 Classi Repository

3.3.1 Utenza Repository

Nome	UtenzaRepository
Descrizione	Questa repository rappresenta il punto di accesso principale per la classe Utente, come astrazione dei modelli Eloquent forniti dal framework Laravel. In particolare, questa classe non ritorna mai l'hash della password degli utenti agli strati superiori.
Attributi	<ul style="list-style-type: none">- N/A
Firme Metodi	<ul style="list-style-type: none">+ allUsers(): array+ getUtente(int idUtente): ?Utenza+ findUsername(string username): ?Utenza+ findEmail(string email): ?Utenza+ createUtente(string username, string password, string email, UploadedFile avatar = null): Authenticatable+ modificaDatiProfilo(int idUtente, string username = null, string nuovaPassword = null, string email = null, boolean isSviluppatore = null, UploadedFile avatar = null): boolean+ checkPassword(int idUtente, string password): boolean
Pre-condizioni	<ul style="list-style-type: none">• context UtenzaRepository::modificaDatiProfilo(idUtente, username, password, email, isSviluppatore, avatar) pre: User::find(idUtente) <> null
Post-condizioni	<ul style="list-style-type: none">• context UtenzaRepository::getUtente(idUtente) post: result.id == idUtente• context UtenzaRepository::findUsername(username) post: (result == null) or (result.username == username)• context UtenzaRepository::findEmail(email) post: (result == null) or (result.email == email)
Invarianti	<ul style="list-style-type: none">• N/A

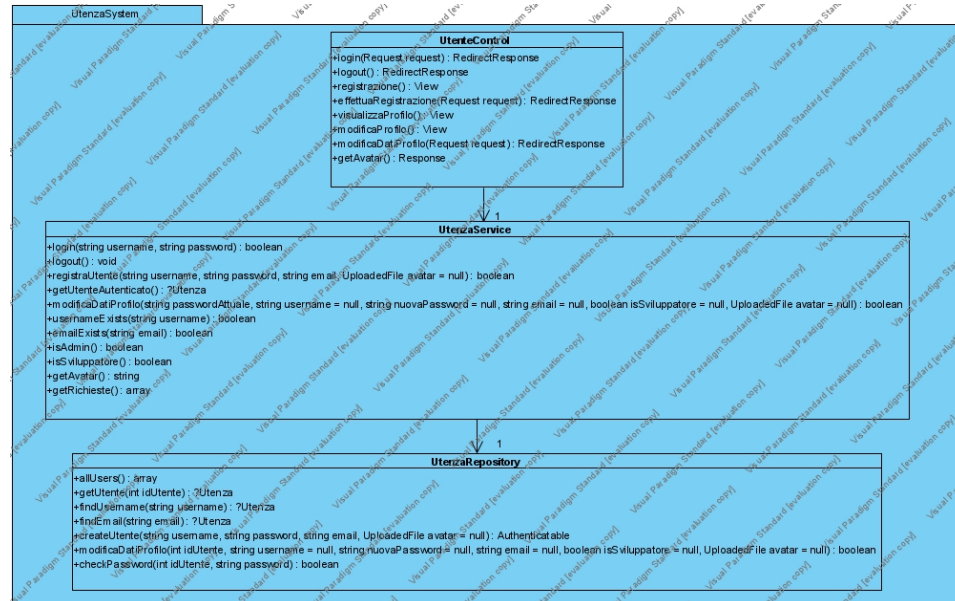
3.3.2 Videogioco Repository

Nome	VideogiocoRepository
Descrizione	Questa repository rappresenta il punto di accesso principale per la classe Videogioco e le classi strettamente correlate, ovvero Tags, Versioni, Sponsorizzazioni, Recensioni e ValutazioneRecensioni, assieme alle classi pivot di collegamento che collegano una di queste con altre classi, astraendo la comunicazione con i modelli di Eloquent per i dati relazionali e con il filesystem per i file.
Attributi	- N/A
Firme Metodi	<ul style="list-style-type: none"> + getVideogioco(int idVideogioco): ?Videogioco + applicaCriteri(string titolo = null, float prezzo = null, array tagsObbligatorie = null, array tagsOpzionali = null, int idUtente = null, boolean acquistati = false, string ordine = 'DESC'): array + creaVideogioco(Videogioco videogioco): void + aggiornaDatiVideogioco(Videogioco videogioco): void + getVersionVideogioco(int idVideogioco): array + getRichiesta(int idRichiesta): Richiesta
Pre-condizioni	<ul style="list-style-type: none"> • context VideogiocoRepository::getVersionVideogioco(idVideogioco) • pre: Videogiochi::find(idVideogioco) <> null
Post-condizioni	<ul style="list-style-type: none"> • N/A
Invarianti	<ul style="list-style-type: none"> • N/A

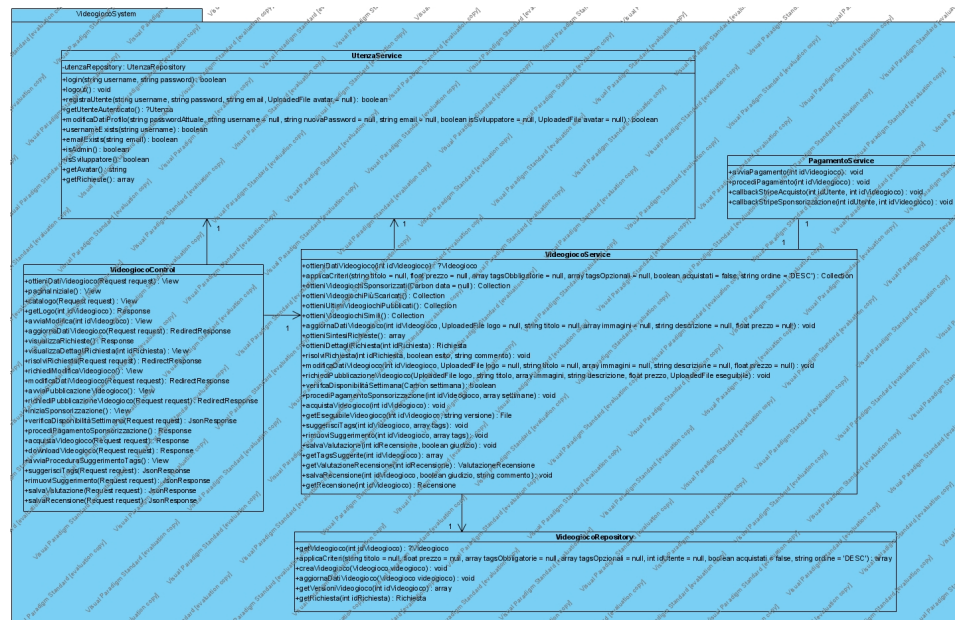
3.3.3 Forum Repository

Nome	ForumRepository
Descrizione	Questa repository rappresenta il punto di accesso principale per le classi Discussioni e Commenti.
Attributi	<ul style="list-style-type: none">- N/A
Firme Metodi	<ul style="list-style-type: none">+ getDiscussione(int idDiscussione): ?Discussione+ getCommenti(int idDiscussione): array+ creaCommento(int idDiscussione, string corpo): void+ creaDiscussione(string titolo, string corpo): void
Pre-condizioni	<ul style="list-style-type: none">• context ForumRepository::getCommenti(idDiscussione) pre: getDiscussione(idDiscussione) <> null• context ForumRepository::creaCommento(idDiscussione) pre: getDiscussione(idDiscussione) <> null
Post-condizioni	<ul style="list-style-type: none">• N/A
Invarianti	<ul style="list-style-type: none">• N/A

Sistema Utenza



Sistema Videogiocchi



Sistema Forum

