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	1.1	Conversione in formato TeX dell'intero documento e release iniziale

Indice

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 creata un package contenente i servizi del nostro sistema e le classi contenuti 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'istanziazione 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 interfacciarci 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 Utenza, 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 + tentaRecuperoPassword(Request request): Response + resetPassword(Request request): Response + visualizzaProfilo(): View + modificaProfilo(): View + modificaDatiProfilo(Request request): RedirectResponse + getAvatar(): Response
Pre-condizioni	• **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’, ’avatar’]) • **context** UtenteControl::tentaRecuperoPassword(request) **pre:** !utenzaService.isAuthenticated() and request.has(‘email’) • **context** UtenteControl::resetPassword(request) **pre:** !utenzaService.isAuthenticated() and request.hasValidSignature() and request.has([‘email’, ‘password’, ‘confermaPassword’]) • **context** UtenteControl::visualizzaProfilo() **pre:** utenzaService.isAuthenticated() • **context** UtenteControl::modificaProfilo() **pre:** utenzaService.isAuthenticated() • **context** UtenteControl::modificaDatiProfilo(request) **pre:** request.hasAny([‘username’, ‘email’, ‘isSviluppatore’, ‘avatar’, ‘nuovaPassword’]) and utenzaService.isAuthenticated() • **context** UtenteControl::getAvatar() **pre:** utenzaService.isAuthenticated()

UtenteControl	Post-condizioni	• **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::resetPassword(request) **post:** utenzaService.isAuthenticated() and utenzaService.getUtenteAutenticato().password == request.input('password') • **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	• **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	- videogiocoService: VideogiocoService - utenzaService: UtenzaService - pagamentoService: PagamentoService
Firme Metodi	+ ottieniDatiVideogioco(Request request): Response + paginaIniziale(): View + catalogo(): View + getLogo(int idVideogioco): Response + getListaVideogiochi(): Response + applicaCriteri(Request request): JsonResponse + videogiochiInEvidenza(): Response + avviaModifica(): Response + aggiornaDatiVideogioco(Request request): Response + visualizzaRichieste(): Response + visualizzaDettagliRichiesta(Request request): Response + risolviRichiesta(Request request): Response + richiediModificaVideogioco(): Response() + modificaDatiVideogioco(Request request): Response + richiediPubblicazioneVideogioco(Request request): Response + iniziaSponsorizzazione(): Response + verificaDisponibilitàSettimana(Request request): JsonResponse + procediPagamentoSponsorizzazione(): Response + acquistaVideogioco(Request request): Response + downloadVideogioco(Request request): Response + avviaProceduraSuggerimentoTags(): Response + suggerisciTags(Request request): JsonResponse + rimuoviSuggerimento(Request request): JsonResponse + salvaValutazione(Request request): JsonResponse + salvaRecensione(Request request): JsonResponse + iniziaReport(): Response + creaReport(Request request): Response + nascondi(Request request): Response

VideogiocoControl	Pre-condizioni	• **context** VideogiocoControl::ottieniDatiVideogioco(request) **pre:** request.has('idVideogioco') • **context** VideogiocoControl::applicaCriteri(request) **pre:** request.hasAny(['titolo', 'prezzo', 'tagsObbligatorie', 'tagsOpzionali', 'acquistati', 'criterioOrdine']) • **context** VideogiocoControl::aggiornaDatiVideogioco(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has('idVideogioco') and request.hasAny(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo']) • **context** VideogiocoControl::visualizzaRichieste(request) **pre:** Auth::check() and utenzaService.isAdmin() • **context** VideogiocoControl::visualizzaDettagliRichiesta(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has('idRichiesta') • **context** VideogiocoControl::risolviRichiesta(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has(['idRichiesta', 'esito', 'commento']) • **context** VideogiocoControl::modificaDatiVideogioco(request) **pre:** Auth::check() and utenzaService.isSviluppatore() and request.has('idVideogioco') and request.hasAny(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo']) • **context** VideogiocoControl:: richiediPubblicazioneVideogioco(request) **pre:** Auth::check() and utenzaService.isSviluppatore() and request.has(['logo', 'titolo', 'immagini', 'descrizione', 'prezzo', 'eseguibile']) • **context** VideogiocoControl::iniziaSponsorizzazione() **pre:** Auth::check() and utenzaService.isSviluppatore() • **context** VideogiocoControl:: verificaDisponibilitàSettimana(request) **pre:** Auth::check() and utenzaService.isSviluppatore() and request.has('settimana') • **context** VideogiocoControl:: procediPagamentoSponsorizzazione(request) **pre:** Auth::check() and utenzaService.isSviluppatore() and request.has(['idVideogioco', 'settimane']) and videogiocoService.settimaneDisponibili(request.input('settimane')) • **context** VideogiocoControl::acquistaVideogioco(request) **pre:** Auth::check() and request.has('idVideogioco')

VideogiocoControl	Pre-condizioni	• **context** VideogiocoControl::downloadVideogioco(request) **pre:** Auth::check() and request.hasAll([‘idVideogioco’, ‘versione’]) and videogiocoService.getVideogiochiAcquistati(auth().id)->includes(videogiocoService .getVideogioco(request.input(‘idVideogioco’))) • **context** VideogiocoControl::avviaProceduraSuggerimentoTags() **pre:** Auth::check() and request.has(‘idVideogioco’) and videogiocoService.getVideogiochiAcquistati(auth().id)->includes(videogiocoService .getVideogioco(request.input(‘idVideogioco’))) • **context** VideogiocoControl::suggerisciTags(request) **pre:** Auth::check() and request.hasAll([‘idVideogioco’, ‘tags’]) and videogiocoService.getVideogiochiAcquistati(auth()-¿id)->includes(videogiocoService .getVideogioco(request.input(‘idVideogioco’))) • **context** VideogiocoControl::rimuoviSuggerimento(request) **pre:** Auth::check() and request.hasAll([‘idVideogioco’, ‘tags’]) and videogiocoService.getTagsSuggerite(request.input(‘idVideogioco’), auth().id)->intersection(request.input(‘tags’))->size != 0 • **context** VideogiocoControl::salvaValutazione(request) **pre:** Auth::check() and request.has(‘idRecensione’) and videogiocoService.getValutazioneRecensione(request.input(‘idRecensione’), auth().id) == null • **context** VideogiocoControl::salvaRecensione(request) **pre:** Auth::check() and request.has(‘idVideogioco’) and videogiocoService.getRecensione(request.input(‘idVideogioco), auth().id) == null • **context** VideogiocoControl::iniziaReport() **pre:** Auth::check() • **context** VideogiocoControl::creaReport(request) **pre:** Auth::check() and request.hasAll([‘idCommento’, ‘motivo’]) • **context** VideogiocoControl::nascondi(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has(‘idVideogioco’) and videogiocoService.getVideogioco(request.input(‘idVideogioco’)).nascosto == false

VideogiocoControl	Post-condizioni	• **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') .getNumRichieste() = videogiocoService .getVideogioco(request.input('idVideogioco') @pre.getNumRichieste() + 1 • **context** VideogiocoControl ::richiediPubblicazioneVideogioco(request) **post:** utenzaService .getNumRichieste(utenzaService.ottieniProfilo()) = utenzaService @pre.getNumRichieste(utenzaService.ottieniProfilo()) + 1 • **context** VideogiocoControl ::procediPagamentoSponsorizzazione(request) **post:** videogiocoService .getNumSponsorizzazioni(request.input('idVideogioco')) = videogiocoService @pre.getNumSponsorizzazioni(request .input('idVideogioco')) + request.input('settimane').size • **context** VideogiocoControl::acquistaVideogioco(request) **post:** videogiocoService .getVideogiochiAcquistati(auth().id) = videogiocoService @pre.getVideogiochiAcquistati(auth().id) + 1 and videogiocoService. checkVideogiocoAcquistato(request.input('idVideogioco')) == true • **context** VideogiocoControl::suggerisciTags(request) **post:** videogiocoService .getTags(request.input('idVideogioco')) ->intersection(request.input('tags')) == request.input('tags') • **context** VideogiocoControl::rimuoviSuggerimento(request) **post:** videogiocoService.getTagsSuggerite(request.input('idVideogioco'), auth().id)->intersection(request.input('tags'))->size == 0

VideogiocoControl

Post-condizioni	• **context** VideogiocoControl::salvaRecensione(request) **post:** videogiocoService.getRecensione(request.input('idVideogioco'), auth().id) != null • **context** VideogiocoControl::nascondi(request) **post:** videogiocoService.getVideogioco(request.input('idVideogioco')).nascosto == true
Invarianti	• **context** VideogiocoControl **inv:** videogiocoService != null and utenzaService != null and pagamentoService != 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	- videogiocoService: VideogiocoService - utenzaService: UtenzaService - forumService: ForumService
Firme Metodi	+ chiudiDiscussione(Request request): JsonResponse + creaNuovaDiscussione(Request request): Response + creaDiscussione(Request request): Response + commenta(Request request): JsonResponse + iniziaReport(): Response + creaReport(Request request): JsonResponse + poniInRilievo(Request request): JsonResponse + nascondi(Request request): Response + visualizzaDettagliReport(Request request): Response + risolviReport(Request request): Response
Pre-condizioni	• **context** ForumControl::chiudiDiscussione(request) **pre:** Auth::check() and request.has('idDiscussione') and forumService .checkPermessiDiscussione(request.input('idDiscussione')) and forumService .getDiscussione(request.input('idDiscussione')).chiusa == false • **context** ForumControl::creaNuovaDiscussione(request) **pre:** Auth::check() and request.has('idVideogioco') and videogiocoService.checkVideogiocoAcquistato(request.input('idVideogioco')) == true • **context** ForumControl::creaDiscussione(request) **pre:** Auth::check() and request.hasAll(['idVideogioco', 'titolo', 'corpo']) and videogiocoService.checkVideogiocoAcquistato(request.input('idVideogioco')) == true

ForumControl	Pre-condizioni	• **context** ForumControl::commenta(request) **pre:** Auth::check() and request.hasAll([‘idDiscussione, ‘corpo’]) and videogiocoService.checkVideogiocoAcquistato(request.input(‘idVideogioco’)) == true • **context** ForumControl::iniziaReport() **pre:** Auth::check() • **context** ForumControl::creaReport(request) **pre:** Auth::check() and request.hasAll([‘idCommento’, ‘motivo’]) • **context** ForumControl::poniInRilievo(request) **pre:** Auth::check() and request.has(‘idDiscussione’) and forumService.checkPermessiDiscussione(request.input(‘idDiscussione’)) and forumService.getDiscussione(request.input(‘idDiscussione’)).in_rilievo == false • **context** ForumControl::nascondi(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has(‘idCommento’) and forumService.getCommento(request.input(‘idCommento’)).nascosto == false • **context** ForumControl::visualizzaDettagliReport(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has(‘idReport’) • **context** ForumControl::risolviReport(request) **pre:** Auth::check() and utenzaService.isAdmin() and request.has(‘idReport’) and forumService.getReport(request.input(‘idReport’)).esito == null
	Post-condizioni	• **context** ForumControl::chiudiDiscussione(request) **post:** forumService.getDiscussione(request.input(‘idDiscussione’) == true • **context** ForumControl::creaDiscussione(request) **post:** forumService.getNumDiscussioni(request.input(‘idVideogio == forumService@pre.getNumDiscussioni(request.input(‘idVideo + 1

ForumControl	Post-condizioni	• **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 • **context** ForumControl::nascondi(request) **post:** forumService .getCommento(request.input('idCommento')).nascosto == true • **context** ForumControl::risolviReport(request) **post:** forumService .getReport(request.input('idReport')).esito != null
	Invarianti	• **context** ForumControl **inv:** forumService != null and videogiocoService != null and utenzaService != null

3.2 Classi Service

3.2.1 Utenza 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	- utenzaRepository: UtenzaRepository
Firme Metodi	+ login(string username, string password): boolean + logout(): void + registraUtente(string username, string password, string email, File avatar): boolean + tentaRecuperoPassword(string email): boolean + resetPassword(string email, string nuovaPassword) + ottieniProfilo(): Utente + modificaDatiProfilo(string username, string email, File avatar, boolean isSviluppatore) + usernameExists(string username): boolean + emailExists(string email): boolean + isAdmin(): boolean + isSviluppatore(): boolean + getNumRichieste(Utente utente): integer
Pre-condizioni	• **context** UtenzaService::login(username, password) **pre:** utenzaRepository.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::tentaRecuperoPassword(email) **pre:** emailExists(email) • **context** UtenzaService::modificaDatiProfilo(username, email, avatar, isSviluppatore) **pre:** !emailExists(email) and !usernameExists(username) • **context** UtenzaService::getNumRichieste(utente) **pre:** utente.ruolo == 'Sviluppatore'

Utenza Service	Post-condizioni	• **context** UtenzaService::login(username, password) **post:** ottieniProfilo().username == username • **context** UtenzaService::logout() **post:** ottieniProfilo() == null • **context** UtenzaService::registraUtente(username, password, email, avatar) **post:** ottieniProfilo().username == username and ottieniProfilo().password == Hash::make(password) and ottieniProfilo().email == email and Hash::make(ottieniProfilo().avatar) == Hash::make(avatar) • **context** UtenzaService::resetPassword(email, password) **post:** ottieniProfilo().password == password • **context** UtenzaService::modificaDatiProfilo(username, email, avatar, isSviluppatore) **post:** (username == null or ottieniProfilo().username == username) and (email == null or ottieniProfilo().email == email) and (avatar == null or Hash::make(ottieniProfilo().avatar) == Hash::make(avatar)) and (isSviluppatore == null or ((ottieniProfilo().ruolo == 'Sviluppatore' and isSviluppatore) or (ottieniProfilo.ruolo == 'Cliente' and !isSviluppatore))
	Invarianti	• **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	- videogiocoRepository: VideogiocoRepository - utenzaRepository: UtenzaRepository - pagamentoService: PagamentoService
Firme Metodi	+ ottieniDatiVideogioco(int idVideogioco): Videogioco + getListaVideogiochi(): Videogioco[] + applicaCriteri(string titolo, float prezzo, string[] tagsObbligatorie, string[] tagsOpzionali, boolean acquistati == false, string ordine == 'DESC'): Videogioco[] + ottieniVideogiochiSponsorizzati(Carbon data == now()): Videogioco[] + ottieniVideogiochiPiùScaricati(): Videogioco[] + ottieniUltimiVideogiochiPubblicati(): Videogioco[] + ottieniVideogiochiSimili(int idUtente): Videogioco[] + aggiornaDatiVideogioco(int idVideogioco, File logo, string titolo, File[] immagini, string descrizione, float prezzo): void + ottieniSintesiRichieste(): Richiesta[] + ottieniDettagliRichiesta(int idRichiesta): Richiesta + risolviRichiesta(int idRichiesta, int idRisolutore, boolean esito, string commento): void + modificaDatiVideogioco(int idVideogioco, File logo, string titolo, File[] immagini, string descrizione, float prezzo): void + richiediPubblicazioneVideogioco(int idAutore, File logo, string titolo, File[] immagini, string descrizione, float prezzo, File eseguibile): void + verificaDisponibilitàSettimana(Carbon settimana): boolean + procediPagamentoSponsorizzazione(int idVideogioco, Carbon[] settimane): void + acquistaVideogioco(int idVideogioco) + getEseguibileVideogioco(int idVideogioco, string versione): File + suggerisciTags(int idUtente, int idVideogioco, string[] tags): void

Videogioco Service

Firme Metodi	+ rimuoviSuggerimento(int idUtente, int idVideogioco, string[] tags): void + salvaValutazione(int idUtente, int idRecensione, boolean giudizio) + getTagsSuggerite(int idVideogioco, int idUtente): Tags[] + getValutazioneRecensione(int idRecensione, int idUtente): ValutazioneRecensione + salvaRecensione(int idUtente, int idVideogioco, boolean giudizio, string commento): void + getRecensione(int idVideogioco, int idUtente): Recensione + creaReportVideogioco(int idVideogioco, int idUtente, string motivo): void + nascondiVideogioco(int idVideogioco): void + getVideogioco(int idVideogioco): Videogioco
Pre-condizioni	• **context** VideogiocoService::ottieniDatiVideogioco(idVideogioco) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null • **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(idUtente) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null • **context** VideogiocoService::aggiornaDatiVideogioco(idVideogioco logo, titolo, immagini, descrizione, prezzo) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != nulland (logo != null or titolo != null or immagini != null or immagini->size != 0 or descrizione != null or prezzo != null) • **context** VideogiocoService::ottieniDettagliRichiesta(idRichiesta) **pre:** idRichiesta != null

Videogioco Service	Pre-condizioni	• **context** VideogiocoService::risolviRichiesta(idRichiesta, idRisolutore, esito, commento) **pre:** idRichiesta != null and videogiocoRepository.getRichiesta(idRichiesta) != null and idRisolutore != null and utenzaRepository.getUtente(idRisolutore) != null and esito != null • **context** VideogiocoService::modificaDatiVideogioco(idVideogioco logo, titolo, immagini, descrizione, prezzo) **pre:** idVideogioco != null and 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(idAu logo, titolo, immagini, descrizione, prezzo, eseguibile) **pre:** idAutore != null and utenzaRepository.getUtente(idAutore) != null (logo != null or titolo != null or immagini != null or immagini->size != 0 or descrizione != null or prezzo != null) • **context** VideogiocoService::verificaDisponibilitàSettimana(settima **pre:** settimana != null • **context** VideogiocoService::procediPagamentoSponsorizzazione(id settimane) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and settimane != null and settimane->size != 0 • **context** VideogiocoService::acquistaVideogioco(idVideogioco) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null • **context** VideogiocoService::getEseguibileVideogioco(idVideogioco versione) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and videogiocoRepository.getVersioniVideogioco(idVideogioco)->s • **context** VideogiocoService::suggerisciTags(idUtente, idVideogioco, tags) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and tags != null and tags->size != 0

Videogioco Service	Pre-condizioni	• **context** VideogiocoService::rimuoviSuggerimenti(idUtente, idVideogioco, tags) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and tags != null and tags->size != 0 • **context** VideogiocoService::salvaValutazione(idUtente, idRecensione, giudizio) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and tags != null and tags->size != 0 and giudizio != null • **context** VideogiocoService::getTagsSuggerite(idVideogioco, idUtente) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null • **context** VideogiocoService::getValutazioneRecensione(idRecensio idUtente) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idRecensione != null and videogiocoRepository.getRecensione(idRecensione) != null • **context** VideogiocoService::salvaRecensione(idUtente, idVideogioco, giudizio, commento) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and giudizio != null and commento != null • **context** VideogiocoService::salvaRecensione(idVideogioco, idUtente) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null • **context** VideogiocoService::creaReportVideogioco(idVideogioco, idUtente, motivo) **pre:** idUtente != null and utenzaRepository.getUtente(idUtente) != null and idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and motivo != null

Videogioco Service	Pre-condizioni	• **context** VideogiocoService::nascondiVideogioco(idVideogioco) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null • **context** VideogiocoService::getVideogioco(idVideogioco) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null
	Post-condizioni	• **context** VideogiocoService::ottieniVideogiochiPiùScaricati() **post:** result->sortedBy(videogioco | -videogioco.downloads) • **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, idRisolutore, esito, commento) **post:** videogiocoRepository.getRichiesta(idRichiesta).esito == esito and videogiocoRepository.getRichiesta(idRichiesta).commento == commento • **context** VideogiocoService::verificaDisponibilitàSettimana(settim **post:** videogiocoRepository.getSponsorizzazioni->forAll(sp | sponsorizzazione.data_inizio.startOfWeek() <> settimana.startOfWeek() and sponsorizzazione.data_fine.endOfWeek() <> settimana.endOfWeek()) and result == true • **context** VideogiocoService::suggerisciTags(idUtente, idVideogioco, tags) **post:** videogiocoRepository.getTagsVideogioco(idVideogioc == tags • **context** VideogiocoService::rimuoviSuggerimenti(idUtente, idVideogioco, tags) **post:** videogiocoRepository.getTagsVideogioco(idVideogioc == 0 • **context** VideogiocoService::nascondiVideogioco(idVideogioco) **post:** videogiocoRepository.getVideogioco(idVideogioco).nascosto == true

Videogioco Service

Invarianti	• **context** VideogiocoService **inv:** videogiocoRepository != null and utenzaRepository != 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	- forumRepository: ForumRepository - videogiocoRepository: VideogiocoRepository - utenzaRepository: UtenzaRepository
Firme Metodi	+ chiudiDiscussione(int idDiscussione): void + creaDiscussione(int videogiocoId, string titolo, string corpo): Discussione + commenta(int idDiscussione, string corpo): Commento + creaReportCommento(int idCommento, string motivo): void + poniInRilievo(int idDiscussione): void + nascondiCommento(int idCommento): void + ottieniDettagliReport(int idReport): Report + risolviReport(int idReport, string giudizio): void
Pre-condizioni	• **context** ForumService::chiudiDiscussione(idDiscussione) **pre:** idDiscussione != null and forumRepository.getDiscussione(idDiscussione) != null • **context** ForumService::creaDiscussione(idVideogioco, titolo, corpo) **pre:** idVideogioco != null and videogiocoRepository.getVideogioco(idVideogioco) != null and titolo != null and corpo != null • **context** ForumService::commenta(idDiscussione, corpo) **pre:** idDiscussione != null and forumRepository.getDiscussione(idDiscussione) != null and corpo != null • **context** ForumService::creaReportCommento(idCommento, motivo) **pre:** idCommento != null and forumRepository.getCommento(idCommento) != null and motivo != null

Forum Service

Pre-condizioni	• **context** ForumService::poniInRilievo(idCommento, motivo) **pre:** idDiscussione != null and forumRepository.getDiscussione(idDiscussione) != null and forumRepository.getDiscussione(idDiscussione).in_rilievo == false • **context** ForumService::nascondiCommento(idCommento) **pre:** idCommento != null and forumRepository.getCommento(idCommento) != null and forumRepository.getCommento(idCommento).nascosto == false • **context** ForumService::ottieniDettagliReport(idReport) **pre:** idReport != null and forumRepository.getReport(idReport) != null • **context** ForumService::risolviReport(idReport, giudizio) **pre:** idReport != null and forumRepository.getReport(idReport) != null and giudizio != null
Post-condizioni	• **context** ForumService::chiudiDiscussione(idDiscussione) **post:** forumRepository.getDiscussione(idDiscussione).chiusa == true • **context** ForumService::commenta(idDiscussione, corpo) **post:** forumRepository.getNumCommenti(idDiscussione) == forumRepository@pre.getNumCommenti(idDiscussione) + 1 • **context** ForumService::commenta(idDiscussione, corpo) **post:** forumRepository.getNumCommenti(idDiscussione) == forumRepository@pre.getNumCommenti(idDiscussione) + 1 • **context** ForumService::poniInRilievo(idDiscussione) **post:** forumRepository.getDiscussione(idDiscussione).in_rilievo == true • **context** ForumService::nascondiCommento(idCommento) **post:** forumRepository.getCommento(idCommento).nascosto == true • **context** ForumService::risolviReport(idReport, giudizio) **post:** forumRepository.getReport(idReport).giudizio == giudizio

Forum Service

Invarianti	• **context** ForumService **inv:** forumRepository != null and videogiocoRepository != null and utenzaRepository != 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	• **context** PagamentoService::avviaPagamento(idVideogioco) **pre:** idVideogioco != null and videogiocoService.getVideogioco(idVideogioco) != null • **context** PagamentoService::procediPagamento(idVideogioco) **pre:** idVideogioco != null and videogiocoService.getVideogioco(idVideogioco) != null
Post-condizioni	• **context** PagamentoService::callbackStripeAcquisto() **post:** videogiocoService.getVideogiochiAcquistati(idUtente)->size == videogiocoService@pre.getVideogiochiAcquistati(idUtente)->size + 1 • **context** PagamentoService::callbackStripeAcquisto() **post:** videogiocoService.getNumSponsorizzazioni(idVideogioco)->size == videogiocoService@pre.getNumSponsorizzazioni(idVideogioco)->size + 1
Invarianti	• **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	- N/A
Firme Metodi	+ allUsers(): Utente[] + getUtente(int idUtente): ?Utente
Pre-condizioni	• N/A
Post-condizioni	• **context** UtenzaRepository::getUtente(idUtente) **post:** result.id == idUtente
Invarianti	• 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	+ getVideogioco(int idVideogioco): Videogioco[] + getVideogiochiPerTitoloSimile(string titolo): Videogioco[] + getVideogiochiMaxPrezzo(float prezzo): Videogioco[] + getVideogiocoTagsObbligatorie(string[] tags): Videogioco[] + getVideogiocoTagsOpzionali(string[] tags): Videogioco[] + getVideogiochiSponsorizzati(Carbon data): Videogioco[] + getVideogiochiSimili(int idUtente): Videogioco[] + aggiornaDatiVideogioco(Videogioco videogioco): void + getRichiesta(int idRichiesta): Richiesta
Pre-condizioni	• N/A
Post-condizioni	• **context** UtenzaRepository::getUtente(idUtente) **post:** result.id == idUtente
Invarianti	• N/A