

Ingegneria, Gestione ed Evoluzione del Software Change

Request RaafGaming

Antonio De Lucia, Francesco Peluso e Antonio Maddaloni



1 Change request

1.1 Migrazione delle tecnologie front-end e back-end con ristrutturazione architetture basata sui principi della programmazione orientata agli oggetti.;

Situazione attuale: L'applicazione RaafGaming, nella sua forma attuale, è costruita utilizzando Java con JSP e Servlet, che svolgono sia il ruolo di presentazione sia quello di controller. L'accesso ai dati è gestito tramite DAO/DTO. Pur essendo presenti DAO e DTO, questa suddivisione è più organizzativa che architettonica: l'applicazione adotta infatti un approccio fortemente data-driven, in cui ogni chiamata applicativa si traduce direttamente nell'esecuzione di una query, senza un livello dedicato di logica di business. Anche la gestione delle relazioni tra entità è implementata “a mano”, coordinando diversi DAO. A questo si aggiunge un ulteriore problema: le tecnologie adottate sono ormai obsolete, rendendo la manutenzione, l'estensione e la modifica del codice più complessa e costosa, soprattutto in assenza di una struttura architettonica moderna. Dal punto di vista architettonico, le Servlet finiscono per includere parte della logica applicativa, violando il pattern MVC. Il Model è costituito da DTO passivi, privi di comportamento proprio. I diagrammi di sequenza evidenziano chiaramente questa impostazione: il flusso dell'applicazione si riduce a una catena lineare JSP → Servlet → DAO → Database, senza un livello intermedio che incapsuli o organizzi la logica applicativa.

Modifica: La nuova architettura prevede una migrazione completa a Laravel, adottando il framework PHP per implementare un vero pattern MVC. La mappatura delle entità viene affidata a Eloquent ORM, mentre le interfacce di presentazione vengono realizzate utilizzando le Blade View, sostituendo così le JSP. Un elemento fondamentale della nuova architettura è l'introduzione di un Application Layer formale. In particolare, viene creato un Service Layer che incapsula la logica di business: questo non solo permette di rispettare pienamente il pattern MVC, separando responsabilità di controller e modello, ma consente anche di ottimizzare le operazioni sui dati, evitando l'esecuzione di query ad ogni singola chiamata e migliorando così le prestazioni complessive dell'applicazione. L'accesso ai dati viene ottimizzato grazie alle funzionalità native di Eloquent, come Eager Loading, Lazy Loading e

l'uso della cache tramite Cache::remember, che migliorano sensibilmente le prestazioni nella gestione delle relazioni e del caricamento delle entità. Ulteriori ottimizzazioni sono possibili tramite meccanismi come pagination e chunking dei risultati. Il processo di refactoring include inoltre una ristrutturazione completa dei flussi applicativi: vengono eliminati i vecchi DAO/DTO, sostituiti dai Model Eloquent e dal loro sistema di relazioni (ad esempiohasMany, belongsTo), rendendo l'architettura più coerente, manutenibile e allineata alle moderne best practice.

Impatto: La migrazione dell'applicazione RaafGaming a Laravel comporta un impatto significativo, che va ben oltre la semplice riscrittura del codice. Si tratta di una ri-progettazione completa dell'architettura, della documentazione e dei test, con l'obiettivo di modernizzare l'applicazione e migliorarne manutenibilità e prestazioni. Dal punto di vista architettonico, cambiano i layer dell'applicazione: le JSP e le Servlet vengono sostituite rispettivamente da Blade View e Controller Laravel, viene aggiunto un Service Layer per la logica applicativa, i DAO/DTO vengono rimossi e i model diventano attivi grazie a Eloquent ORM con le sue relazioni native (hasMany, belongsTo, ecc.). Questo permette di rispettare pienamente il pattern MVC e di ottimizzare le operazioni sui dati, evitando query ad ogni chiamata. La documentazione va completamente aggiornata: requisiti non funzionali, class diagram, sequence diagram devono riflettere la nuova architettura, le nuove relazioni tra componenti e il flusso delle operazioni. Anche i flussi di navigazione vanno rivisti. Il testing richiede attenzione: bisogna rivedere i test funzionali e i test di integrazione. Il testing richiede particolare attenzione: è necessario rivedere sia i test funzionali sia i test di sistema. I test devono essere riscritti a livello di codice per adattarsi alle nuove tecnologie e agli oggetti che sono cambiati. La documentazione dei test deve essere aggiornata con attenzione, riflettendo le modifiche apportate a tecnologie e oggetti.