Cenni di ODD
Object Design Document

SoccerHub

Riferimento	NC28_SoccerHub_ODD.pdf
Versione	1.1
Data	10/12/2024
Destinatario	Prof. Carmine Gravino Prof. Gianmaria Giordano
Presentato da	Domenico Rago Gaetano Pascarella Fabian Andres Scalera Gabriele Armando Scialla
Approvato da	Prof. Carmine Gravino Prof. Gianmaria Giordano

Team members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Rago Domenico	Team Member	RD	d.rago6@srtudenti.unisa.it
Scialla Gabriele Armando	Team Member	SG	g.scialla2@studenti.unisa.it
Pascarella Gaetano	Team Member	PG	g.pascarella6@studenti.unisa.it
Scalera Fabian Andres	Team Member	SF	f.scalera1@studenti.unisa.it

Revison History

Data	Versione	Descrizione	Autori
10/12/2024	0.1	Prima stesura	Tutto il team
11/12/2024	0.2	Rifinitura del documento e stesura del Glossario	g.scialla2@studenti.unisa.it f.scalera1@studenti.unisa.it
12/12/2024	1.0	Revisione e primo rilascio	d.rago6@studenti.unisa.it g.pascarella6@studenti.unisa.it
13/12/2024	1.1	Ultima revisione e accorgimenti	Tutto il team

Sommario

1 Introduzione

SoccerHub si propone di semplificare le interazioni tra Club e membri del club intesi non solo come giocatori, ma anche allenatori e manager, al fine di rinvigorire il settore bibliotecario italiano creando uno strumento di gestione, monitoraggio e comunicazione delle squadre create.

In questa prima sezione del documento, verranno descritti i trade-offs e le linee guida per la fase di implementazione, riguardanti la nomenclatura, la documentazione e le convenzioni sui formati.

1.1 Object design goals

Riusabilità:

Il sistema SoccerHub deve basarsi sulla riusabilità, attraverso l'utilizzo di ereditarietà e design patterns.

Robustezza:

Il sistema deve risultare robusto, reagendo correttamente a situazioni impreviste attraverso il controllo degli errori e la gestione delle eccezioni.

Sicurezza:

Il sistema garantisce la segretezza sui dettagli implementativi delle classi grazie all'utilizzo delle interfacce, rendendo possibile l'utilizzo di funzionalità offerte da diversi componenti o layer sottoforma di black- box.

1.2 Linee guida per la documentazione dell'interfaccia

Le linee guida includono una lista di regole che gli sviluppatori dovrebbero rispettare durante la progettazione delle interfacce. Per la loro costruzione si è fatto riferimento alla convenzione java nota come **Sun Java Coding Conventions** [Sun, 2009].

Link a documentazione ufficiale sulle convenzioni

Di seguito una lista di link alle convenzioni usate per definire le linee guida:

- **Java Sun**: https://checkstyle.sourceforge.io/sun_style.html
- **HTML**: https://www.w3schools.com/html/html5_syntax.asp

1.3 Definizioni, acronimi, e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento:

- **Design pattern**: template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità;
- **View**: nel pattern MVC rappresenta ciò che viene visualizzato a schermo da un utente e che gli permette di interagire con le funzionalità offerte dalla piattaforma;

- **lowerCamelCase**: è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione nel mezzo della frase inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi;
- **UpperCamelCase**: è la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi;

1.4 Riferimenti

Di seguito una lista di riferimenti ad altri documenti utili durante la lettura:

- Statement of work;
- Requirements Analysis Document;
- System Design Document;
- Test Plan;
- Test Incident Report;
- Test Case Scenario;

2 Design Patterns

Nella presente sezione si andranno a descrivere e dettagliare i design patterns utilizzati nello sviluppo dell'applicativo SoccerHub. Per ogni pattern si darà:

- Una brevissima introduzione teorica.
- Il problema che doveva risolvere all'interno di SoccerHub.
- Una brevissima spiegazione di come si è risolto il problema in SoccerHub.

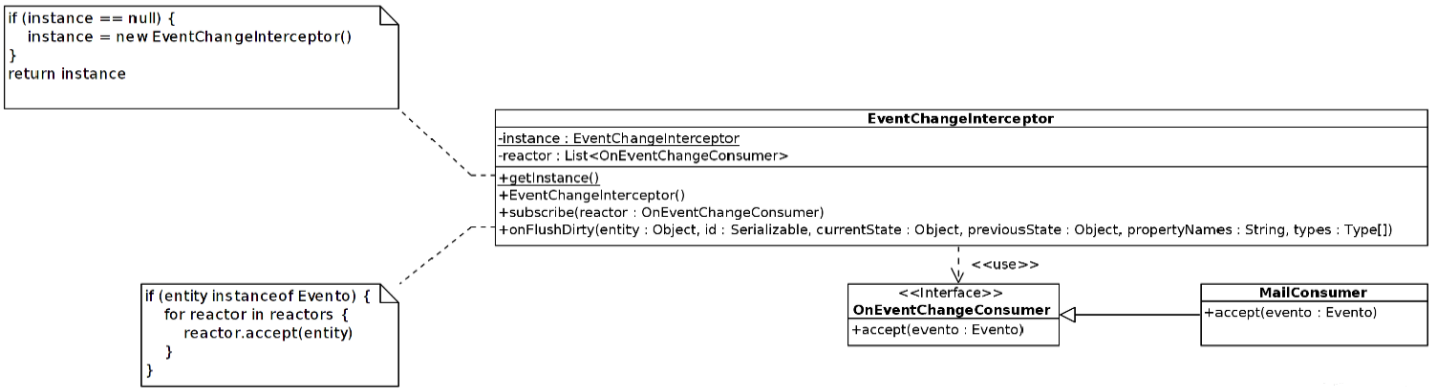
Singleton

Singleton è un design pattern creazionale, ossia un design pattern che si occupa dell'istanziazione degli oggetti, che ha lo scopo di garantire che di una determinata classe venga strutturata una sola istanza e di fornire un punto di accesso globale a tale istanza. Lo scopo principale di questo pattern è evitare la creazione di più istanze della stessa classe nel ciclo di vita dell'applicazione, garantendo così coerenza e riducendo il consumo di risorse. Inoltre, il pattern Singleton offre il vantaggio di centralizzare l'accesso all'istanza, rendendo la gestione più semplice e prevedibile.

SoccerHub prevede un sistema di notifica via email per la modifica degli allenamenti.

Per gestire ciò a livello implementativo viene creato un oggetto Singleton che conserva una propria lista di osservatori dotati di un metodo "accept" che accetti un singolo Allenamento come argomento. Ogni volta che viene modificato un allenamento sul database, l'oggetto "notifica" così la lista degli osservatori tramite il metodo onFlushDirty.

Ciò rende possibile la creazione in un momento successivo di multipli osservatori legati agli allenamenti(ad esempio per le notifiche push per un'eventuale applicazione mobile).



DAO

Un DAO (Data Access Object) è un pattern che offre un'interfaccia astratta per alcuni tipi di database. Mappando le chiamate dell'applicazione allo stato persistente, il DAO fornisce alcune operazioni specifiche sui dati senza esporre i dettagli del database. I DAO sono utilizzabili nella maggior parte dei linguaggi e la maggior parte dei software con bisogni di persistenza, principalmente viene associato con applicazioni JavaEE che utilizzano database relazionali.

Essendo SoccerHub una web application che punta di gestire sia il mercato acquisti dei giocatori che numerose squadre presenta un database molto vasto, quindi ha bisogno di poter interagire con database in modo rapido e sicuro con numerosi query per quella che è la moltitudine di dati da gestire. Per questo motivo abbiamo usato varie interfacce DAO all'interno del nostro sistema le quali vengono implementate in maniera del tutto automatica e trasparente per il programmatore.

3 Glossario

Sigla/Termine	Definizione
Package	Raggruppamento di classi ed interfacce.
DAO	Data Access Object, implementazione dell'omonimo pattern architetturale che si occupa di fornire un accesso in modo astratto ai dati persistenti.
Controller	Classe che si occupa di gestire le richieste effettuate dal client.
Service	Classe che implementa la logica di business, viene utilizzata dal controller o da un altro sottosistema.

Model	Parte del design architetturale MVC che fornisce al sistema i metodi per accedere ai dati utili al sistema.
MVC	Model-View-Controller: design architetturale che permette di separare la logica di presentazione dalla logica di business alla base del sistema.

Singleton	È un design pattern creazionale che ha lo scopo di garantire che di una determinata classe venga struttura una solo istanza e di fornire un punto di accesso globale a tale istanza.