Laurea Triennale in informatica-Università di Salerno
Corso di *Ingegneria del Software* - Prof. C. Gravino

Cenni su ODD

Object Design Document

AstroVerse

Riferimento	NC31 ODD
Versione	2.0
Data	25/11/2024
Destinatario	Professore C. Gravino
Presentato da	Gruppo NC31

Revision History

Data	Versione	Descrizione	Autori
25/11/2024	0.1	Prima stesura	Christian Fontana, Pellegrino Piccolo
09/12/2024	0.2	Definizione e aggiunta del primo DP e del Glossario	Adriano De Vita, Christian Bianco
10/12/2024	1.0	Revisione generale e prima release	Tutto il team
13/12/2024	1.1	Modifiche all'1.3	Adriano De Vita
20/12/2024	2.0	Aggunta ultimo DP e Review Pre-release	Tutto il team

Team members

Nome	Ruolo progettuale	Contatto
Pellegrino Piccolo	Membro del team	p.piccolo4@studenti.unisa.it
Adriano De Vita	Membro del team	a.devita40@studenti.unisa.it
Christian Fontana	Membro del team	c.fontana7@studenti.unisa.it
Christian Bianco	Membro del team	c.bianco9@studenti.unisa.it

Sommario

1. Introduzione

In questa prima sezione del documento, verranno introdotti gli obiettivi e le linee guida per la fase di implementazione.

1.1 Obiettivi di object design

Riusabilità:
Il sistema AstroVerse si basa sulla riusabilità, attraverso l'utilizzo di ereditarietà e design patterns

Robustezza:
Il sistema AstroVerse deve essere progettato per garantire robustezza, rispondendo in modo adeguato a situazioni impreviste grazie al controllo degli errori.

Manutenibilità:
Il sistema AstroVerse deve essere sviluppato in modo da assicurare la manutenibilità del codice nel tempo, semplificando le attività di manutenzione dopo il rilascio in produzione.

Sicurezza:
Il sistema AstroVerse deve assicurare la riservatezza e l'integrità dei dati attraverso l'implementazione di sistemi di autenticazione robusti, progettati per prevenire eventuali vulnerabilità.

1.2 Linee guida per la documentazione dell'interfaccia

L'interfaccia del sistema AstroVerse è sviluppata utilizzando il framework Vue.js per garantire modularità, reattività e scalabilità. Queste linee guida mirano a standardizzare lo sviluppo e facilitare la collaborazione.
Link alla documentazione ufficiale Vue: https://vuejs.org/guide/introduction.html

1.3 Definizioni, acronimi e abbreviazioni

Vengono riportate di seguito alcune definizioni presenti nel documento:

- **ODD:** Object Design Document
- **DP:** Design Pattern

1.4 Riferimenti

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

SOW AstroVerse.docx
RAD AstroVerse.docx
SDD AstroVerse.docx

1.5 Panoramica del documento

Questa sezione descrive come viene suddiviso il documento ODD:

Capitolo 1. **Introduzione:** descrive in generale lo scopo del sistema, gli obiettivi di design individuati e i trade-off da rispettare.

Capitolo 2. **Cenni su scelte di Object Design:** vengono descritte le scelte dei design pattern individuati e utilizzati per lo sviluppo del sistema.

Capitolo 3. **Glossario:** descrive i termini tecnici utilizzati nell'ODD che non sono stati spiegati nei documenti di precedente fattura.

2. Cenni su scelte di Object Design

Di seguito verranno elencati 2 dei design pattern individuati di maggiore importanza all'interno del progetto AstroVerse

2.1 Design patterns

Un design pattern è un modello di soluzioni a problemi ricorrenti nell'ambito dell'implementazione di un sistema, di seguito presentiamo 2 design pattern individuati ed implementati nel codice

DP1: Singleton

Il Singleton (conosciuto anche come Single Point of Instance o Monostate) è un design pattern creazionale che garantisce che una classe abbia una singola istanza che fornisca un punto di accesso globale. Lo scopo del DP è assicurare che una sola istanza della classe sia creata in tutto il ciclo di vita dell'applicazione e fornire un accesso centralizzato all'istanza unica, prevenendo così il rischio di inconsistenza e il consumo innecessario di risorse. Abbiamo inoltre la necessità di garantire la thread-safety e la flessibilità per estensioni future dell'applicazione. Questo DP è applicabile in tutte le classi della nostra applicazione, esso viene utilizzato all'interno dei costruttori della nostra applicazione Java, qui di seguito un esempio:

```
private final PostRepository postRepository;

public PostService(PostRepository postRepository) {
    this.postRepository = postRepository;
}
```

La cache delle classi singleton è progettata per essere thread-safe, assicurando che due thread non creino istanze separate dal bean nello stesso momento. Lo svantaggio che deriva dall'utilizzo di questo DP è il testing unitario che diventa più difficile da realizzare attraverso i mock.

DP2: Facade

Il Facade Pattern (conosciuto anche come Aggregator o Service Facade) è un design pattern strutturale che fornisce un'interfaccia semplificata per l'interazione con un sistema complesso, nascondendo i dettagli delle sue implementazioni. Esso definisce un'interfaccia di livello superiore che rende il sottosistema più facile da utilizzare, proteggendo i client dall'esposizione diretta ai componenti del sottosistema e promuovendo un accoppiamento debole tra i client e i componenti del sottosistema. All'interno di un sottosistema, il Facade Pattern può essere particolarmente utile per centralizzare la logica di business che coinvolge più servizi o componenti, migliorando la manutenibilità e la chiarezza del codice. Di seguito un esempio:

```
@Service
public class OrderFacade {
    private final ProductService productService;
    private final PaymentService paymentService;
    private final NotificationService notificationService;

    public OrderFacade(ProductService productService,
PaymentService paymentService, NotificationService
notificationService) {
        this.productService = productService;
        this.paymentService = paymentService;
        this.notificationService = notificationService;
    }

    public String processOrder(String productId, String orderId,
double amount) {
        // Coordinamento dei servizi
        String productStatus =
productService.checkProductAvailability(productId);
        String paymentStatus =
paymentService.processPayment(orderId, amount);
        String notificationStatus =
notificationService.sendNotification("Order " + orderId + "
completed.");

        return productStatus + "\n" + paymentStatus + "\n" +
notificationStatus;
    }
}
```

3. Glossario

Nella presente sezione sono raccolte le sigle o i termini del documento ODD che necessitano di una definizione più appropriata e chiara per l'utente finale e che non sono state definite nei documenti precedenti.

Sigla/Termine	Definizione
Thread-safety	Capacità di in programma di funzionare correttamente se eseguito su più thread in maniera parallela
Thread	La più piccola unità di esecuzione di un processo che può essere eseguito parallelamente alle altre
Mock (Testing)	Oggetto o funzione creata appositamente per simulare il comportamento di un componente reale durante il testing
Scope	Indice di visibilità e di durata delle variabili del sistema
Bean	In Java, si riferisce ad una classe che esegue specifiche operazioni su oggetti attraverso specifici metodi d'accesso
Cache	Memoria volatile caratterizzata da un'ottima velocità nel memorizzare e operare su piccoli blocchi di dati
Testing unitario (di unità)	Tecnica di test software in cui le unità individuali del codice vengono testate atomicamente per verificare che funzionino come previsto