*Corso di Laurea in Informatica, prof. A. De Lucia,*

*a.a 2021/2022*

*Progetto di Ingegneria del Software*



*Object Design Document*

|  |  |
| --- | --- |
| Partecipanti | Matricola |
| Marta Coiro | 0512108154 |
| Katia Buonocore | 0512106528 |
| Rita Cuccaro | 0512109495 |
|  |  |
|  |  |
|  |  |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| Data | Versione | Descrizione | Autore |
| 16/11/2021 | 1.0 | Prima stesura del documento(Problem [Statement](#_top)) | Membri del team |
| 30/11/2021 | 1.0 | Requirement Analysis Document | Membri del team |
| 06/12/2021 | 1.0 | System Design Document | Membri del team |
| 20/12/2021 | 1.0 | Gestione Dati Persistenti\_MusicConsole | Membri del team |
| 27/12/2021 | 1.0 | Object Design Document | Membri del team |
|  |  | Test Plan | Membri del team |
|  |  |  |  |

Indice

1. INTRODUZIONE

1.1 Object Design Trade-Offs

1.2 Linee Guida per la Documentazione delle Interfacce

1.3 Definizioni, acronimi e abbreviazioni.

1.4 Riferimenti.

**1. INTRODUZIONE**

Dopo la realizzazione dei documenti RAD e SDD abbiamo descritto, in linea di massima, quello che sarà il nostro sistema e quindi i nostri obiettivi, tralasciando gli aspetti dell’implementazione. Il seguente documento ha lo scopo di produrre un modello capace di integrare in modo coerente e preciso tutte le diverse funzionalità individuate nelle fasi precedenti. In particolare, questo documento si vanno a descrivere i trade-offs generali realizzati dagli sviluppatori, le linee guida sulla documentazione delle interfacce e le convenzioni di codifica, le Interfacce delle classi, le operazioni, i tipi, gli argomenti e la signature dei sottosistemi definiti nel System Design.

**1.1 Object Design Trade-offs**

* **Comprensibilità vs Tempo**: Il codice deve essere al quanto più comprensibile per poter facilitare la fase di testing ed eventuali future modifiche del codice. A tale scopo, il codice sarà quindi accompagnato da commenti che ne semplifichino la comprensione. Questa caratteristica incrementerà il tempo di sviluppo, ma allo stesso tempo lo renderà più comprensibile.

* **Interfaccia vs Usabilità**: Il sistema verrà sviluppato con un’interfaccia grafica realizzata in modo da poter essere molto semplice, chiara ed intuitiva. Nell’interfaccia saranno presenti form, menù e pulsanti, disposti in maniera da rendere semplice l’utilizzo del sistema da parte dell’utente finale.

● **Sicurezza vs Efficienza**: La sicurezza, come descritto nei requisiti non funzionali, rappresenta uno degli aspetti importanti del sistema. A causa dei tempi di sviluppo molto limitati, ci limiteremo ad implementare un sistema di sicurezza basato sull’utilizzo di username e password degli utenti.

**1.2 Linee Guida per la Documentazione delle Interfacce**

Gli sviluppatori seguiranno alcune linee guida per la scrittura del codice:

**Naming convention**

● E’ buona norma utilizzare nomi:

1. Descrittivi

2. Pronunciabili

3. Di uso comune

4. Di lunghezza medio-corta

5. Non abbreviati

6. Evitando la notazione ungherese

7. Utilizzando solo caratteri consentiti (a-z, A-Z, 0-9)

**Variabili:**

● I nomi delle variabili devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Quest’ultime devono essere dichiarate ad inizio blocco, solamente una per riga e devono essere tutte allineate e facilitarne la leggibilità. Esse possono essere annotate con dei commenti. Esempio: titoloBrano

● E’ inoltre possibile, in alcuni casi, utilizzare il carattere underscore (“\_”) per la definizione del nome.

**Metodi:**

● I nomi dei metodi devono cominciare con una lettera minuscola, e le parole seguenti con la lettera maiuscola. Il nome del metodo tipicamente consiste in un verbo che identifica una azione, seguito dal nome di un oggetto. I nomi dei metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile() e SetNomeVariabile().

● I commenti dei metodi devono essere raggruppati in base alla loro funzionalità, la descrizione dei metodi deve apparire prima di ogni dichiarazione di metodo, e deve descriverne lo scopo. Deve includere anche informazioni sugli argomenti, sul valore di ritorno, e se applicabile, sulle eccezioni.

**Classi e pagine:**

● I nomi delle classi e delle pagine devono cominciare con una lettera maiuscola, e anche le parole seguenti all’interno del nome devono cominciare con una lettera maiuscola. I nomi di quest’ultime devono fornire informazioni sul loro scopo.

● La dichiarazione di classe deve essere caratterizzata da:

1. Dichiarazione della classe pubblica

2. Dichiarazioni di costanti

3. Dichiarazioni di variabili di classe

4. Dichiarazione di variabili d’istanza

5. Costruttore

6. Commento e dichiarazione dei metodi

**1.3 Definizioni, acronimi e abbreviazioni**

● RAD : Requirements Analysis Document

● SDD : System Design Document

● ODD : Object Design Document

**1.4 Riferimenti**

● Documento RAD del progetto MusicConsole.

● Docuemento Dati Persistenti del progetto MusicConsole.