 Università degli studi di Salerno  
Corso di Ingegneria del Software

**RooManageR**

System Design Document

Versione 1.1



15/12/2015

Coordinatore del progetto:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Andrea De Lucia |  |

Partecipanti:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Gargiulo Emanuele | 0512102244 |
| Izzo Giandomenico | 0512102292 |
| Malinconico Francesco | 0512103309 |

|  |
| --- |
| Scritto da: Gargiulo Emanuele, Izzo Giandomenico e Malinconico Francesco |

Sommario

[Introduzione 4](#_Toc437939978)

[Compromessi di Design 4](#_Toc437939979)

[Sicurezza ed efficienza 4](#_Toc437939980)

[Documentazione delle guideline dell’interfaccia 4](#_Toc437939981)

[Definizioni e Acronimi 4](#_Toc437939982)

[Riferimento 5](#_Toc437939983)

[Pacchetti 5](#_Toc437939984)

# Introduzione

## Compromessi di Design

La prima sezione del ODD è una introduzione al documento. Descrive il generale trade-offs da sviluppare, orientamenti e convenzioni, e una veduta d'insieme del documento. Le soluzioni progettuali individuate nel corso della fase di System Design impongono che la gestione dei dati persistenti avvenga in un database online, e che in locale ci sia memorizzata solo una. Per quel che riguarda invece le altre funzionalità dell'applicazione, si può affermare che verranno realizzate senza il riutilizzo di alcuna utility software esistente.

## Sicurezza ed efficienza

Saranno controllati tutti quei dati che possono generare conflitti con il database. Tutti i dati che hanno formato INT, DOUBLE, LONG saranno controllati per evitare di creare errori durante l'inserimento di una tupla nel database.

I dati che avranno formato String, per non aumentare il tempo di risposta dell'applicazione, non saranno controllati, bensì saranno inseriti automaticamente nel database.

Poiché l'applicazione sarà usabile con tutti gli OS, i controlli sui dati verranno effettuati in parte lato server e in parte lato client.

## Documentazione delle guideline dell’interfaccia

Gli sviluppatori del sistema dovranno seguire alcune linee guida durante la scrittura del codice:

* I nomi delle classi sono nomi singolari scritti in camel case come da convenzione Java. Nel caso si tratti di classi dedicate alla grafica, esse avranno il prefisso “UI”. Esempio: UIHomeProprietario
* I nomi dei metodi saranno principalmente composti dal verbo che indica l’azione e dal sostantivo dell’entità coinvolta nell’azione. Esempio. AddAnagrafica
* I nomi dei pacchetti dovranno essere costituiti essenzialmente da caratteri minuscoli
* I nomi delle classi controller saranno costituiti dal nome dell’entità più “manager”
* I metodi per l’accesso e la modifica delle variabili dovranno essere del tipo getNomeVariabile(), setNomeVariabile()
* Gli eventuali errori saranno ritornati tramite eccezioni
* I commenti alle classi, ai metodi e alle variabili di istanza dovranno seguire standard Javadoc, quindi iniziare con /\*\* e terminare con \*/.

## Definizioni e Acronimi

ODD: Object Design Document  
RAD: Requirement Analysis Document  
SDD: System Design Document   
SI: Sistema informativo  
OS: Operative System

## Riferimento

**B. Bruegge & A. Dutoit:** Object-Oriented Software Engineering: Using UML, Patterns and Java

**Sommerville:** Software Engineering

# Pacchetti

L’individuazione dei package è un’operazione che è già stata fatta in parte durante la fase di System Design. Sono state raggruppate in pacchetti tutte quelle classi che vengono utilizzate, in modo diverso, da più sottosistemi: dunque tale modifica è stata effettuata allo scopo di evitare ridondanze all’interno del codice, in modo che una stessa classe non debba essere definita più volte in maniera identica, all’interno di due o più package diversi tra loro. I package individuati in questa fase risultano quindi essere quelli citati di seguito:

* Cache  
  Contiene la classe cacheManager implementata attraverso il pattern Singleton. Questa Classe si occupa di tenere in memoria e di dare un accesso veloce alle strutture dati sulle quali lavora l’applicazione.
* Constants  
  Contiene le classi che definiscono le costanti per i tipi delle varie entità sulle quali lavora l’applicazione
* Entities  
  Contiene le classi che identificano le entità su cui si basa il sistema. Le entità sono entità POJO
* Exceptions  
  Contiene le eccezioni pensate per gestire casi particolari all’interno dell’applicazione
* Images

Contiene le immagini necessarie all’applicazione

* Interfaces  
  Contiene l’interfaccia di Callback dei vari web services
* Main  
  Contiene la classe di avvio del programma
* Ui  
  Contiene tutti i file relativi all’interfaccia grafica dell’applicazione
* Web\_services  
  Contiene le classi di controllo che si occupano della comunicazione con il web server.