

|  |
| --- |
| Object Design Document  Versione 1.0  **ANNO ACCADEMICO 2020/2021** |
|  |
| RISTOMANAGER  Ambrosio Salvatore  Costante Marco  Benitozzi Simone  Nappo Carla Alessia |



PARTECIPANTI

|  |  |
| --- | --- |
| **NOME** | **MATRICOLA** |
| Salvatore Ambrosio | 0512106166 |
| Costante Marco | 0512105772 |
| Benitozzi Simone | 0512105742 |
| Nappo Carla Alessia | 0512105956 |

REVISION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **DATA** | **VERSIONE** | **DESCRIZIONE** | **AUTORE** |
| 19/01/2021 | 1.0 | Prima stesura del template | Benitozzi Simone |
| 19/01/2021 | 1.1 | Definizione di Introduzione e Design Pattern | Benitozzi Simone |
| 25/01/2021 | 1.2 | Correzioni e aggiunta dettagli | Costante Marco |
| 29/01/2021 | 1.3 | Definizione dei packages | Benitozzi Simone |

# INDICE

[INDICE 4](#_Toc62807089)

[1.Introduzione 5](#_Toc62807090)

[1.1 Object Design Trade-offs 5](#_Toc62807091)

[1.2 Componenti off-the-shelf 6](#_Toc62807092)

[1.3 Linee Guida 7](#_Toc62807093)

[1.3.1 Package 7](#_Toc62807094)

[1.3.2 Naming Convention 7](#_Toc62807095)

[1.3.3 Variabili 8](#_Toc62807096)

[1.3.4 Costanti 8](#_Toc62807097)

[1.3.5 Metodi 8](#_Toc62807098)

[1.3.6 Classi e Pagine 8](#_Toc62807099)

[1.4 Acronimi 9](#_Toc62807100)

[1.5 Riferimenti 10](#_Toc62807107)

[3. Packages 12](#_Toc62807108)

[4. Class Interfaces 13](#_Toc62807109)

[5. Class Diagram Completo 14](#_Toc62807110)

# 1.Introduzione

## **1.1 Object Design Trade-offs**

* Usabilità vs Funzionalità: Il sistema dovrà essere prima di tutto intuitivo e di facile comprensione da parte dell’utente, questo anche al costo di rinunciare alla implementazione di alcune funzionalità che potrebbero complicarne l’utilizzo.
* Tempi di Sviluppo vs Sicurezza: Il meccanismo di sicurezza implementato, a causa dei tempi e le risorse limitate, prevederà un semplice sistema di login e differenziazione degli utenti in ruoli, con filtri per il controllo degli accessi ad operazioni dedicate.
* Comprensibilità vs Costi: Il codice sviluppato dovrà risultare quanto più possibile comprensibile e chiaro, questo a costo di un maggiore tempo dedicato alla scrittura, allo scopo di avere come risultato un codice riutilizzabile e integrabile in futuro con modifiche e aggiornamenti.

## **1.2 Componenti off-the-shelf**

Per l’implementazione del sistema verrà fatto uso di componenti software ausiliari che ne facilitino lo sviluppo

* JUnit: framework utilizzato per il test di unità delle componenti implementate.
* Mockito: framework di test open source per Java per il test di componenti lato server.
* Selenium: suite di tool utilizzati per automatizzare i test di sistema eseguendoli sul web browser.

## **1.3 Linee Guida**

Per la produzione di un codice comprensibile ed uniforme, che favorisca il riutilizzo e futuri aggiornamenti, andranno seguite nella sua stesura le linee guida di seguito elencate:

## **1.3.1 Package**

Il codice sorgente deve essere opportunamente suddiviso in package, per modellare i sottosistemi che compongono il sistema, che minimizzino l’accoppiamento tra le classi ed evidenzino la coesione tra classi con comportamenti simili.

A tale proposito, i package principali nei quali le classi verranno suddivise, comprensivi di eventuali sub-package per un maggiore partizionamento, sono i seguenti:

* Control: contenente le classi Servlet, quindi i comportamenti lato server del sistema.
* Model: contenente le classi che definiscono i Bean del sistema, i quali possono rappresentare sia entità persistenti nel DB, che oggetti temporanei utili per l’esecuzione delle funzionalità del programma a runtime.
* DAO: contenente le classi che implementano il pattern architetturale DAO, per la gestione della persistenza, le quali presentano dalle più semplici operazioni CRUD, a più complesse query di interrogazione al DB.
* Test: contenente le classi utilizzate per il testing del sistema.

## **1.3.2 Naming Convention**

Per la documentazione delle interfacce sarà necessario seguire una naming convention precisa, nel rispetto delle classiche notazioni dell’Object Oriented, che porti ad adottare nomi:

* Descrittivi
* Intuitivi e comprensibili
* Di lunghezza medio-corta
* Comprensivi possibilmente di carattere alfabetici, evitando quelli numeri ove possibile

## **1.3.3 Variabili**

Le variabili dovranno essere identificate attraverso la classica “camelCase” notation, evitando underscore laddove il nome sia costituito da più parole.

La dichiarazione delle stesse dovrà essere chiara e possibilmente separata dall’inizializzazione per garantire un migliore leggibilità del codice.

## **1.3.4 Costanti**

Le costanti dovranno invece essere identificate attraverso una “UPPER\_CASE” notation, utilizzando underscore laddove il nome sia costituito da più parole.

La dichiarazione delle stesse dovrà essere chiara e coincidente con l’inizializzazione, pertanto non dovranno esserne dichiarate più di una nello stesso rigo, seppur dello sesso tipo, questo per garantire un migliore leggibilità del codice.

## **1.3.5 Metodi**

I metodi dovranno essere identificati attraverso la classica “camelCase” notation.

Il nome dei metodi sarà tipicamente costituito da un verbo che ne specifica l’azione e l’oggetto su cui essa viene applicata.

Tutti i metodi saranno commentati secondo la documentazione JavaDoc.

## **1.3.6 Classi e Pagine**

Le classi e le pagine dovranno essere identificate attraverso la classica “CamelCase” notation, con l’iniziale maiuscola, per distinguerle da attributi e metodi.

Il loro nome dovrà fornire una descrizione intuitiva di quello che è lo scopo ed essere coerente con le operazioni implementate al loro interno.

## **1.4 Acronimi**

# RAD: Requirements Analysis Document

# SDD: System Design Document

# ODD: Object Design Document

# OCL: Object Constraints Language

# DB: Database

# DAO: Data Object Access

## **1.5 Riferimenti**

* Bern Bruegge, Allen H. Dutoit, Object-Oriented Software Engineering - Using UML, Patterns, and JAVA, 3rd edition.
* RAD\_RistoManager
* SDD\_RistoManager

# 3. Packages

## **3.1 Control**

|  |  |
| --- | --- |
| Subpackage: Control.Comanda | |
| CLASSE | DESCRZIONE |
| AggiornaQuantitaComanda | Permette di aggiornare la quantità di un prodotto della comanda |
| AggiungiProdottoComanda | Permette di aggiungere un prodotto alla comanda |
| InviaComanda | Permette di inviare la comanda alla cucina |
| RimuoviProdottoComanda | Permette di rimuovere un prodotto dalla comanda |
| VisualizzaComanda | Permette di visualizzare la comanda |

|  |  |
| --- | --- |
| Subpackage: Control.Cucina | |
| CLASSE | DESCRZIONE |
| CucinaControl | Permette di gestire le operazioni della cucina, quali accettare una comanda ricevuta, visualizzarla e completarla |

|  |  |
| --- | --- |
| Subpackage: Control.Menu | |
| CLASSE | DESCRZIONE |
| AggiungiProdotto | Permette di aggiungere un prodotto al menu |
| GeneraProdottoCasuale | Permette di generare un prodotto casuale dal menu |
| ModificaProdotto | Permette di modificare gli attributi di un prodotto |
| RimuoviProdotto | Permette di rimuovere un prodotto dal menu |
| SingoloProdotto | Permette di visualizzare un singolo prodotto |
| VisualizzaMenu | Permette di visualizzare il menu di tutti i prodotti |
| VisualizzaProdotti | Permette di visualizzare tutti i prodotti per la gestione |
| VisualizzaProdottiCategoria | Permette di visualizzare i prodotti filtrati per categoria |
| VisualizzaProdottiIngredienti | Permette di visualizzare i prodotti filtrati per ingredienti |
| VisualizzaProdottiPrezzo | Permette di visualizzare i prodotti filtrati per prezzo |

## **3.1.1 Control.Utente**

|  |  |
| --- | --- |
| CLASSE | DESCRZIONE |
| LoginControl | Permette ai membri dello staff di effettuare il login |
| LogoutControl | Permette ai membri dello staff di effettuare il logout |

|  |  |
| --- | --- |
| Subpackage: Control.Utente.Cliente | |
| CLASSE | DESCRZIONE |
| ConfermaPrenotazione | Permette di confermare la prenotazione attraverso il link ricevuto tramite e-mail |
| DispatchCliente | Permette di reindirizzare il cliente dopo alla pagina di registrazione o prenotazione |
| PrenotazioneTavolo | Permette al cliente di prenotarsi |
| Registrazione | Permette al cliente di registrarsi |
| SingoloProdotto | Permette di visualizzare un singolo prodotto |

|  |  |
| --- | --- |
| Subpackage: Control.Utente.Gestione | |
| CLASSE | DESCRZIONE |
| EliminaUtente | Permette di rimuovere l’account di un membro dello staff dalla piattaforma |
| VisualizzaDati | Permette di visualizzare i clienti del locale in un certo range di date |
| VisualizzaUtenti | Permette visualizzare i membri dello staff registrati alla piattaforma |
| Subpackage: Control.Utente.Sala | |
| CLASSE | DESCRZIONE |
| GeneraCodice | Permette di generare il codice di un tavolo per un cliente |
| VisualizzaCodicePrenotato | Permette di visualizzare il codice associato ad un cliente prenotato |

|  |  |
| --- | --- |
| Subpackage: Control.Utente.Filter | |
| CLASSE | DESCRZIONE |
| CucinaFilter | Controlla se un utente non autorizzato sta cercando di accedere all’area riservata al personale di cucina |
| GestioneFilter | Controlla se un utente non autorizzato sta cercando di accedere all’area riservata al personale di gestione |
| SalaFilter | Controlla se un utente non autorizzato sta cercando di accedere all’area riservata al personale di sala |

## **3.2 Model**

|  |  |
| --- | --- |
| CLASSE | DESCRZIONE |
| AcountStaffBean | Rappresente l’account di un membro dello staff |
| ClienteBean | Rappresenta un cliente del locale |
| ComandaBean | Rappresenta la comanda di un cliente |
| ComandaItemBean | Rappresenta un prodotto inserito sulla comanda |
| ProdottoBean | Rappresenta un prodotto del menu |
| RiepilogoBean | Rappresenta il riepilogo finale di un cliente, comprensivo di tutte le comande |

## **3.3 DAO**

|  |  |
| --- | --- |
| CLASSE | DESCRZIONE |
| AcountStaffDAO | Modella le interazioni di un membro dello staff con il database |
| ClienteDAO | Modella le interazioni di un cliente del locale con il database |
| ComandaDAO | Modella le interazioni con la comanda di un cliente nel database con il database |
| ProdottoDAO | Modella le interazioni con un prodotto del menu con il database |
| DriverManagerConnectionPool | Instanzia e restituisce le connessioni al database |

## **3.4 Test**

|  |  |
| --- | --- |
| CLASSE | DESCRZIONE |
| AllTests | Suite che esegue tutti i test case |

|  |  |
| --- | --- |
| Subpackage: Test.Control | |
| CLASSE | DESCRZIONE |
| TestAggiungiProdotto | Permette di testare le operazioni fornite dalla classe AggiungiProdotto |
| TestDispatchCliente | Permette di testare le operazioni fornite dalla classe DispatchCliente |
| TestLogin | Permette di testare le operazioni fornite dalla classe LoginControl |
| TestModificaProdotto | Permette di testare le operazioni fornite dalla classe ModificaProdotto |
| TestPrenotazioneTavolo | Permette di testare le operazioni fornite dalla classe PrenotazioneTavolo |
| TestRegistrazione | Permette di testare le operazioni fornite dalla classe Registrazione |
| TestVisualizzaCodicePrenotato | Permette di testare le operazioni fornite dalla classe VisualizzaCodicePrenotato |

|  |  |
| --- | --- |
| Subpackage: Test.DAO | |
| CLASSE | DESCRZIONE |
| TestAcountStaffDAO | Permette di testare le operazioni fornite dalla classe AccountStaffDAO |
| TestClienteDAO | Permette di testare le operazioni fornite dalla classe ClienteDAO |
| TestComandaDAO | Permette di testare le operazioni fornite dalla classe ComandaDAO |
| TestProdottoDAO | Permette di testare le operazioni fornite dalla classe ProdottoDAO |

## **3.5 Utils**

|  |  |
| --- | --- |
| CLASSE | DESCRZIONE |
| CodeGenerator | Permette di generare il codice per un cliente |
| EmailBodyGenerator | Permette di generare il corpo di una e-mail da inviare al cliente per confermare la prenotazione |

## **3.6 View**

|  |  |
| --- | --- |
| URL: RistoManager/ | |
| PAGINA | DESCRZIONE |
| 404.jsp | Pagina di errore in caso di errori nell’url |
| accedi.jsp | Pagina per inserire il codice del tavolo |
| comanda.jsp | Pagina per visualizzare la comanda di un cliente e inviarla |
| confermato.jsp | Pagina che notifica la conferma della prenotazione al cliente |
| index.jsp | Pagina di indice che permette di prenotare o ordinare |
| login.jsp | Pagina per effettuare il login da parte dei membri dello staff |
| menu.jsp | Pagina per visualizzare il menu dei prodotti |
| prenotazione.jsp | Pagina per effettuare la prenotazione del tavolo |
| prodotto.jsp | Pagina di informazioni dettagliate su un singolo prodotto |
| registrazione.jsp | Pagina per registrare i dati di un cliente |
| riepilogocomanda.jsp | Pagina per visualizzare il riepilo della comanda di un cliente |
| thankyou.jsp | Pagina di ringraziamento per un cliente che si è prenotato |

|  |  |
| --- | --- |
| URL: RistoManager/cucina/ | |
| PAGINA | DESCRZIONE |
| index.jsp | Pagina di indice per il personale di cucina, mostra tutte le comande |
| dettaglio.jsp | Pagina per visualizzare una comanda nel dettaglio |

|  |  |
| --- | --- |
| URL: Ristomanager/gestione/ | |
| PAGINA | DESCRZIONE |
| index.jsp | Pagina di indice per il personale di gestione |
| clienti.jsp | Pagina per visualizzare i clienti in un certo range di date |
| modifica.jsp | Pagina per modificare un prodotto del menu |
| nuovoProdotto.jsp | Pagina per inserire un nuovo prodotto nel menu |
| personale.jsp | Pagina per visualizzare i membri del personale registrati |
| prodotti.jsp | Pagina per visualizzare i prodotti del menu ed effettuare operazioni |

|  |  |
| --- | --- |
| URL: Ristomanager/sala/ | |
| PAGINA | DESCRZIONE |
| index.jsp | Pagina di indice per il personale di sala |
| genera.jsp | Pagina per generare un codice tavolo per un cliente |
| cercaCodice.jsp | Pagina per cercare il codice tavolo di un cliente prenotato, data in input l’e-mail |

4. Class Interfaces

# 5. Class Diagram Completo