­

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Object Design Document  Progetto  SalernArte   |  |  | | --- | --- | | Versione | 0.1 | | Data | 28/05/2022 | | Destinatario | Studenti di Ingegneria del Software 2021/22 | | Presentato da | Martino Lucia [0512105234], Longo Marco [0512105945], Della Pepa Alessia [0512105720] | | Approvato da |  | |

Membri del Team

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome** | **Matricola** | **Acronimo** | **Informazioni di contatto** |
| Della Pepa Alessia | 0512105720 | DPA | a.dellapepa5@studenti.unisa.it |
| Longo Marco | 0512105945 | LM | m.longo36@studenti.unisa.it |
| Martino Lucia | 0512105234 | ML | l.martino11@studenti.unisa.it |

RevisionHistory

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autori** |
| 28/05/2022 | 0.1 | Prima stesura: struttura del documento e inserimento del package SalernArte | DPA, ML, LM |
| 30/05/2022 | 0.2 | Inserimento package gestione acquisti e class interface package gestione acquisti | LM |

Sommario

[Membri del Team 2](#_Toc104628762)

[RevisionHistory 2](#_Toc104628763)

[1. Introduzione 4](#_Toc104628764)

[1.1. Object Design goals 4](#_Toc104628765)

[1.2. Linee guida per la documentazione dell’interfaccia 4](#_Toc104628766)

[1.3. Definizioni, Acronimi e Abbreviazioni 4](#_Toc104628767)

[1.4. Riferimenti 4](#_Toc104628768)

[2. Packages 4](#_Toc104628769)

[3. Class Interfaces 6](#_Toc104628770)

[7. Design Patterns 6](#_Toc104628771)

[8. Glossario 6](#_Toc104628772)

1. Introduzione

## Object Design goals

## Linee guida per la documentazione dell’interfaccia

## Definizioni, Acronimi e Abbreviazioni

|  |  |
| --- | --- |
| **Acronimo/Abbreviazione** | **Definizione** |
| Package | Raggruppamento di classi, interfacce o file correlati |
| Design pattern | Template di soluzioni a problemi ricorrenti impiegati per ottenere riuso e flessibilità |
| Interfaccia | Insieme di signature delle operazioni offerte dalla classe |
| 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 | E’ 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 | E’ la pratica di scrivere frasi in modo tale che ogni parola o abbreviazione inizi con una lettera maiuscola, senza spazi o punteggiatura intermedi |
| Javadoc | Sistema di documentazione offerto da Java, che viene generato sottoforma di interfaccia in modo da rendere la documentazione accessibile e facilmente leggibile. |

## Riferimenti

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

1. Packages

In questa sezione viene mostrata la suddivisione del sistema in package, in base a quanto definito nel documento di System Design. Tale suddivisione è motivata dalle scelte architetturali prese e ricalca la struttura di directory standard definita da Maven

**Package SalernArte**

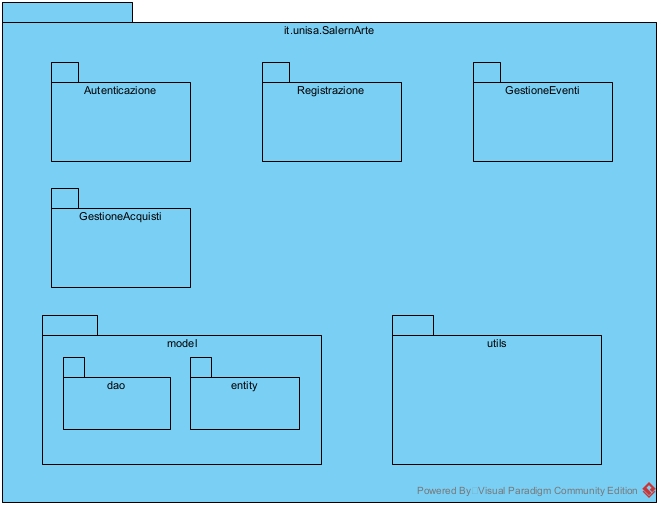
Nella presente sezione si mostra la struttura del package pricipale di SalernArte. La struttura generale è stata ottenuta a partire da tre principali scelte:

1. Creare un package separato per ogni sottosistema, contenente le classi service e controller del sottosistema, ed eventuali classi di utilità usate unicamente da esso.

2. Creare un package separato per le classi del *model*, contenente le classi entity e i DAO per l’accesso al DB. Tale scelta è stata presa vista l’elevata complessità del database di SalernArte che prevede numerose relazioni tra le entità. Si è quindi preferito tenere tutto in un package separato e collegato a tutti gli altri package dei sottosistemi.

3. Creare un package chiamato *utils* in cui inserire eventuli classi di utlità per il sistema e usabili da più sottosistemi.

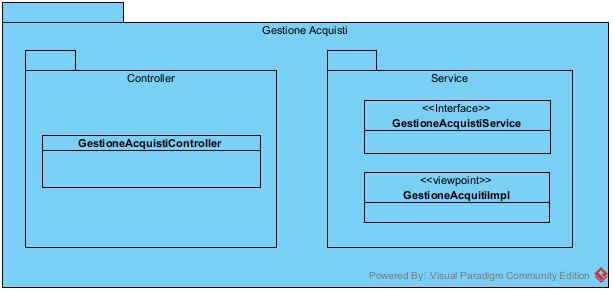
Per ciò che concerne la dipendenza tra i packages, la suddivisione precedentemente illustrata a portato alla creazione di una relazione tra il package model e tutti gli altri package del sistema.



REGISTRAZIONE AUTENTICAZIONE LUCIA

GESTIONE EVENTI ALESSIA

**Package Gestione Acquisti**



1. Class Interfaces

Di seguito saranno presentate le interfacce di ciascun package:

1. REGISTRAZIONE AUTENTICAZIONE LUCIA
2. GESTIONE EVENTI ALESSIA

**Package Gestione Acquisti**

|  |  |
| --- | --- |
| **Nome Classe** | **GestioneAcquistiService** |
| **Descrizione** | Questa classe permette di gestire le operazioni riguardanti gli acquisti degli eventi |
| **Metodi** | +acquistaBiglietti(List<Evento> listaeventi, Integer id): UtenteRegistrato  +modificaCarrello(Integer quantità, Integer idEvento): UtenteRegistrato |
| **Invariante di classe** | / |

|  |  |
| --- | --- |
| **Nome Metodo** | +acquistaBiglietti(List<Evento> listaeventi, Integer id) |
| **Descrizione** | Questo metodo permette di acquistare gli eventi nel carrello con la relativa quantità di ogni evento. |
| **Pre-condizione** | listaeventi.get(i).getQuantità > 0 |
| **Post-condizione** | setOrdersAccount(Integer id, Ordine o)  “lista ordini account +1; quantità di ogni evento iniziale – quantità di ogni evento acquistato”. |

|  |  |
| --- | --- |
| **Nome Metodo** | + modificaCarrello(Integer quantità, Integer idEvento) |
| **Descrizione** | Questo metodo permette di modificare la quantità di ogni singolo evento del carrello |
| **Pre-condizione** | listaeventi.get(i).getQuantità > 0 &&  quantità < listaeventi.get(i).getQuantità |
| **Post-condizione** | / |

1. Design Patterns
2. Glossario