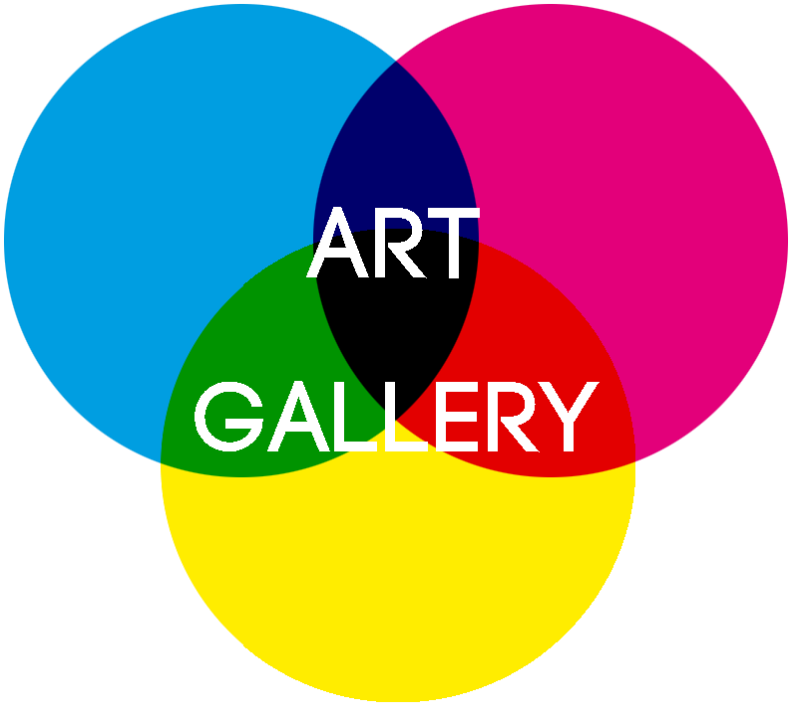
Università degli Studi Di Salerno
Corso di Ingegneria del Software

ArtGallery
Object Design Document
Versione 1.0



DATA 15/12/2020

Coordinatori del progetto

Nome
De Lucia Andrea
Pecorelli Fabiano

Partecipanti al progetto

Nome	Matricola
Emanuele Fittipaldi	05121-05816
Fedele Mauro	05121-04496
Giuseppe Martinelli	05121-05972

Scritto da	Emanuele Fittipaldi, Giuseppe Martinelli, Fedele Mauro

REVISION HISTORY

Data	Versione	Descrizione	Autore
15/12/2020	1.0	Introduction	Emanuele Fittipaldi
15/12/2020	1.0	Object design trade-offs	
15/12/2020	1.0	Interface documentation guidelines	
15/12/2020	1.0	Definitions, acronyms and abbreviations	
15/12/2020	1.0	Design Pattern	Fedele Mauro
15/12/2020	1.0	Packages	Giuseppe Martinelli
15/12/2020	1.0	Class Interfaces	

Object Design Document

Sommario

1. INTRODUCTION

1.1 OBJECT DESIGN TRADE-OFFS

Di seguito sono riportare delle considerazioni sui trade-off che sono stati individuati durante la progettazione del sistema.

Spazio di memoria vs. Tempi di risposta

Nonostante Il nostro sistema sia incentrato sulla gestione di quadri e quindi sulla gestione di centinaia, migliaia di immagini raffiguranti le opere messe presenti nel sistema, si avrà che lo spazio che il trade-off migliore per non degradare i tempi di risposta è imporre un limite sulla dimensione dell'immagine del prodotto. Abbiamo individuato questo limite intorno ad 1 Mb il quale ci consente di gestire meglio lo spazio e rendere i caricamenti delle immagini dei prodotti più rapide, anche per le connessioni più lente.

Prestazioni vs. Costi

Per alcune funzionalità ci siamo affidati a componenti off-the-shelf in modo da non dover reinventare la ruota rischiando di fornire un servizio peggiore di qualcosa di già esistente e funzionante, e soprattutto per limitare i costi che ne sarebbero derivati dalla progettazione delle stesse ad hoc, in approccio Greenfield. Nello specifico, per la gestione delle transazioni nel tratto finale del processo di acquisto, ci siamo affidati a gestori delle finanze esterni, uno tra questi Paypal.

Interfaccia vs. Usabilità

Il trade-off che siamo stati portati a fare è stato quello tra l'aspetto visivo del sistema, per essere più eye-catching al fine di indurre l'utente all'acquisto di un'opera, che ricordiamo essere l'obiettivo finale del sistema, e quello di rendere l'interfaccia il più esente possibile da distrazioni, testi superflui, e informazioni ambigue, in modo da guidare in un certo senso l'utente attraverso la navigazione delle pagine e durante l'acquisto di un'opera. Questo approccio ci garantisce anche un sistema meno prono a dover gestire errori e quindi a subire tutto l'overhead che ne deriverebbe da continue eccezioni.

Sicurezza vs. Efficienza

Noi di ArtGallery vogliamo garantire sempre la massima sicurezza ai nostri utenti, perché siamo consapevoli della fiducia che viene riposta in un sistema informativo, soprattutto durante l'acquisto di un bene, e vogliamo mantenere intatta tale fiducia nei confronti dei nostri clienti. Per garantire sicurezza abbiamo adottato dei meccanismi di crittografia dei dati considerati sensibili, come per esempio, i numeri di carte di credito e le password scelte dagli utenti iscritti. Questo significa che ogniqualvolta che c'è una comunicazione di questi dati da e per l'utente, viene eseguito un algoritmo incaricato di crittografare e de-crittografare tali dati end-to-end. Questo può compromettere un leggero calo in termini di efficienza, ma siamo del parere che in questo caso, questo è il trade-off migliore, al fine di garantire un servizio affidabile e sicuro.

1.2 INTERFACE DOCUMENTATION GUIDELINES

Per definire le interfacce sono state adottate le seguenti convenzioni. Queste convenzioni costituiranno il punto di riferimento per tutti i team al fine da essere consistenti nella progettazione delle interfacce e per non creare ambiguità in chi dovrà andare a svilupparle, e infine per leggibilità.

1.2.1. Class conventions

I nomi delle classi devono essere soltanto sostantivi, e al singolare. Le classi devono iniziare con una lettera Maiuscola, non devono contenere underscore o trattini o qualsiasi altro carattere speciale.

1.2.2. Methods conventions

I nomi dei metodi devono contenere un verbo (es. restituisciNumero), devono seguire per tale ragione lo stile Camel case per ogni inizio di parola. Nel caso di metodi che non hanno un nome composto, iniziare con una lettera maiuscola (es. Aggiungi).

1.2.3. Parameters conventions

I nomi dei parametri devono essere dei sostantivi, così come i nomi delle variabili. Devono iniziare con una lettera minuscola, e non possono contenere caratteri speciali. Nel caso sia necessario combinare due parole per un parametro o una variabile, usare il Camel case.

1.2.4. Error conventions

Lo stato di errore deve essere ritornato come una eccezione e non come un valore di ritorno.

1.2.5. JSP, HTML, Coding conventions

Le pagine JSP possono contenere codice Java,HTML, JavaScript e devono seguire le buone norme di indentazione. Ogni gruppo di tag che si riferiscono logicamente ad una stessa azione devono essere sullo stesso livello di indentazione come blocco unico. Usare l'indentazione per mostrare anche la gerarchia del codice, impiegando il Tab per operazioni più interne.
Deve essere adottato lo stile di apertura e chiusura delle parentesi in colonna piuttosto che Egyptian, in modo da rendere facilmente comprensibile dove una parentesi è stata aperta e dove è stata chiusa, così anche per i Tag. Come per le JSP, il codice deve rientrare, tramite l'utilizzo di TAB per indicare la gerarchia delle istruzioni e i blocchi logici delle azioni.

1.3 DEFINITIONS, ACRONYMS, AND ABBREVIATIONS

1.3.1 Definizioni

Design pattern: Descrive un problema ricorrente e la soluzione a tale problema al fine da poter applicare questa soluzione nei casi in cui si manifestino problemi per cui esiste un Design pattern che ne descrive già una soluzione. Il design pattern ha quattro elementi:

- Nome: che lo identifica
- Descrizione del problema: che descrive le situazioni in cui il pattern può essere usato.
- Soluzione: la soluzione presentata come un insieme di classi e interfacce
- Conseguenze: un insieme di conseguenze che descrivono i trade-off e che devono essere considerate rispetto ai design goal fissati

Package: collezione di classi e interfacce correlate

MVC: modello architetturale che si basa sull'utilizzo di 3 componenti, Model, View e controller, tipicamente usato per la progettazione e lo sviluppo di applicazioni web.

Class Diagram: Tipo di diagramma che descrive la struttura di un sistema mostrando le sue classi, gli attributi di tali classi, le operazioni e le relazioni tra gli oggetti.

1.3.2 Acronimi e Abbreviazioni

- ODD: Object Design Document
- RAD: Requirement Analysis Document
- SDD: System Design Document

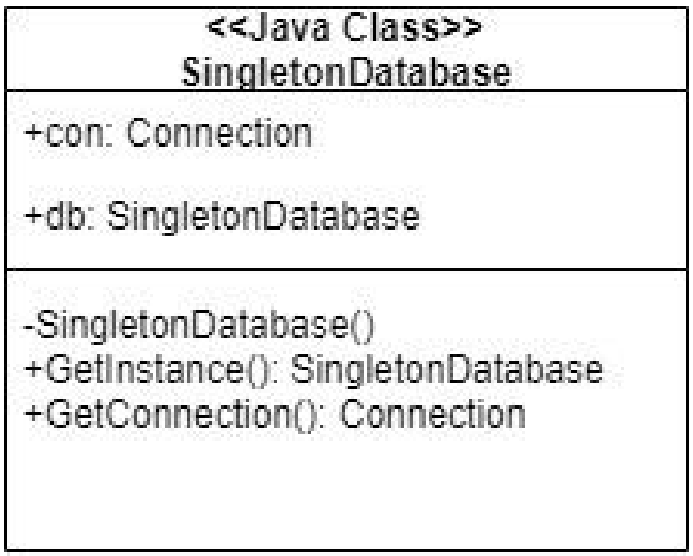
1.3.3. References

- Requirement Analysis Document
- System Design Document

2. DESIGN PATTERN

2.1. Singleton

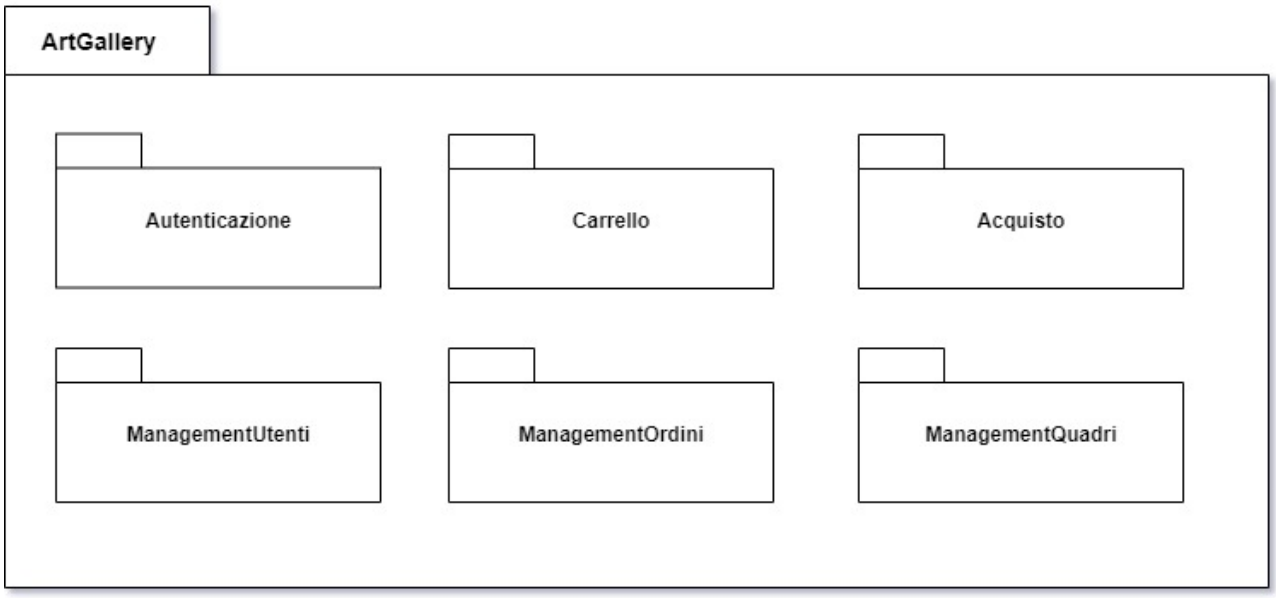
Il singleton è un design pattern creazionale che ha lo scopo di garantire che di una determinata classe venga creata una e una sola istanza, e di fornire un punto di accesso globale a tale istanza. L'implementazione più semplice di questo pattern prevede che la classe singleton abbia un unico costruttore privato, in modo da impedire l'istanziazione diretta della classe. La classe fornisce inoltre un metodo "getter" statico che restituisce l'istanza della classe (sempre la stessa), creandola preventivamente o alla prima chiamata del metodo, e memorizzandone il riferimento in un attributo privato anch'esso statico.
Nel nostro sistema risulta necessario l'utilizzo di una classe che abbia esattamente una sola istanza per quanto riguarda la connessione al database. Infatti tale classe sarà responsabile di tener traccia della sua unica istanza per la connessione al database e per avere i dati sempre aggiornati.



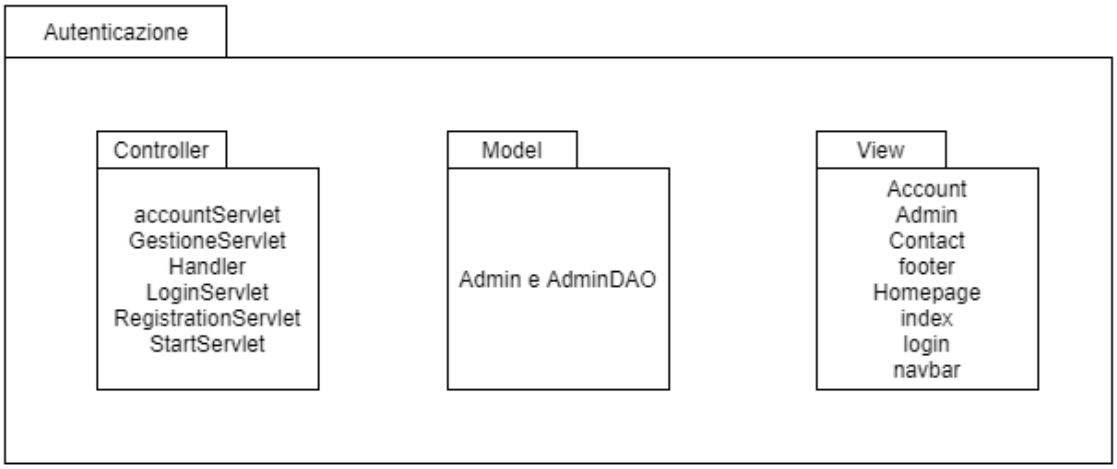
3. PACKAGES

3.1. Diagramma package ArtGallery

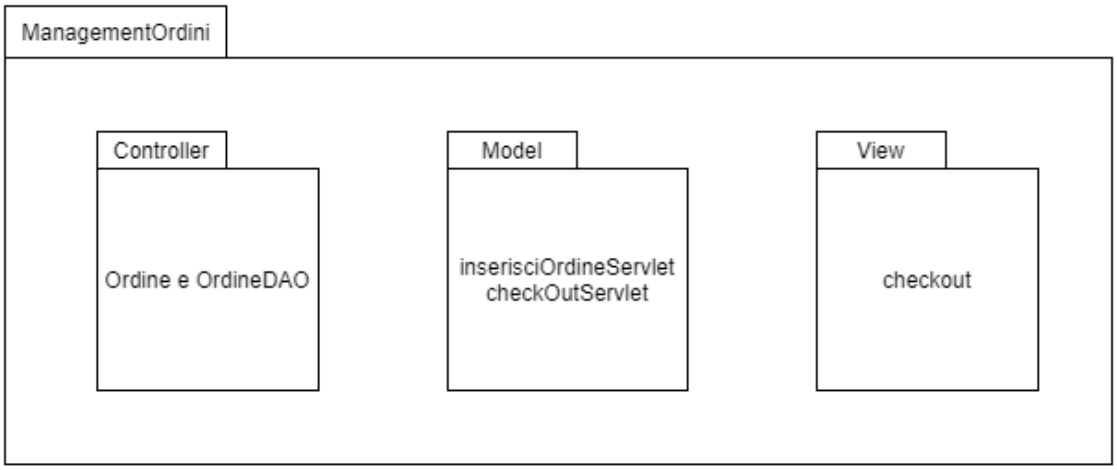
Packages: *Diagramma della suddivisione in packages per i singoli sottosistemi*



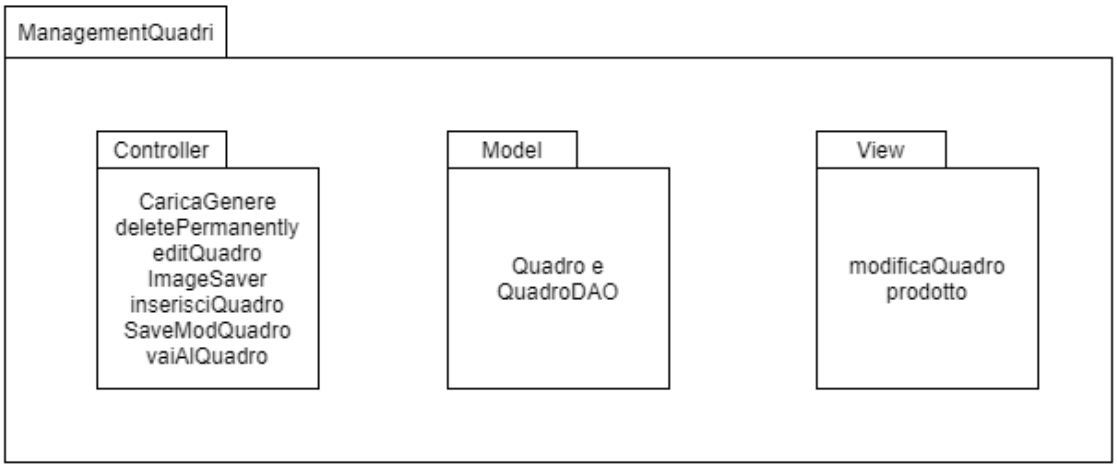
3.2. Diagramma package Autenticazione



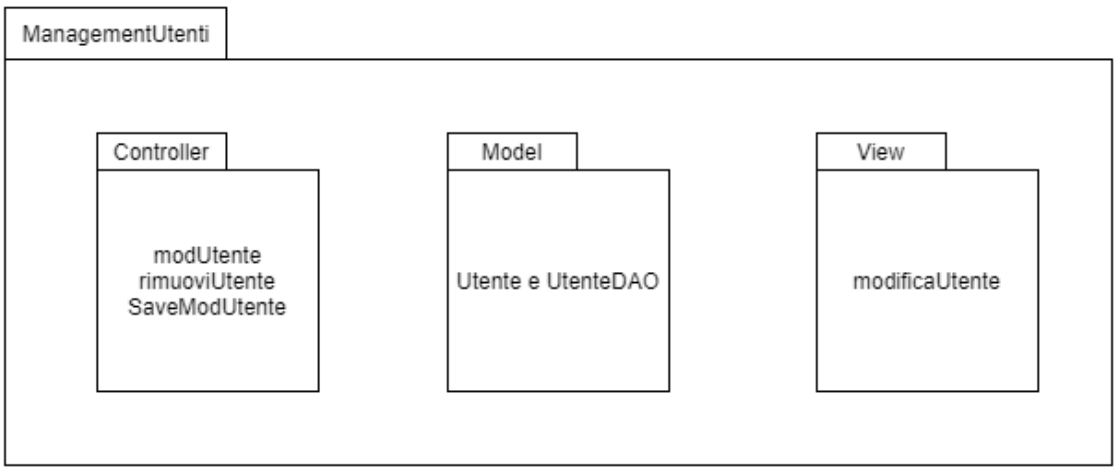
3.3. Diagramma package ManagementOrdini



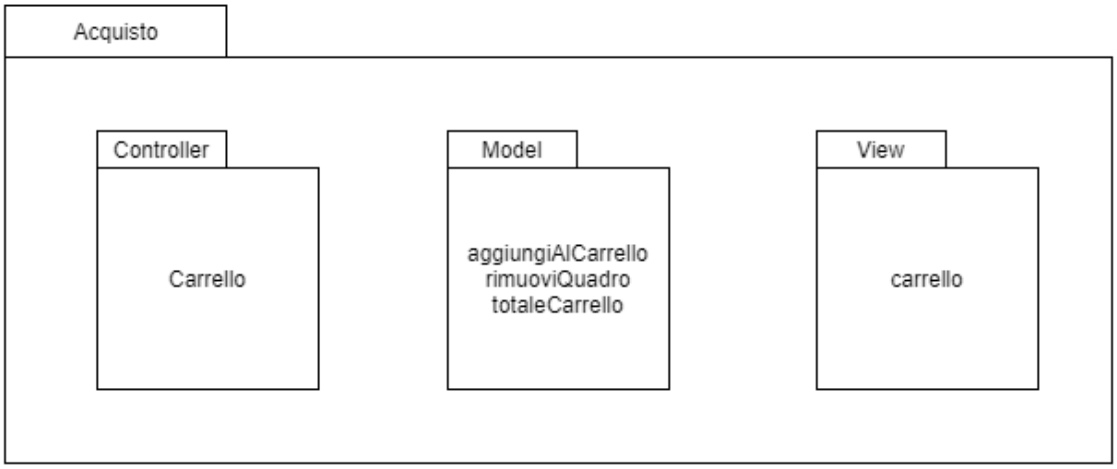
3.4. Diagramma package ManagementQuadri



3.5. Diagramma package ManagementUtenti



3.6. Diagramma package Acquisto



3.7. Diagramma package Carrello

Dipendenze:

AUTENTICAZIONE:

Utilizza il model Utente,UtenteDAO appartenenti al package ManagementUtenti per reperire un determinato utente date le credenziali

ACQUISTO:

Utilizza Quadro, QuadroDAO appartenenti al package di ManagementQuadri per tenere traccia dei quadri inseriti nel carrello

MANAGEMENT ORDINI

Utilizzano Utente e UtenteDAO , Quadro e QuadroDAO rispettivamente appartenente ai packages Management Utenti e Management Quadri per permettere l'associazione di un utente ad un ordine ed ai quadri presenti nell'ordine

CARRELLO

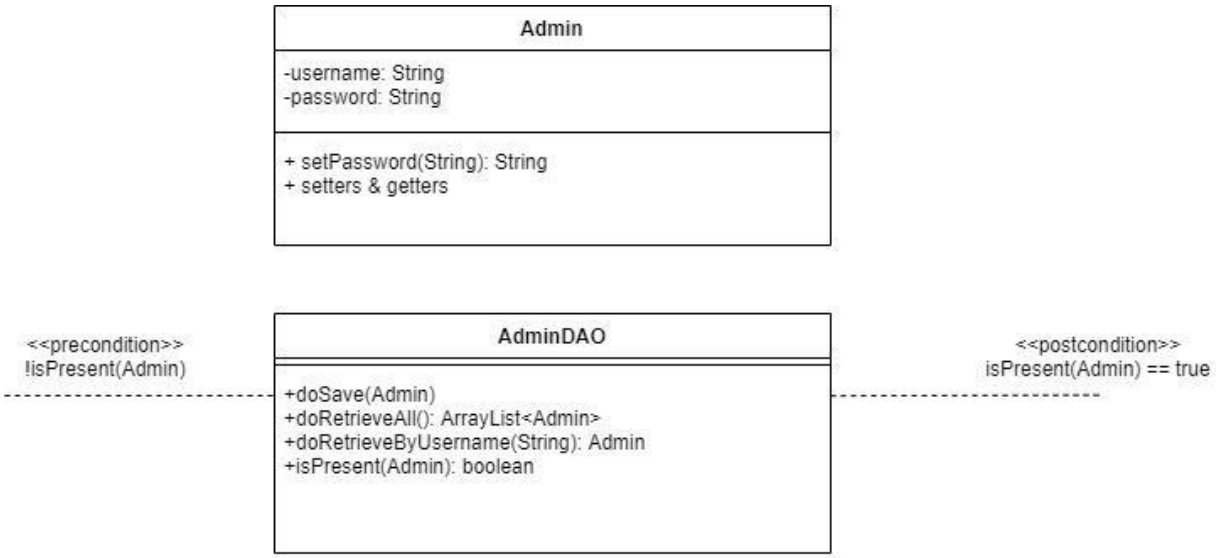
Utilizza Quadro presente all'interno di Management Quadro

4. CLASS INTERFACES GLOSSARY

In questa sezione abbiamo voluto volgere uno sguardo alle classi più complesse di ogni package descrivendone i metodi e le eventuali interfacce

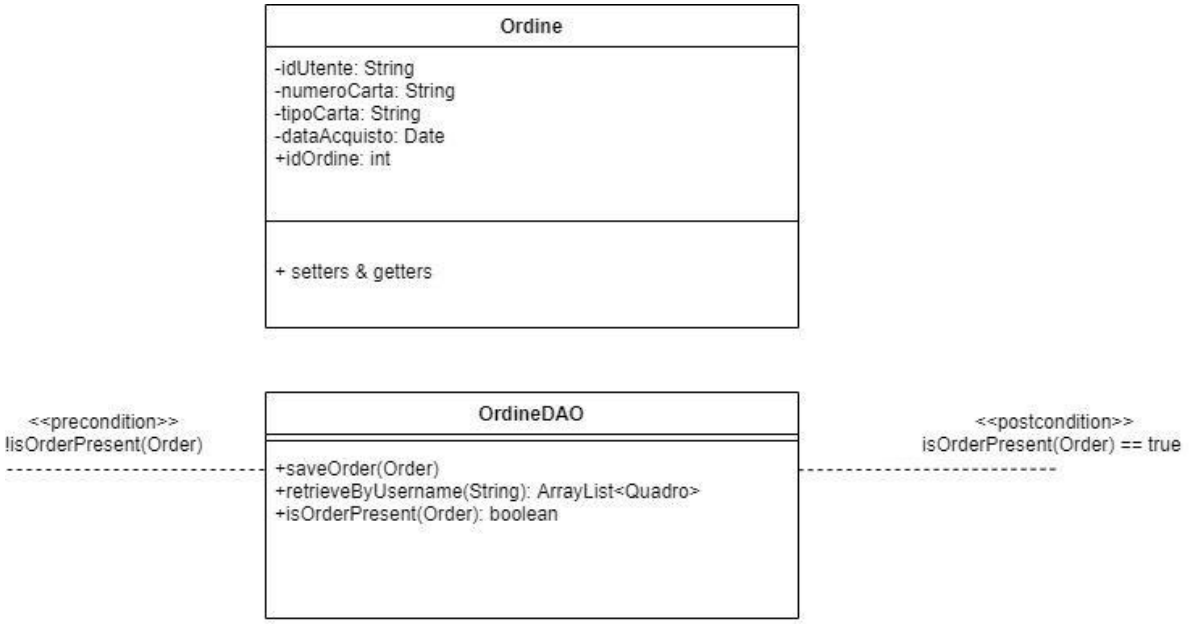
4.1. Package Autenticazione

Package Autenticazione			
Classe	Breve descrizione	Dipendenze eventuali	Eccezioni
Admin	Modella la figura dell'amministratore all'interno del sistema	Null	Null
AdminDAO	Si occupa della gestione della persistenza dell'entità Admin all'interno del database	Admin	RunTimeException



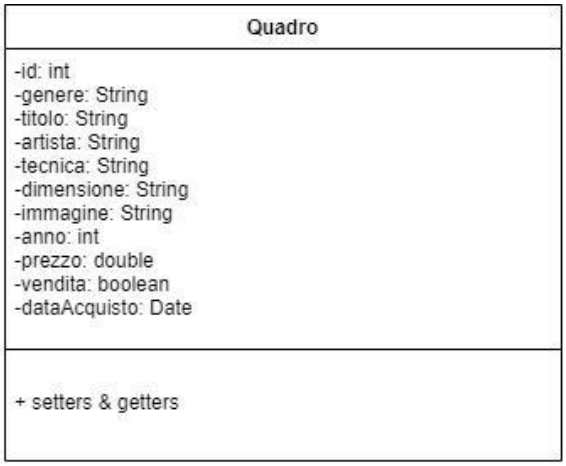
4.2. Package ManagementOrdini

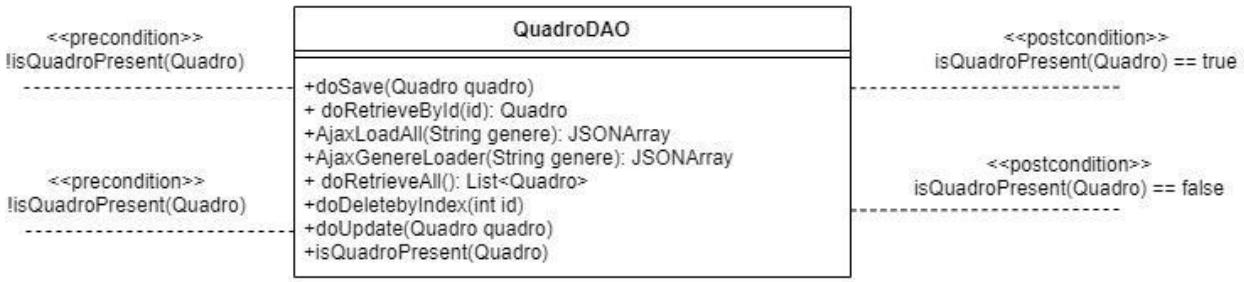
Package ManagementOrdini			
Classe	Breve descrizione	Dipendenze eventuali	Eccezioni
Ordine	Modella la figura dell'ordine all'interno del sistema	Java.sql.Date	Null
OrdineDAO	Si occupa della gestione della persistenza dell'entità Ordine all'interno del database	Ordine	RunTimeException



4.3. Package ManagementQuadro

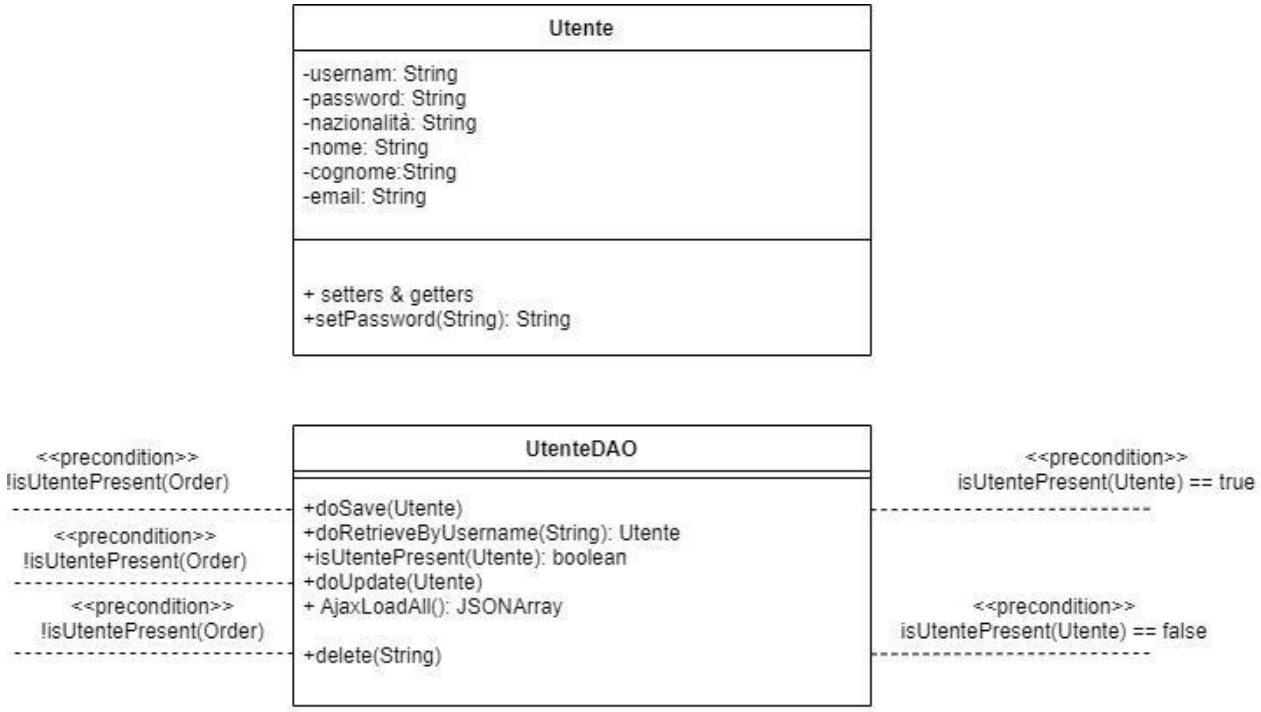
Package ManagementQuadro			
Classe	Breve descrizione	Dipendenze eventuali	Eccezioni
Quadro	Modella la figura del quadro all'interno del sistema	Java.sql.Date	Null
QuadroDAO	Si occupa della gestione della persistenza dell'entità Quadro all'interno del database	JSONArray,Quadro	RunTimeException





4.4. Package ManagementUtenti

Package ManagementUtenti			
Classe	Breve descrizione	Dipendenze eventuali	Eccezioni
Utente	Modella la figura del utente iscritto all'interno del sistema	Java.sql.Date	Null
UtenteDAO	Si occupa della gestione della persistenza dell'entità Utente all'interno del database	Utente,JSONArray	RunTimeException



4.5. Package Carrello

Package Carrello			
Classe	Breve descrizione	Dipendenze eventuali	Eccezioni
Carrello	Modella la figura del carrello di un utente	ManagementQuadro.Quadro	Null