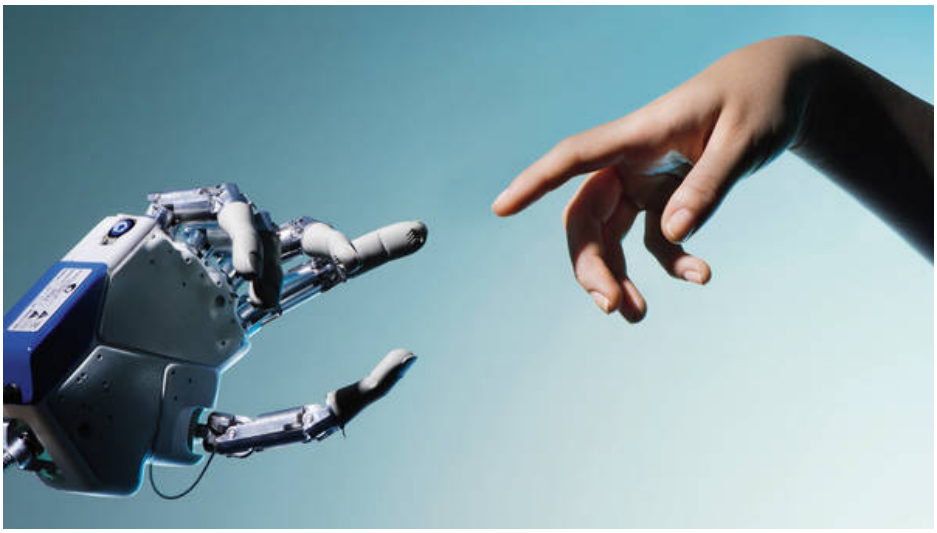
PROGETTO INGEGNERIA DEL SOFTWARE



INTRODUZIONE

Object design trade-offs

Spazio di memoria vs tempi di risposta

Le operazioni di acquisto e controllo degli ordini effettuati al momento si eseguono nel seguente modo: scrivere una lista dei prodotti da acquistare, recarsi all'interno dell'attività e completare l'acquisto.
Per quanto riguarda controllare gli acquisti effettuati c'è bisogno che l'utente conservi gli scontrini con se oppure tenga un registro dei prodotti acquistati.
Il team di analisi e sviluppo conviene sul fatto che,
volendo digitalizzare le operazioni riguardanti l'acquisto di un prodotto caseario ed il controllo degli acquisti effettuati i tempi di risposta saranno più rapidi, essi sono un punto fondamentale su cui premere.
A questo scopo, si sceglie di dar più importanza a questo aspetto, contenendo il quantitativo di memoria utilizzata.

Comprensibilità vs tempi di consegna

Il team di analisi e sviluppo si impegna nello scrivere un codice che sia il più leggibile e comprensibile possibile (tramite l'utilizzo di commenti e tutte le convenzioni che verranno elencate nella sezione delle linee guida).
Questo, ovviamente, potrebbe portare a dei ritardi nei tempi di consegna.
Fermo restando che il team si impegna a consegnare entro i tempi stabiliti, si impegna altresì a rispettare tutte le convenzioni elencate in modo tale da rendere facile la lettura sia al team stesso sia ad altri sviluppatori che lavoreranno col codice in futuro (in caso di manutenzione o aggiornamento).

Robustezza vs tempi di risposta

Il sistema che si va a creare dovrà prevedere ragionevoli soluzioni a situazioni impreviste di errori o malfunzionamenti.
Questo, ovviamente, potrebbe portare a tempi di risposta maggiori.

Componenti off-the-shelf

Le componenti off-the-shelf permettono di creare il nostro sistema hardware/ software.
Nel nostro sistema utilizzeremo diverse tecnologie quali:

1) Java: linguaggio di programmazione Object Oriented;
2) Android studio: sistema Open Source utilizzato per la creazione di applicazioni per dispositivi mobile;
3) MySQL: sistema Open Source di gestione di database relazionali SQL.

Linee guida della documentazione

Le linee guida di questo documento racchiudono le principali regole e convenzioni che il team di sviluppo seguirà durante tutto il processo di sviluppo delle interfacce.

Nomi dei file

- Le classi devono avere nomi al singolare;
- I nomi dei file di configurazione XML devono rispettare i requisiti del framework o tool che li usa.

Dichiarazioni

- Per ogni dichiarazione di variabile locale deve seguire l'inizializzazione nella stessa linea o in quella seguente;
- Ogni dichiarazione di variabile d'istanza deve definire solo una variabile che dovrà essere privata.

Nomenclatura

- I nomi dei metodi devono contenere solo verbi e nomi degli attributi della classe.

Convenzioni

- Per le condizioni e per i cicli dovranno sempre essere utilizzate le parentesi graffe, anche con singoli statement;
- Le condizioni sui valori booleani dovranno seguire il seguente pattern: **if(variabile)** se si vuole controllare che il valore della variabile sia true, **if(!variabile)** viceversa.

Definizioni, acronimi e abbreviazioni

Definizioni

- Off-the-shelf: servizi esterni al sistema di cui viene fatto utilizzo;
- MySQL: database open source che utilizza il linguaggio SQL;

Acronimi

- RAD: Requirements Analysis Document;
- SDD: System Design Document;
- ODD: Object Design Document;
- XML: eXtensible Markup Language;
- SQL: Structured Query Language.

Riferimenti

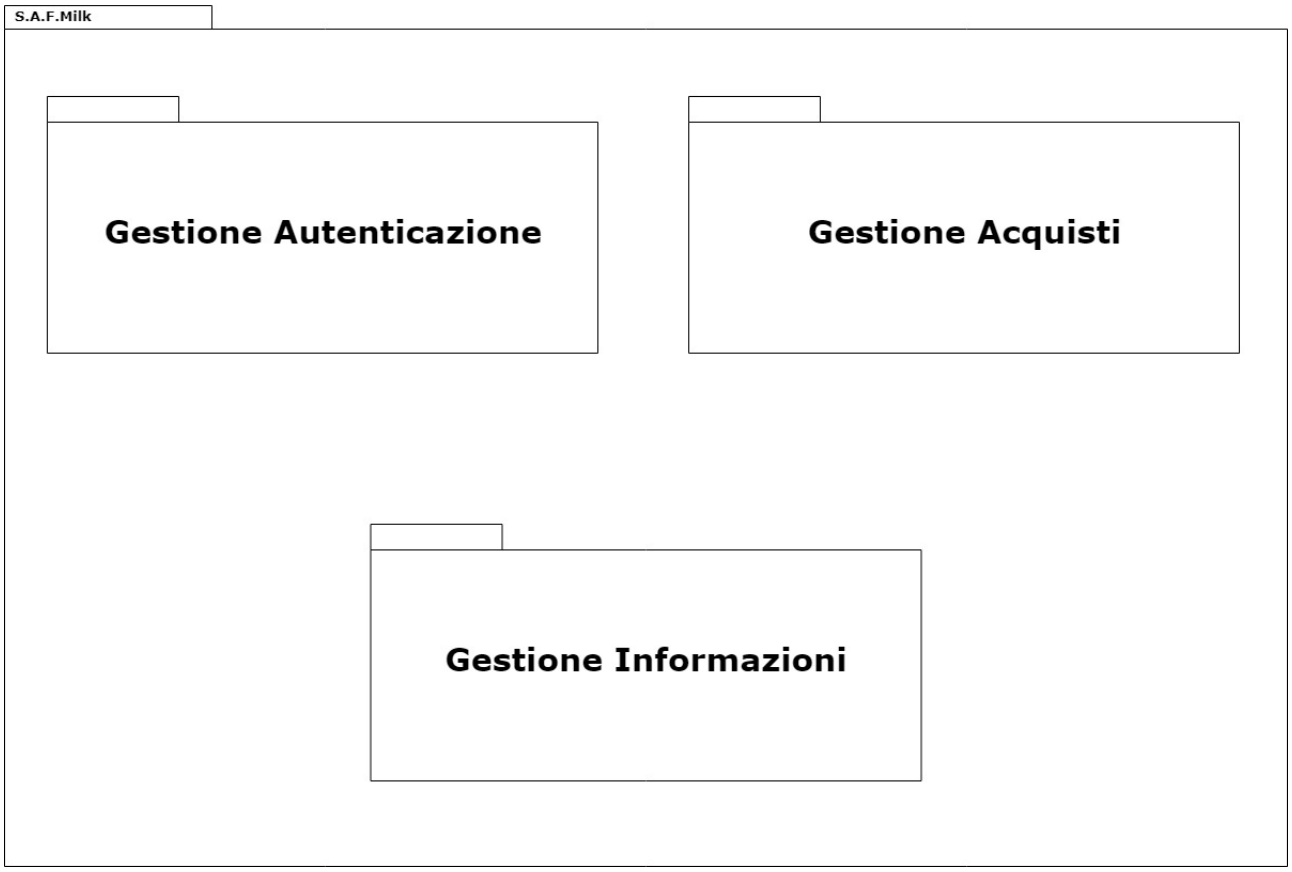
Object-Oriented Software Engineering Using UML, Patterns, and Java - Bernd Bruegge & Allen H. Dutoit- Pearson.

RAD_S.A.F.Milk

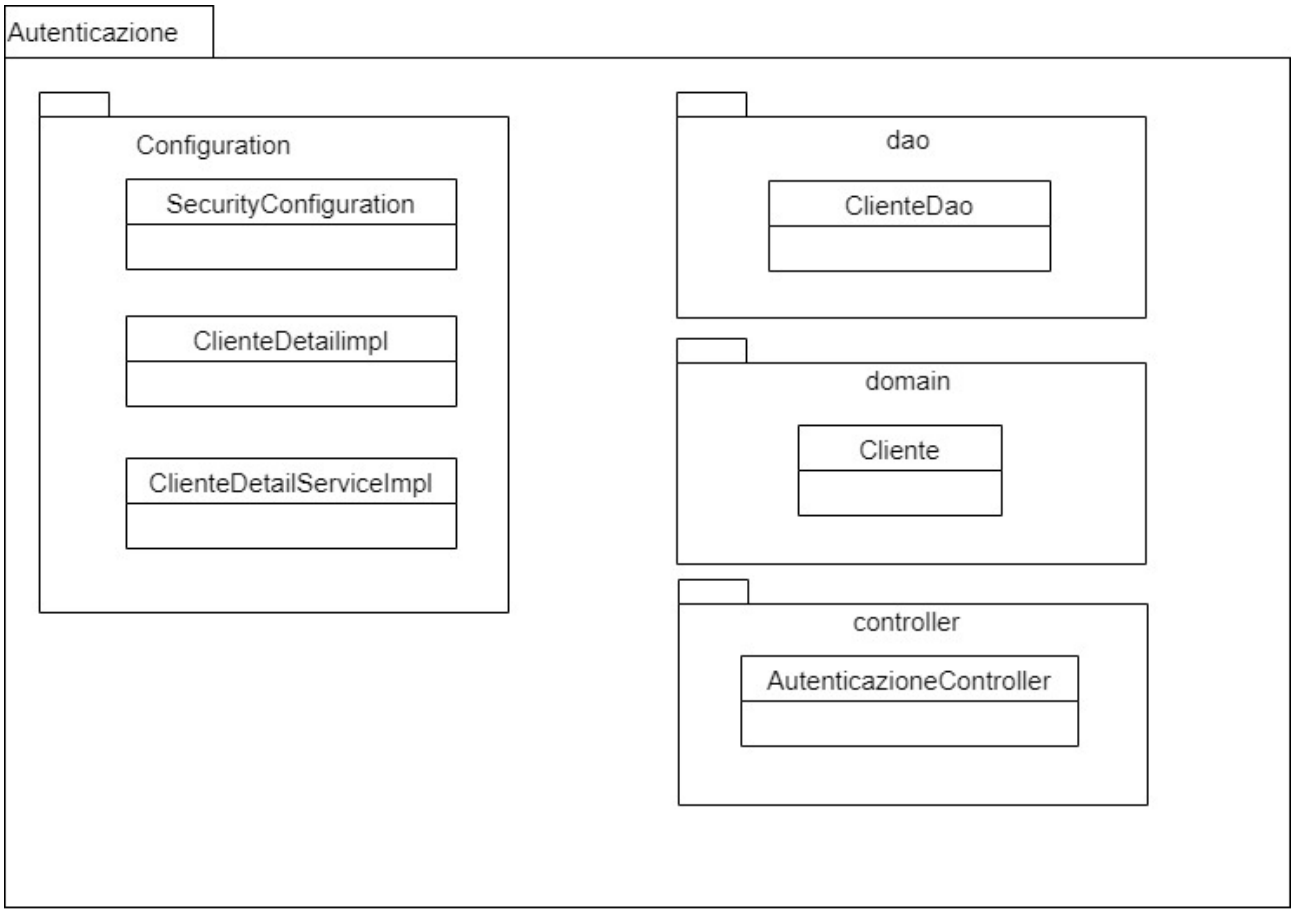
SDD_S.A.F.Milk

PACCHETTI

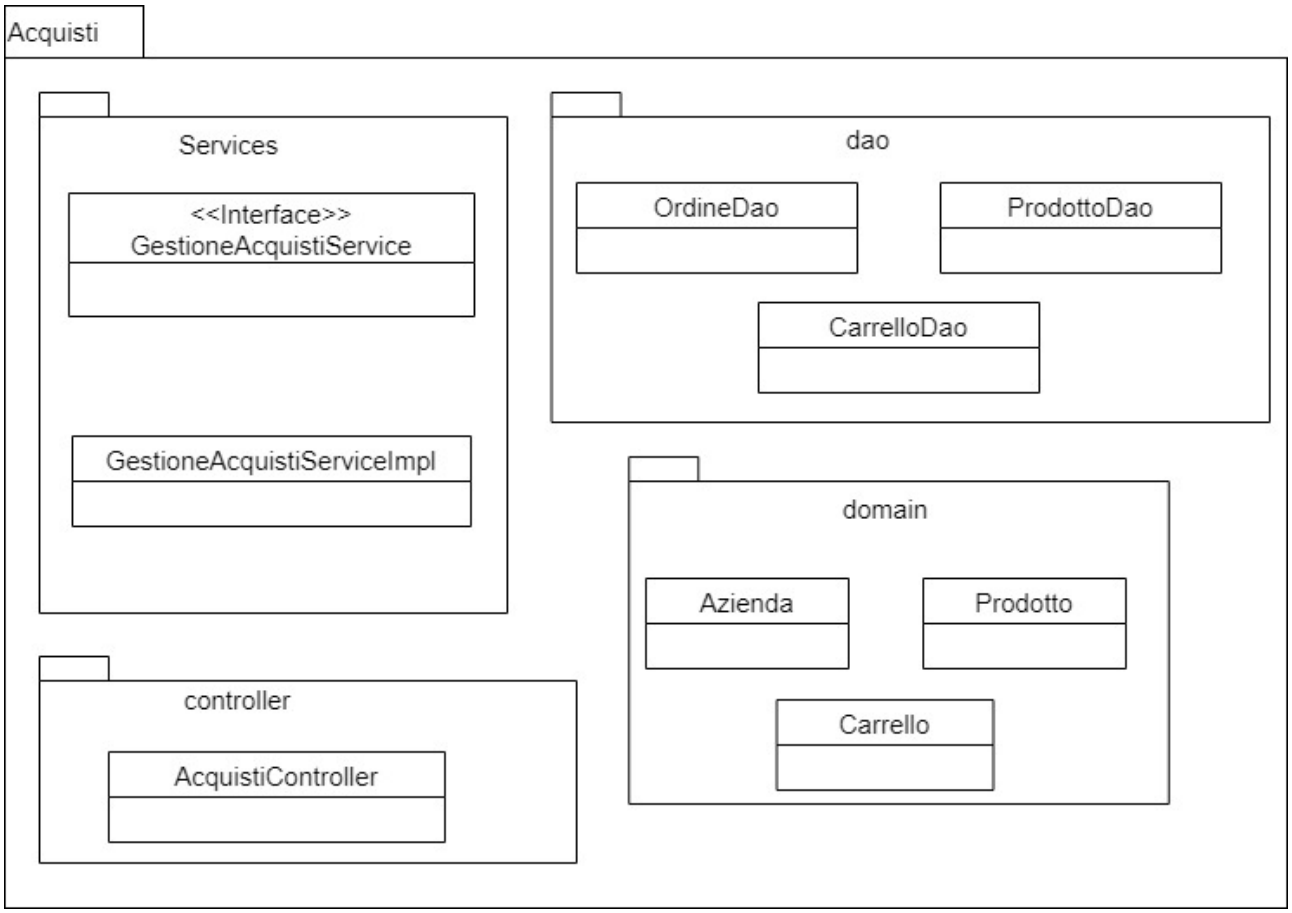
In questa sezione viene mostrata la suddivisione del sistema in package, in base a quanto definito nel documento di System Design.
Viene mostrata prima il top-level, per poi mostrare il dettaglio di ciascun sottopackage, che ricalcano il pattern architetturale Three Layers.



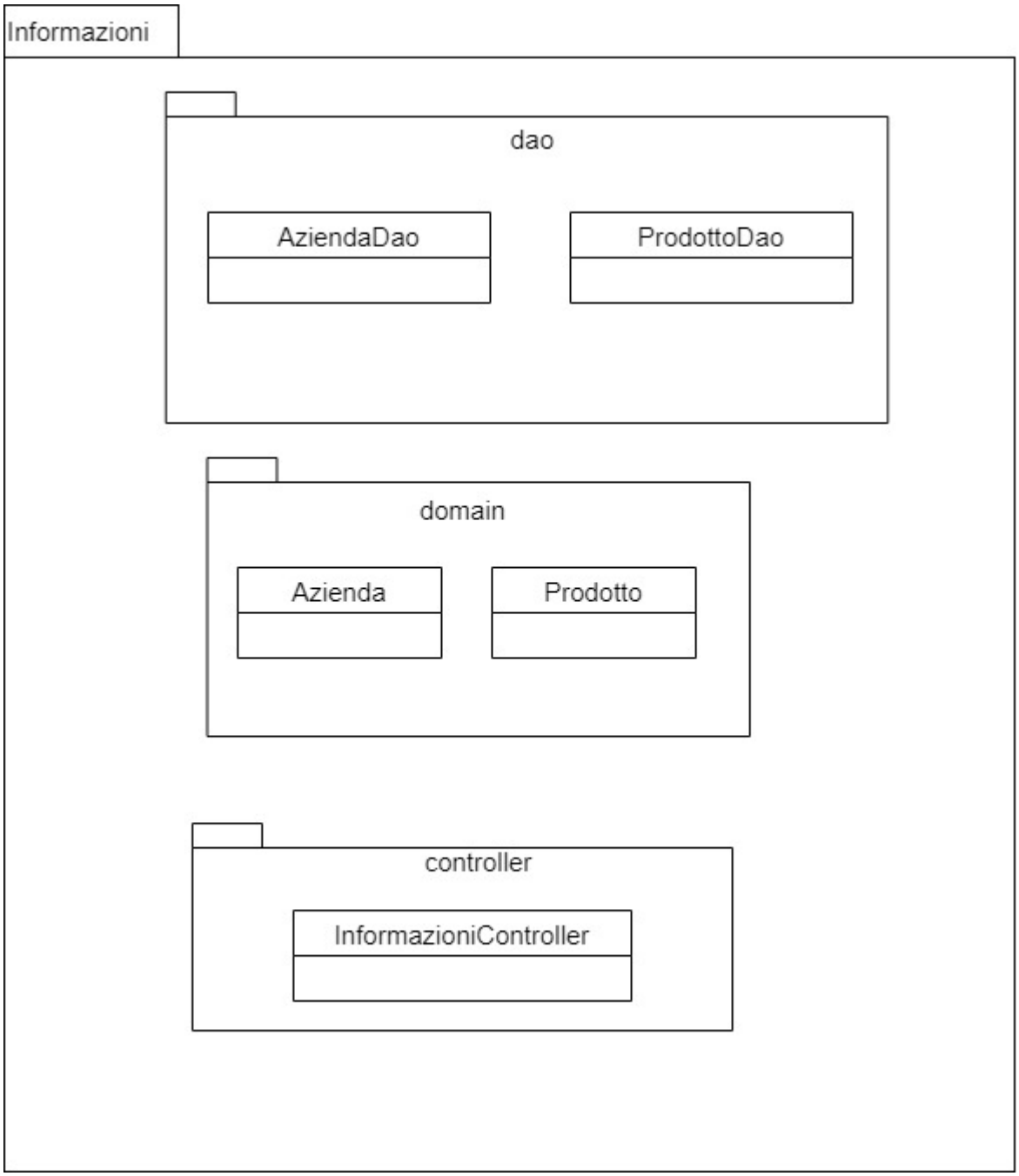
Package autenticazione



Package acquisti



Package informazioni



INTERFACCE DELLE CLASSI

Package autenticazione

NOME CLASSE	ClienteDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Cliente.
METODI	+findByUsername(String username): Cliente +login(String username, String password):Cliente +findAll():List<Cliente> +registrazioneCliente (String nome, String cognome, String cF, String email, String password): void

METODO +findByUsername (String Username): Cliente	
DESCRIZIONE	Questa funzionalità consente di trovare un cliente tramite la sua username.
PRE-CONDIZIONE	**context:** ClienteDAO:: findByUsername(username) **pre:** username!= null
METODO +login (String username, String password): Cliente	
DESCRIZIONE	Questa funzionalità consente l'accesso alla piattaforma da parte del cliente.
PRE-CONDIZIONE	**context:** ClienteDAO:: login(username,password) **pre:** username!= null , password !=null
METODO +findAll ():List <Cliente>	
DESCRIZIONE	Questa funzionalità consente di trovare tutti i

	clienti registrati.
PRE-CONDIZIONE	**context:** ClienteDAO:: findAll()
METODO +registrazione Cliente (String nome, String cognome, String cF, String email, String password): void	
DESCRIZIONE	Questa funzionalità consente di registrare un nuovo cliente.
PRE-CONDIZIONE	**context:** registrazione Cliente (String nome, String cognome, String cF, String email, String password) **pre:** nome != null, cognome != null, cF != null, email != null, password != null

Package acquisti

NOME CLASSE	GestioneAcquistiService
DESCRIZIONE	Questa classe permette la gestione degli acquisti.
METODI	+completaAcquisto(List<Prodotto> object, String titCarta, String numCarta, String cVv): void

METODO +completaAcquisto(List<Prodotto> object, String titCarta, String numCarta, String cVv): void	
DESCRIZIONE	Questa funzionalità consente di completare un acquisto.
PRE-CONDIZIONE	**context:** GestioneAcquistiService:: completaAcquisto(List<Prodotto> object, String titCarta, String numCarta, String cVv) **pre:** object != null, titCarta != null, numCarta != null, cVv != null

NOME CLASSE	OrdineDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Ordine.
METODI	+addOrdine(List<Prodotto> object): void +viewOrdini():List<Prodotto>

METODO +addOrdine(List<Prodotto> object): void	
DESCRIZIONE	Questa funzionalità consente di inserire i prodotti acquistati all'interno della sezione ordini.
PRE-CONDIZIONE	**context:** Ordine:: addOrdine(List<Prodotto> object) **pre:** object != null
METODO +viewOrdini():List<Prodotto>	
DESCRIZIONE	Questa funzionalità consente di visualizzare tutti i prodotti presenti della sezione ordini.
PRE-CONDIZIONE	**context:** Ordine:: addOrdine(List<Prodotto> object) **pre:** object != null

NOME CLASSE	ProdottoDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Prodotto.
METODI	+searchInfoProdotto(Prodotto object): String

METODO +searchInfoProdotto(Prodotto object): String	
DESCRIZIONE	Questa funzionalità consente di ricavare le informazioni di uno specifico prodotto.
PRE-CONDIZIONE	**context:** Prodotto:: searchInfoProdotto(Prodotto object) **pre:** object != null

NOME CLASSE	CarrelloDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Carrello.
METODI	+addProdotto(Prodotto object): void +remouveProdotto(Prodotto object): void +findAllProduct(): List<Prodotto>

METODO +addProdotto(Prodotto object): void	
DESCRIZIONE	Questa funzionalità consente di inserire un prodotto all'interno del carrello.
PRE-CONDIZIONE	**context:** Carrello:: addProdotto(Prodotto object) **pre:** object != null
METODO +remouveProdotto(Prodotto object): void	
DESCRIZIONE	Questa funzionalità consente di rimuovere un prodotto all'interno del carrello.
PRE-CONDIZIONE	**context:** Carrello:: remuveProdotto(Prodotto object) **pre:** object != null
METODO	

+findAllProduct(): List<Prodotto>	
DESCRIZIONE	Questa funzionalità consente di ricavare tutti i prodotti presenti all'interno del carrello.
PRE-CONDIZIONE	**context:** Carrello:: findAllProduct()

Package informazioni

NOME CLASSE	AziendaDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Azienda.
METODI	+viewInfoAzienda(): String

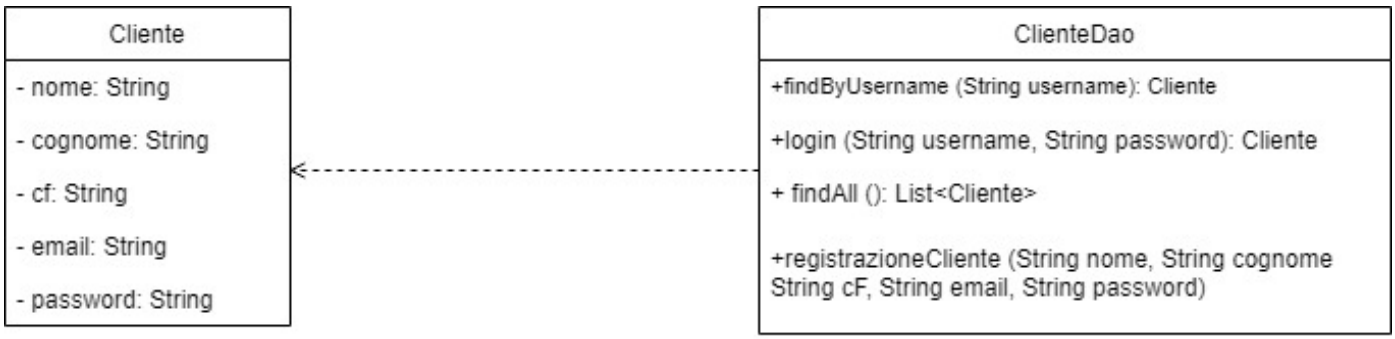
METODO +viewInfoAzienda(): String	
DESCRIZIONE	Questa funzionalità consente di ricavare le informazioni dell'azienda.
PRE-CONDIZIONE	**context:** Carrello:: viewInfoAzienda()

NOME CLASSE	ProdottoDAO
DESCRIZIONE	Questa classe permette di gestire le query relative all'oggetto Prodotto.
METODI	+vewInfoProdotto(Prodotto object): String

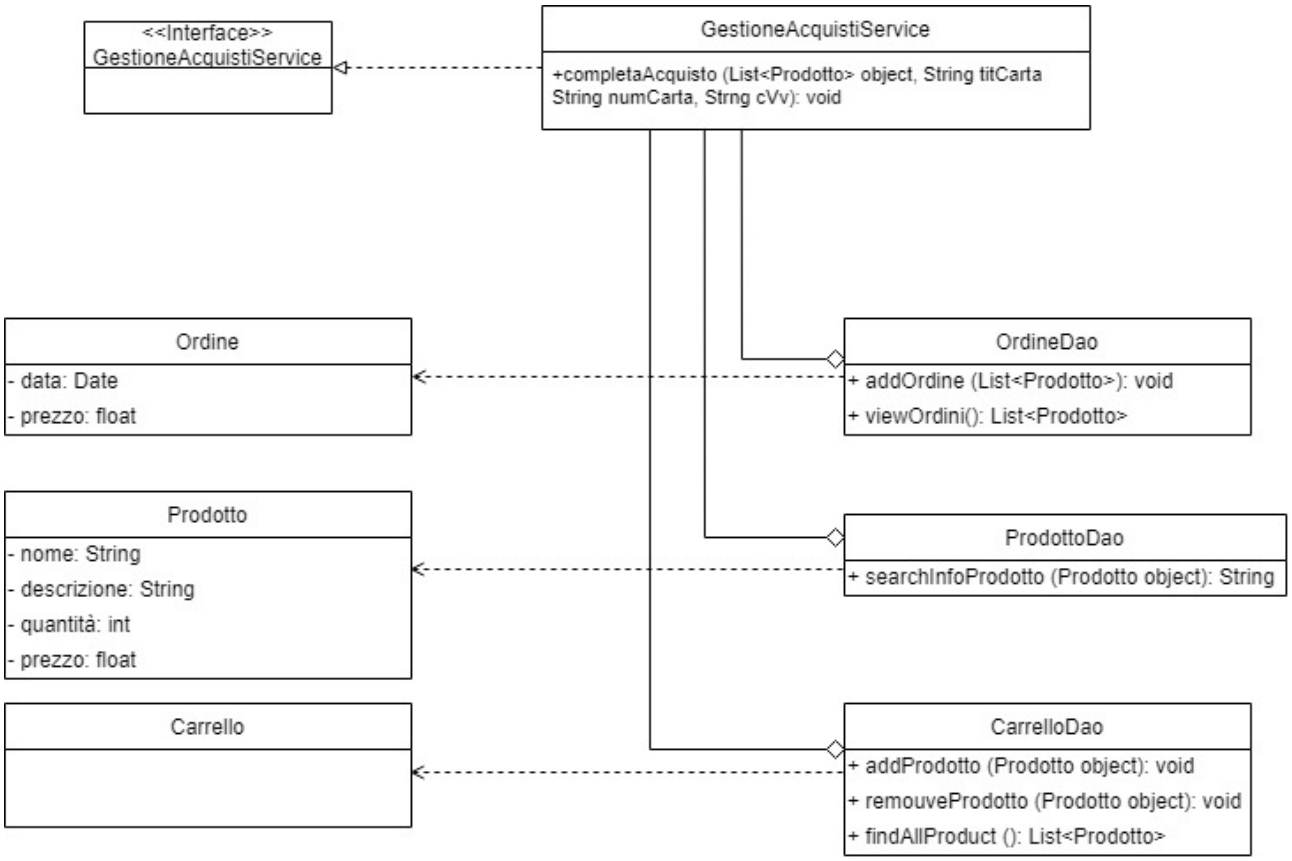
METODO +vewInfoProdotto(Prodotto object): String	
DESCRIZIONE	Questa funzionalità consente di ricavare le informazioni di un singolo prodotto.
PRE-CONDIZIONE	**context:** Carrello:: vewInfoProdotto(Prodotto object) **pre:** object != null

DIAGRAMMA DELLE CLASSI

Autenticazione



Acquisti



Informazioni