**SOFAPIZZA**

**ObjectDesignDocument – ODD**

1. **INTRODUZIONE**
   1. **OBJECT DESIGN TRADE-OFF**
      1. **MODULARITA’ CONTRO EFFICIENZA**
      2. **PRESTAZIONI CONTRO COSTI**
      3. **INTERFACCIA CONTRO EASY-USE**
   2. **INTERFACE DOCUMENTATION GUIDELINES**
      1. **FILE JAVA**
      2. **NAMING**
      3. **ALTRE REGOLE DI STILE**
   3. **DEFINIZIONI, ACRONIMI E ABBREVIAZIONI**
2. **PACKAGES**
3. **INTERFACCE DELLE CLASSI**
   1. **CLASS DIAGRAMS**
   2. **DESCRIZIONE DELLE CLASSI**
      1. **CLIENTE**
      2. **PIZZERIA**
      3. **PRENOTAZIONI**
      4. **PIZZE**

# Introduzione

* 1. ***Object design trade-off***

### Modularità contro efficienza

La modularità definita nel progetto SOFAPIZZA va in contrasto con l’efficienza nell’elaborazione in lato server. La modularità rende più facile la creazione e la manutenzione del programma, inoltre ci garantisce un riutilizzo del codice in altri progetti ed applicazioni. Questo metodo però, aumenta leggermente i tempi di risposta che restano comunque accettabili.

### Prestazioni contro costi

Considerando il sistema che stiamo realizzando, possiamo dire che il non eccessivo budget a nostra disposizione ci ha consentito di realizzare il prodotto utilizzando materiale open source partendo da zero minimizzando così i costi e rendendo il sistema più che soddisfacente.

### Interfaccia contro Easy-Use

L’interfaccia, grazie all’utilizzo delle form e di una impostazione semplice e intuitiva, permette un uso facile (Easy-Use) della gestione del sistema di database prodotti anche considerevolmente grande così da rendere immediata l’attività anche ai meno esperti col computer.

***1.2 Interface Documentation Guidelines***

Gli sviluppatori dovranno seguire alcune linee guida per la scrittura del codice.

### File Java

Ogni file Java deve contenere una sola classe o interfaccia pubblica. I file saranno predisposti secondo questo ordine:

* Dichiarazione del package
* Sezione import
* Dichiarazione di interfaccia o classe:
* Attributi pubblici
* Attributi privati
* Costruttori
* Altri metodi
* Classi interne

### Naming

L’utilizzo di convenzioni sui nomi rendono il programma più leggibile e comprensibile da tutti i membri del team. è auspicabile che tutti siano in grado di intervenire su una qualsiasi linea di codice.

**Classi e interfacce**

I nomi delle classi sono nomi la cui iniziale è in maiuscolo. Ogni parola che compone un nome ha l’iniziale in maiuscolo.

I nomi delle classi devono essere semplici e intuitivi. Evitare l’uso di acronimi e abbreviazioni per i nomi delle classi.

E’ consigliato l’uso della lingua italiana per i nomi, fatta eccezione per nomi inglesi di uso comune (esempio: TestClass, …).

**Metodi**

I metodi devono essere verbi con iniziale minuscola.

**Costanti**

I nomi di constanti vengono indicati da nomi con tutte le parole in maiuscolo. Le parole vengono separate da underscore “\_”.

Ad esempio:

staticfinalint MIN\_LENGHT = 16;

### Altre regole di stile

E’ importante che vengano considerate anche ulteriori “regole di stile”, al fine di produrre un codice chiaro e leggibile.

Tra queste “regole di stile” elenchiamo le seguenti:

* I nomi di package, classi e metodi devono essere nomi descrittivi e di uso comune
* Evitare l’utilizzo di abbreviazioni di parole
* Omogeneità dei nomi all’interno dell’applicazione
* Ottimizzazioni del codice non devono comunque inficiare la leggibilità dello stesso. Se si è costretti a sviluppare codice poco leggibile, perché le estreme prestazioni sono indispensabili è necessario documentarlo adeguatamente.
* È consigliato l’utilizzo di nomi in italiano. Tuttavia è consigliato l’utilizzo di termini inglesi laddove si tratta di uso comune.
* E’ fondamentale l’utilizzo di un dizionario dei nomi unico per tutto il progetto, che tutti i programmatori saranno tenuti a seguire.
* È consigliato l’utilizzo di nomi inglesi anche nel caso si adoperino termini della libreria standard di Java (ie: OptimizedList anziché ListaOttimizzata)
* Si consiglia l’utilizzo di parti standard dei nomi in casi come:
* Classi astratte, suffisso Abstract- (ie: AbstractProdotto)
* Accezioni terminanti per Exception (ie: UtenteNonTrovatoException)
* Altre situazioni analoghe
* I nomi delle interfacce seguono le regole standard dei nomi. E’ sconsigliato usare il prefisso o suffisso “Interface”
* Dichiarare le variabili ad inizio blocco (sia un metodo o una classe), in modo da raccogliere in un unico punto tutte le dichiarazioni.
* L’inizializzazione delle variabili deve essere eseguita in fase di dichiarazione, impostando un valore di default o il risultato di un metodo.
* Allineare la dichiarazione delle variabili per renderle più leggibili, strutturandole in blocchi omogenei per contesto.
* Nel caso di algoritmi troppo complessi, eseguire un refactoring per separarlo in diversi sotto-metodi più semplici.
* I cicli devono seguire le seguenti regole:
* Per le variabili, utilizzare l’area di visibilità più stretta possibile, dichiarando le variabili appena prima del loro utilizzo.
* Per le chiamate a metodo non utilizzare spazi dopo il nome del metodo.
  1. **DEFINIZIONI, ACRONIMI E ABBREVIAZIONI**

RAD: Requirements Analysis Document

SDD: System Design Document

ODD: Object Design Document

DB: DataBase

DBMS: DataBase Management System

WebBrowser: Client/Pizzeria (utente che accede al sistema)

WebServer: Server gestore dei DataBase

# 2.Packages

# Come possiamo notare dal documento SDD-SofaPizza le componenti base che costituiscono il sistema sono raccolte in moduli a loro volta raccolti in livelli. I tre livelli rappresentano la suddivisione dettata dal modello di architettura preso in considerazione per il sistema SofaPizza (Model View Controller). Ciascun livello rappresenta un package contenente le componenti relative alle funzioni associate al livello.

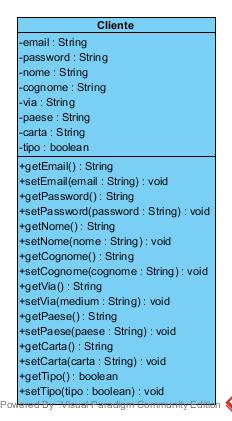
* PACKAGE SOURCE
  + PRESENTAZIONE
    - Home
    - Login
    - Registrazione
    - Ordine
    - Prenotazione
    - Risultati Ricerca
  + GESTORE
    - Esempio Ricerca
    - AutoCompletamento
    - Checkpassword
    - Decremento
    - getComuni
    - Carrello
  + STORAGE
    - DataAccess

# 3. Interfacce delle classi

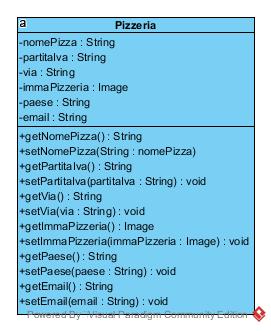
Si procede all’analisi dettagliata delle piccole classi implementate nel sistema.

L’analisi serve ad evidenziare le interfacce di interazione utilizzate nella progettazione del software.

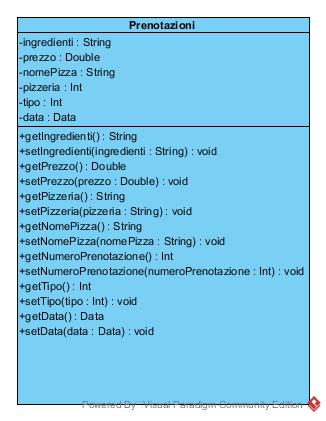
# 3. 2 Descrizione delle classi



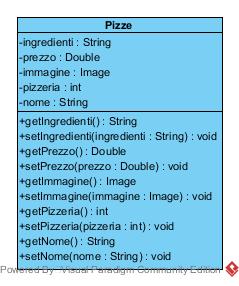
* Private email as string : e-mail del cliente che funge da identificatore
* Private password as string : password del cliente
* Private nome as string : Nome del cliente
* Private cognome as string : cognome del cliente
* Private via as string : via del cliente
* Private paese as string : paese del cliente
* Private carta as string : carta del cliente



* Private nomePizzeria as string : nome della pizzeria
* Private partitaIva as string : partita iva della pizzeria che funge da identificatore
* Private via as string : via della pizzeria
* Private immaPizzeria as image : immagine della pizzeria
* Private paese as string : paese della pizzeria
* Private e-mail as string : e-mail della pizzeria



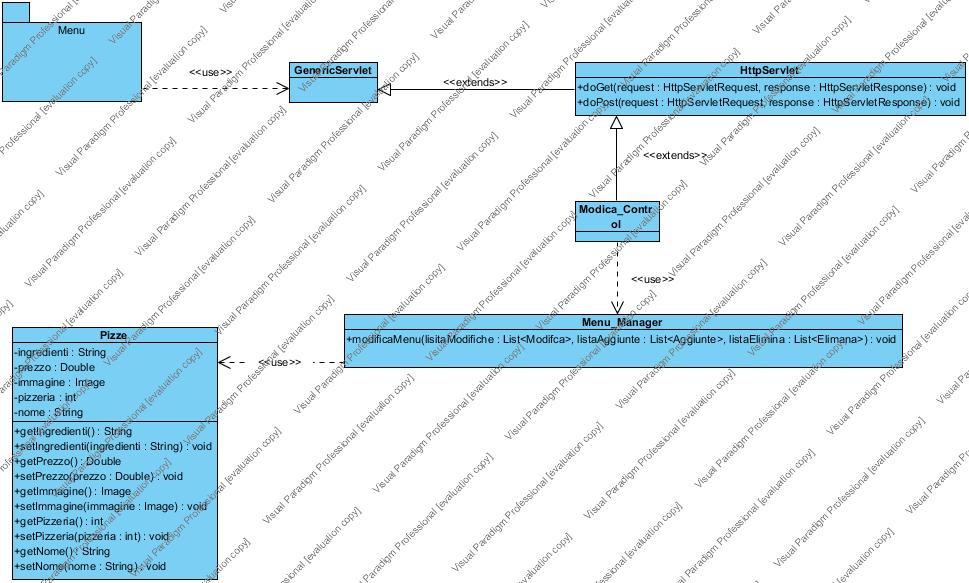
* Private ingredienti as string : ingredienti di una prenotazione
* Private prezzo as double : prezzo di una prenotazione
* Private nomePizza as string : nome della pizza della una prenotazione
* Private pizzeria as int : pizzeria a cui è associata la prenotazione
* Private tipo as int : tipologia di prenotazione se il suo valore è 0 è una prenotazione in corso se è 1 è una prenotazione storica
* Private data as data : data in cui la prenotazione è stata effettuata



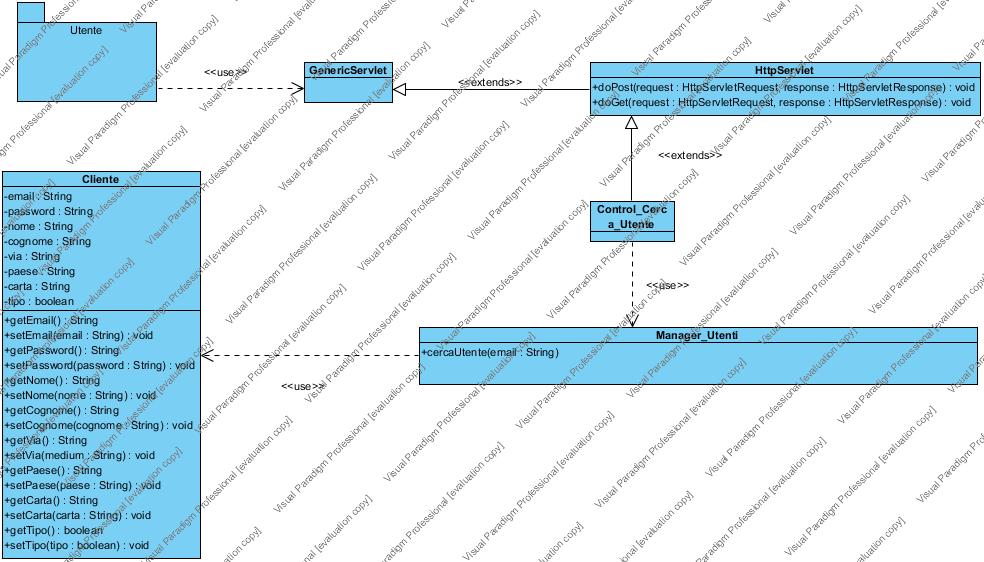
* Private ingredienti as string : ingredienti di una pizza
* Private prezzo as double : prezzo di una pizza
* Private pizzeria as int : pizzeria a cui è associata la prenotazione
* Private nome as String : nome di una pizza

# 3.1 Class diagrams

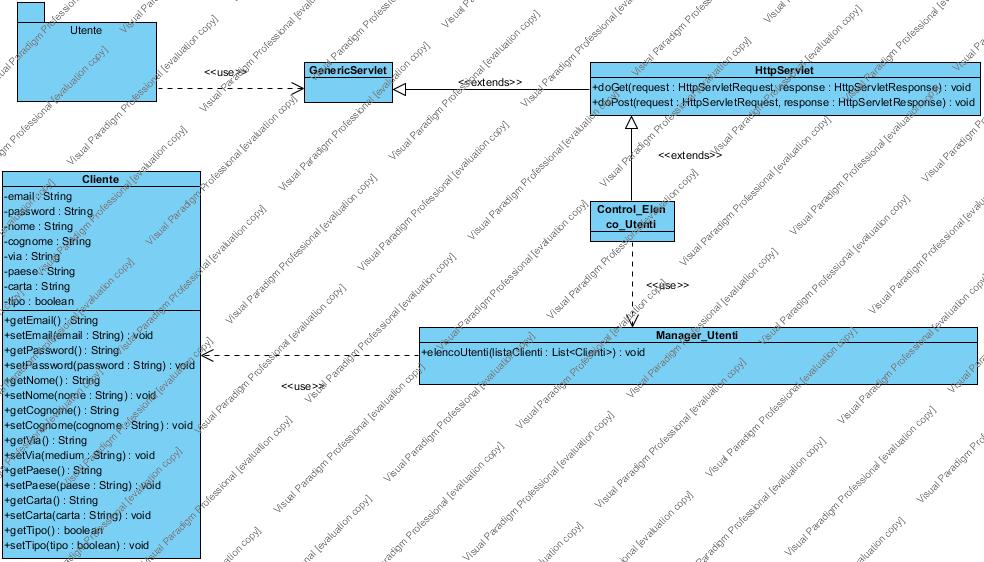
**Caricamento Menu**



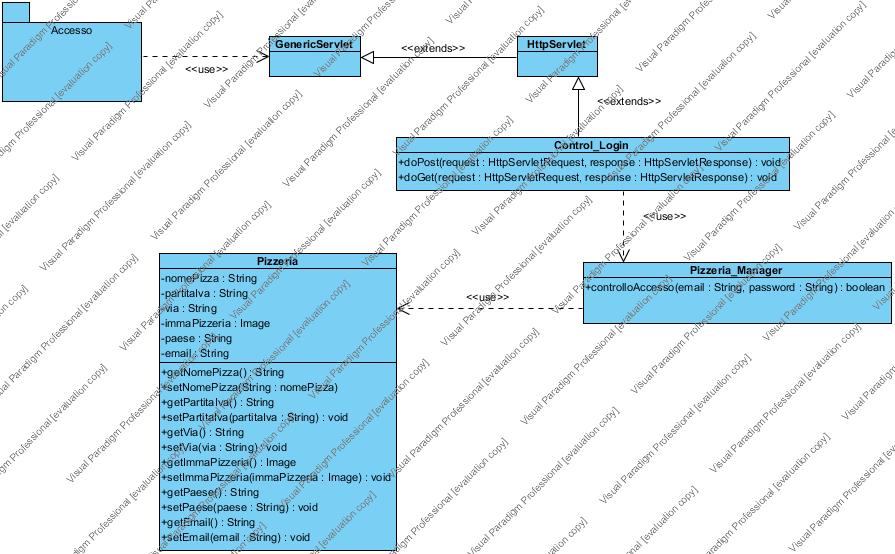
**Cerca Utente**

****

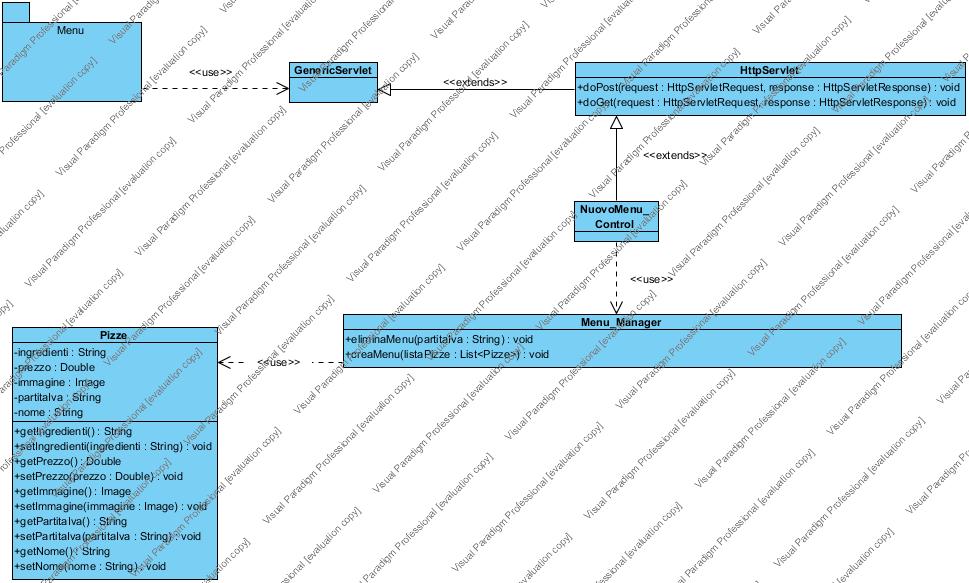
**Elenco Utente**

****

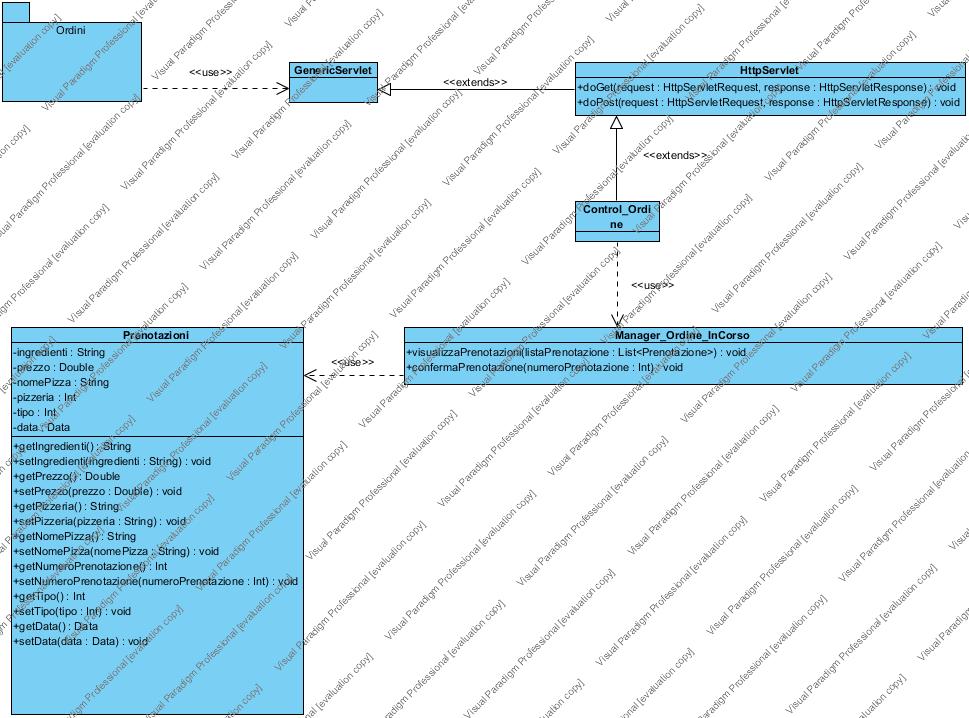
**Login Pizzeria**

****

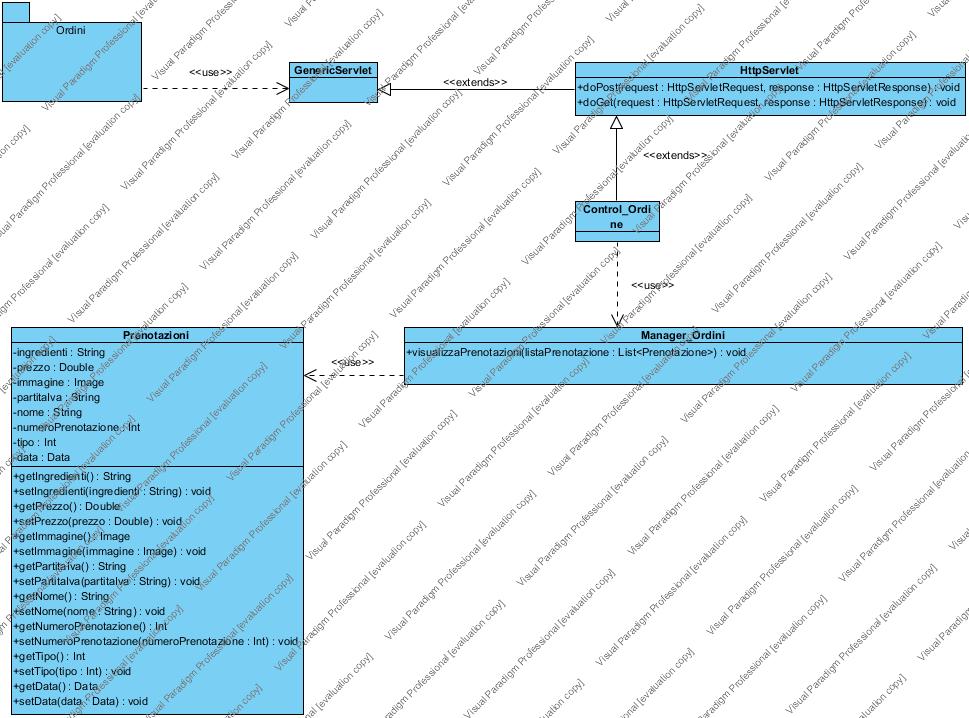
**Nuovo Menu**

****

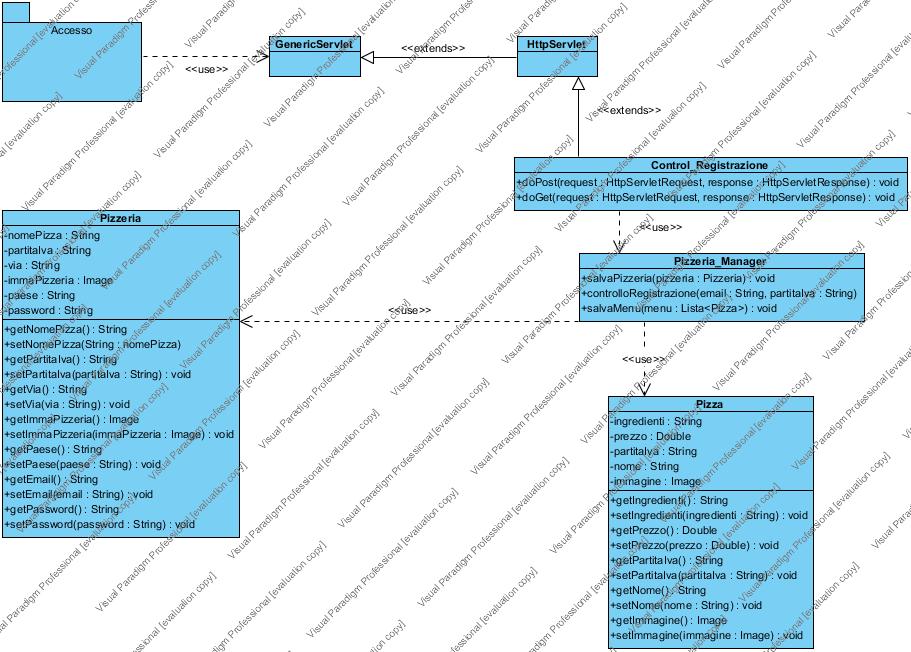
**Ordini in Corso**

****

**Storico Ordini**

****

**Registrazione**

****

**3.2 Descrizione delle classi**