|  |
| --- |
| **Università degli Studi di Salerno Corso di Ingegneria del Software** |

**Edil CommerceDesign  
Object Design Document  
Versione 1.0**

****

Data: 18/01/2023

Partecipanti:

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| De Chiara Luigi | 0512109483 |
| Ferrare Alex | 0512106300 |

Sommario

[**1.Introduzione** 3](#_Toc125797438)

[**2.Object Design Trade-Offs** 3](#_Toc125797439)

[**3. Packeges** 3](#_Toc125797440)

[**3.1. Organizzazione Packeges** 4](#_Toc125797441)

[**3.2. Back-end Packeges** 5](#_Toc125797442)

[**3.3. Front-end Packeges** 7](#_Toc125797443)

[**4. Class Interface** 7](#_Toc125797444)

[**5. Design Pattern** 11](#_Toc125797445)

# **1.Introduzione**

L’object design document è stato realizzato per aggiungere dettagli all’analisi dei requisiti e per prendere decisioni implementative.

In particolare, nel documento saranno definite le interfacce delle classi, le operazioni, i tipi, gli argomenti e le signatures dei sottosistemi definiti nel System Design Document.

Verrà effettuata l’integrazione di tutte le funzionalità individuate nelle fasi precedenti attraverso la produzione di un modello.

# **2.Object Design Trade-Offs**

**Tempi di risposta Vs. Usability**

L’interfaccia grafica è stata realizzata utilizzando elementi essenziali e intuitivi con lo scopo di garantire l’utilizzo del sistema anche ad utenti meno esperti, assicurando così i tempi di risposta stabiliti.

**Performance Vs. Sicurezza**

I tempi di risposta stabiliti vengono garantiti anche se è stato implementato un livello di sicurezza sulle registrazioni e sulle transizioni.

**Performance Vs. Costo**

Per garantire che il sistema rispettasse i requisiti di performance ovvero garantisse risposta entro 5s, si è impattato sui costi di sviluppo.

**Information Hiding Vs. Efficienza**

Tutti gli attributi delle classi del sistema saranno prevalentemente *private* per garantire una maggiore sicurezza e gestione, pertanto potrà essere necessario più tempo per accedervi.

# **3. Packeges**

La struttura del sistema si presenta divisa in due Packeges. La suddivisione viene effettuata per garantire un alto livello di sicurezza alla logica di business, poiché il Back-end è caricato lato Server l’utente visualizzerà solamente il Front-end.

**Back-end**

* Control
* Model
* Utils

**Front-end**

* Admin
* CSS
* Script
* User
* WebContent

## **3.1. Organizzazione Packeges**

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente

Immagine che contiene testo

Descrizione generata automaticamente

## **3.2. Back-end Packeges**

|  |  |
| --- | --- |
| **Control** | |
| **Classe** | **Descrizione** |
| AggiornaQ | Servlet che gestisce la variazione di quantità di un articolo nel carrello. |
| AggiornaSerach | Servlet che gestisce l’ordinamento degli articoli in base a vari criteri. |
| Aggiungi | Servlet che gestisce l’aggiunta di un articolo al carrello. |
| AggiungiArticolo | Servlet che gestisce l’aggiunta di un articolo al catalogo. |
| Autocomplete | Servlet che gestisce l’autocompletamento del nome degli articoli nella SearchBar. |
| ComputaOrdine | Servlet che gestisce l’ordine di un utente |
| Login | Servlet che gestisce l’accesso alla piattaforma |
| Logout | Servlet che gestisce il logout |
| Modifica | Servlet che gestisce la modifica dei dati di un utente. |
| ModificaArticolo | Servlet che gestisce la modifica di un articolo al catalogo. |
| OperaRecensione | Servlet che gestisce le recensioni degli articoli. |
| Registrazione | Servlet che gestisce la registrazione alla piattaforma |
| Rimuovi | Servlet che gestisce la rimozione di un articolo dal carrello. |
| Search | Servlet che gestisce la ricerca degli articoli del catalogo. |

|  |  |
| --- | --- |
| **Model** | |
| **Classe** | **Descrizione** |
| ArticoloBean | Classe che descrive le caratteristiche di un Articolo. |
| ArticoloModelDS | Descrive l’interazione col database per Articolo. |
| Carrello | Classe utilizzata per la gestione del carrello. |
| CartaBean | Classe che descrive le caratteristiche di una Carta di credito. |
| CartaModelDS | Descrive l’interazione col database per la carta di credito. |
| CategoriaBean | Classe che descrive le caratteristiche di una categoria. |
| CategoriaModelDS | Descrive l’interazione col database per le categoria. |
| ComponeBean | Classe che descrive le caratteristiche di compone ordine. |
| ComponeModelDS | Descrive l’interazione col database per definire la tipologia di prodotti che compongono un ordine. |
| ContrassegnoBean | Classe che descrive le caratteristiche di contrassegno. |
| ContrassegnoModelDS | Descrive l’interazione col database per il contrassegno. |
| InfoFatturazioneBean | Classe che descrive le informazioni di fatturazione. |
| InfoFatturazioneModelDS | Descrive l’interazione col database per le informazioni di fatturazione. |
| OrdineBean | Classe che descrive le caratteristiche di un ordine. |
| OrdineModelDS | Descrive l’interazione col database per l’ordine. |
| RecensisceBean | Classe che descrive le caratteristiche di una recensione. |
| RecensiceModelDS | Descrive l’interazione col database per la recensione. |
| RuoloUserBean | Classe che descrive le caratteristiche dei ruoli per gli user. |
| RuoloUserModelDS | Descrive l’interazione col database per Ruolo User. |
| UserBean | Classe che descrive le caratteristiche di un utente. |
| UserModelDS | Descrive l’interazione col database per User. |

|  |  |
| --- | --- |
| **Utils** | |
| **Classe** | **Descrizione** |
| MainContext | File di configurazione dell’accesso al database tramite i driver JDBC. |
| MainSession | Classe che gestisce la sessione. |
| ValidazioneInput | Classe che gestisce gli input esterni. |

## **3.3. Front-end Packeges**

|  |  |
| --- | --- |
| **Admin** | |
| **Classe** | **Descrizione** |
| Admin | Pagina per la manutenzione del catalogo. |

|  |  |
| --- | --- |
| **User** | |
| **Classe** | **Descrizione** |
| Carrello | Pagina dedicata al carrello. |
| Chechout | Pagina dedicata alla conferma dell’ordine |
| ordineEffettuato | Pagina dedicata all’ordine effettuato. |
| Profilo | Pagina dedicata alle informazione dell’utente. |

|  |  |
| --- | --- |
| **WebContent** | |
| **Classe** | **Descrizione** |
| Articolo | Pagina dedicata alla visualizzazione dell’articolo. |
| Home | Pagina dedicata all’homepage. |
| Login | Pagina dedicata all’autenticazione degli utenti. |
| Registrazione | Pagina dedicata alla registrazione degli utenti. |
| Result | Pagina dedicata alla visualizzazione del catalogo. |

# **4. Class Interface**

(private =”- “, protected=” #”, public=” +”)

|  |  |
| --- | --- |
| **Nome classe** | ArticoloBean |
| **Descrizione** | Classe che descrive le caratteristiche di un Articolo. |
| **Signature dei metodi** | * getCodiceArticolo (): String * setCodiceArticolo (String): void * getNome (): String * setNome (String): void * getImmagine (): String * setImmagine(String): void * getDescrizione(): String * setDescrizione(String): void * getCosto(): double * setCosto(double): void * getNomeCategoria(): String * setNomeCategoria(String): void * getMediaRecensioni(): int * setMediaRecensione(int): void * getGiacenza(): int * setGiacenza(int): void * isEmpty(): boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | ArticoloModelDS |
| **Descrizione** | Descrive l’interazione col database per Articolo. |
| **Signature dei metodi** | * doRetriveByKey(String): ArticoloBean * doRetriveByAll(String): Collection<ArticoloBean> * doSave(ArticoloBean): void * doUpdate(ArticoloBean, String): void * doUpdateGiacenza(int, String): void * doRetriveByCategory(String, String, String): Collection<ArticoloBean> |
| **Pre-condizione** | **Context** ArticoloModelDS:: doSave(ArticoloBean)  **Pre:**  ValidazioneInput.ValidazioneAggiungiArticolo(ArticoloBean)!=false  and not(database.articolo->  exists(a|a.codiceArticolo=ArticoloBean.codiceArticolo).  **Context** ArticoloModelDS:: doUpdate(ArticoloBean, String)  **Pre:** database.articolo ->exists(a|a.codiceArticolo=String)  **Context** ArticoloModelDS:: doUpdateGiacenza(int, String)  **Pre:** database.articolo ->exists(a|a.codiceArticolo=String)&&int>=0 |
| **Post-condizione** | **Context** ArticoloModelDS:: doRetriveByKey(String)  **Post:** return ArticoloBean OR null se non esiste.  **Context** ArticoloModelDS:: doRetriveByAll(String)  **Post:** return Collection<ArticoloBean> OR null se non esiste.  **Context** ArticoloModelDS:: doSave(ArticoloBean):  **Post:** database.articolo->exists(a| ArticoloBean.codiceArticolo=a.codiceArticolo && ArticoloBean.nome=a.nome && ArticoloBean.immagine=a.immagine && ArticoloBean.descrizione=a.descrizione && ArticoloBean.costo=a.costo && ArticoloBean.nomeCategoria=a.nomeCategoria && ArticoloBean.mediRecensioni=a.mediaRecensioni &&  ArticoloBean.giacenza=a.giacenza )  **Context** ArticoloModelDS:: doUpdate(ArticoloBean, String)  **Post:** database.articolo->exists(a| ArticoloBean.codiceArticolo=a.codiceArticolo && ArticoloBean.nome=a.nome && ArticoloBean.immagine=a.immagine && ArticoloBean.descrizione=a.descrizione && ArticoloBean.costo=a.costo && ArticoloBean.nomeCategoria=a.nomeCategoria && ArticoloBean.mediRecensioni=a.mediaRecensioni &&  ArticoloBean.giacenza=a.giacenza )  **Context** ArticoloModelDS:: doUpdateGiacenza(int, String)  **Post:** database.articolo->exists(a| ArticoloBean.giacenza=a.giacenza)  **Context** ArticoloModelDS:: doRetriveByCategory(String, String, String)  **Post:** return Collection<ArticoloBean> di una determinate categoria OR null se è vuota. |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | Carrello |
| **Descrizione** | Classe utilizzata per la gestione del carrello. |
| **Signature dei metodi** | * getItems(): List<T> * getQuantità(): List<integer> * addItem(T, int): void * deleteItem(T): int * deleteItems(): void |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | CartaBean |
| **Descrizione** | Classe che descrive le caratteristiche di una Carta di credito. |
| **Signature dei metodi** | * getNumeroOrdine(): int * setNumeroOrdine(int): void * getNumero(): String * setNumero(String): void * getIntestatario(): String * setIntestatario(String): void * getDataScadenza():String * setDataScadenza(String): void * getCvv(): string * setCvv(String): void * isEmpty(): boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | CartaModelDS |
| **Descrizione** | Descrive l’interazione col database per la carta di credito. |
| **Signature dei metodi** | * doRetriveByKey(int): CartaBean * doSave(CartaBean): void |
| **Pre-condizione** | **Contex** CartaModelDS:: doSave(CartaBean)  **Pre:**ValidazioneInput.ValidazioneCarta(CartaBean)!=false |
| **Post-condizione** | **Contex** CartaModelDS:: doRetriveByKey(int)  **Post:** return CartaBean OR null se non esiste.  **Contex** CartaModelDS:: doSave(CartaBean)  **Post:** database.carta->exists(c|  CartaBean.numeroOrdine=c.numeroOrdine&&  CartaBean.numero=c.numero&&  CartaBean.intestatario=c.intestatario&&  CartaBean.dataScadenza=c.dataScadenza&&  CartaBean.cvv=c.cvv) |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | CategoriaBean |
| **Descrizione** | Classe che descrive le caratteristiche di una categoria. |
| **Signature dei metodi** | * getImmagine(): String * setImmagine(String): void * getNome(): String * setNome(String): void * getDescrizione(): String * setDescrizione(String): void |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | CategoriaModelDS |
| **Descrizione** | Descrive l’interazione col database per le categoria. |
| **Signature dei metodi** | * doRetriveByKey(String): CategoriaBean * doRetriveByAll(String): Collection   <CategoriaBean > |
| **Pre-condizione** |  |
| **Post-condizione** | **Contex** CategoriaModelDS:: doRetriveByKey(String)  **Post:** return una categoria OR null se non esiste  **Contex** CategoriaModelDS:: doRetriveByAll(String)  **Post:** return collection<CategoriaBean> OR null se è vuota. |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | ComponeBean |
| **Descrizione** | Classe che descrive le caratteristiche di compone ordine. |
| **Signature dei metodi** | * getNumeroOrdine(): int * setNumeroOrdine(int): void * getCodiceArticolo(): String * getQuantità(): int * setQuantità(int): void |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | ComponeModelDS |
| **Descrizione** | Descrive l’interazione col database per definire la tipologia di prodotti che compongono un ordine. |
| **Signature dei metodi** | * doRetriveByOneKey(String): Collection<ComponeBean > * doSave(ComponeBean ): void |
| **Pre-condizione** |  |
| **Post-condizione** | **Contex** ComponeModelDS:: doRetriveByOneKey(String)  **Post:** return una collection di prodotti di un ordine OR null se è vuota.  **Contex** ComponeModelDS:: doSave(ComponeBean)  **Post:** database.compone->exists(c|  ComponeBean.numeroOrdine =c.numeroOrdine&&  ComponeBean.codiceArticolo =c.codiceArticolo&&  ComponeBean.quantita =c.quantita) |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | ContrassegnoBean |
| **Descrizione** | Classe che descrive le caratteristiche di contrassegno. |
| **Signature dei metodi** | * getNumeroOrdine(): int * setNumeroOrdine(int): void * isEmpty(): Boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | ContrassegnoModelDS |
| **Descrizione** | Descrive l’interazione col database per il contrassegno. |
| **Signature dei metodi** | * doRetriveByKey(int): ContrassegnoBean * doRetriveAll(String): Collection   < ContrassegnoBean >   * doSave(ContrassegnoBean): void |
| **Pre-condizione** |  |
| **Post-condizione** | **Contex** ContrassegnoModelDS:: doRetriveByKey(int)  **Post:** return ContrassegnoBen OR null se non esiste.  **Contex** ContrassegnoModelDS:: doRetriveAll(String)  **Post:** return una collezione di ordini pagati in contrassegno OR null se è vuota.  **Contex** ContrassegnoModelDS:: doSave(ContrassegnoBean)  **Post:** database.contrassegno->exists(c|  ContrassegnoBean.numeroOrdine =c.numeroOrdine) |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | InfoFatturazioneBean |
| **Descrizione** | Classe che descrive le informazioni di fatturazione. |
| **Signature dei metodi** | * getNumeroOrdine(): int * setNumeroOrdine(int): void * getNome(): String * setNome(String): void * getCognome(): String * setCognome(String): void * getEmail(): String * setEmail(String): void * getTelefono(): String * setTelefono(String): void * getIndirizzo(): String * setIndirizzo(String): void * getCittà(): String * setCittà(String): void * getStato():String * setStato(String): void * getCap(): String * setCap(String): void * isEmpty(): boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | InfoFatturazioneModelDS |
| **Descrizione** | Descrive l’interazione col database per le informazioni di fatturazione. |
| **Signature dei metodi** | * doRetriveByKey(int): InfoFatturazione * doSave(InfoFatturazione): void |
| **Pre-condizione** | **Contex** InfoFatturazioneModelDS:: doSave(InfoFatturazione)  **Pre:** ValidazioneInput.InformazioniSpedizioni (InfoFatturazione)!=false |
| **Post-condizione** | **Contex** InfoFatturazioneModelDS:: doRetriveByKey(int)  **Post:** return InfoFatturazione OR null se non esiste.  **Contex** InfoFatturazioneModelDS:: doSave(InfoFatturazione)  **Post:** database.infoFatturazione->exists(i|  infoFatturazione.nome =i.nome&&  infoFatturazione.cognome =i.cognome&&  infoFatturazione.email =i.email&&  infoFatturazione.telefono =i.telefono&&  infoFatturazione.indirizzo =i.indirizzo&&  infoFatturazione.citta =i.citta&&  infoFatturazione.stato =i.stato&&  infoFatturazione.cap =i.cap) |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | OrdineBean |
| **Descrizione** | Classe che descrive le caratteristiche di un ordine. |
| **Signature dei metodi** | + getNumeroOrdine():int  +setNumeroOrdine(int):void  + getData():Date  + setData(Date):void  + getUsername():String  + setUsername(String):void  + getImporto():double  + setImporto(double):void |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | OrdineModelDS |
| **Descrizione** | Descrive l’interazione col database per l’ordine. |
| **Signature dei metodi** | + doRetriveByKey(int): OrdineBean  + doRetribeByAll(String): Collection<OrdineBean>  + doSave(OrdineBean): void  + doUpdateImporto(OrdineBean): void  + doRetriveByUser(String): Collection<OrdineBean> |
| **Pre-condizione** | **Context** OrdineModelDS:: doSave(OrdineBean)  Pre: ValidazioneInput.InformazioniSpedizione(String)!=false  && (ValidazioneInput.ValidazioneCarta(CartaBean)!=false | | scelta.contrassegno=true) |
| **Post-condizione** | **Context** OrdineModelDS:: doRetriveByKey(int)  **Post**: return OrdineBean OR null se non esiste.  **Context** OrdineModelDS:: doRetribeByAll(String): Collection<OrdineBean>  **Post**: retun Collection<OrdineBean> OR null se vuota  **Context** OrdineModelDS:: doSave(OrdineBean): void  **Post**: database.ordine->exists(o|  ordine.numeroOrdine =o.numeroOrdine&&  ordine.data =o.data&&  ordine.username =o.username&&  ordine.importo =o.importo)  **Context** OrdineModelDS:: doUpdateImporto(OrdineBean): void  **Post**: database.ordine->  exists(o|ordine.importo=o.importo)  **Context** OrdineModelDS:: doRetriveByUser(String): Collection<OrdineBean>  **Post**: return Collection<OrdineBean> OR null se vuota. |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | RecensisceBean |
| **Descrizione** | Classe che descrive le caratteristiche di una recensione. |
| **Signature dei metodi** | + getCodiceArticolo(): String  + setCodiceArticolo(String): void  + getUsername(): String  + setUsername(String): void  + getData():Date  + setData(Date):void  + getValore(): int  + setValore(int): void  + getTesto(): String  + set Testo(int): void  + isEmpity(): boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | RecensiceModelDS |
| **Descrizione** | Descrive l’interazione col database per la recensione. |
| **Signature dei metodi** | + doRetriveByKey(String, String): ResencisceBean  + doRetriveByOneKey(String): Collection<RecensiceBean>  + doRetriveByCodiceArticolo(String): Collection<RecensisceBean>  + doSave(RecensisceBean): void  + doUpdate(RecensisceBean): void  + doDelete(RecensisceBean): void |
| **Pre-condizione** | **Context** RecensiceModelDS:: doSave(RecensisceBean)  **Pre:** ValidazioneInput.ValidazioneInserimentoRecensione  (RecensisceBean)!=false  **Context** RecensiceModelDS:: doUpave(RecensisceBean)  **Pre:** ValidazioneInput.ValidazioneInserimentoRecensione  (RecensisceBean)!=false |
| **Post-condizione** | **Context** RecensiceModelDS:: doRetriveByKey(String, String)  **Post:** return di un RecensisceBean OR null se non esiste  **Context** RecensiceModelDS:: doRetriveByOneKey(String)  **Post:** return di una Collection<RecensisceBean> OR null se vuota.  **Context** RecensiceModelDS:: doRetriveByCodiceArticolo(String)  **Post:** return di una Collection<RecensisceBean> OR null se vuota.  **Context** RecensiceModelDS:: doSave(RecensisceBean)  **Post:** database.recensisce->exists(o|  recensisce.codiceArticolo =o.codiceArticolo&&  recensisce.username =o.username&&  recensisce.data =o.data&&  recensisce.valore =o.valore&&  recensisce.testo =o.testo)  **Context** RecensiceModelDS:: doUpdate(RecensisceBean)  **Post:** database.recensisce->exists(o|  recensisce.codiceArticolo =o.codiceArticolo&&  recensisce.username =o.username&&  recensisce.data =o.data&&  recensisce.valore =o.valore&&  recensisce.testo =o.testo)    **Context** RecensiceModelDS:: doDelete(RecensisceBean)  **Post:** not **(**database.recensisce->exists(o|  recensisce.codiceArticolo =o.codiceArticolo&&  recensisce.username =o.username&&  recensisce.data =o.data&&  recensisce.valore =o.valore&&  recensisce.testo =o.testo)) |
| **Invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | RuoloUserBean |
| **Descrizione** | Classe che descrive le caratteristiche dei ruoli per gli user. |
| **Signature dei metodi** | + getUsername(): String  + setUsername(String): void  + getNome(): String  + setNome(String): void  + isEmpity():boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | RuoloUserModelDS |
| **Descrizione** | Descrive l’interazione col database per Ruolo User. |
| **Signature dei metodi** | + doRetriveByOneKey(String) Collection<RuoloUserBean>  + doSave(RuoloUserRole): void |
| **Pre-condizione** |  |
| **Post-condizione** | **Context** RuoloUserModelDS::doRetriveByOneKey(String)  **Post:** return una Collection<RuoloUserBean> OR null se vuota  **Context** RuoloUserModelDS:: doSave(RuoloUserRole)  **Post:** database.ruoloUser->exists(r|  ruoloUser.username =o.username&&  ruoloUser.nome =o.nome ) |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | UserBean |
| **Descrizione** | Classe che descrive le caratteristiche di un utente. |
| **Signature dei metodi** | + getCittà(): String  + setCittà(String): void  + getStato(): String  + setStato(String): void  + getCap(): String  + setCap(String): void  + getUsername(): String  + setUsername(String): void  + getNome(): String  + setNome(String): void  + getCognome(): String  + setCognome(String): void  + getEmail(): String  + setEmail(String): void  + getTelefono(): String  + setTelefono(String): void  + getIndirizzo():String  + setIndirizzo(String):void  + getUserPassword(): String  + setUserPassword(String): String  + isEmpity(): Boolean |
| **Pre-condizione** |  |
| **Post-condizione** |  |
| **invariante** |  |

|  |  |
| --- | --- |
| **Nome classe** | UserModelDS |
| **Descrizione** | Descrive l’interazione col database per User. |
| **Signature dei metodi** | + doRetriveByKey(String): UserBean  + doSave(UserBean): void  + doUpdate(UserBean, String): void |
| **Pre-condizione** | **Context** UserModelDS:: doSave(UserBean)  **Pre:** database.user->exists(u|u.username=String) &&  ValidazioneInput.ValidazioneRegistrazione(UserBean)!=false  **Context** UserModelDS:: doUpdate(UserBean, String)  **Pre:** database.user->exists(u|u.username=String) |
| **Post-condizione** | **Context** UserModelDS:: doRetriveByKey(String)  **Post:** return UserBean OR null se non esiste  **Context** UserModelDS:: doSave(UserBean)  **Post:** database.user->exists(u|  user.username =u.username&&  user.nome =u.nome&&  user.cognome =u.cognome&&  user.email =u.email&&  user.telefono =u.telefono&&  user.indirizzo =u.indirizzo&&  user.citta =u.citta&&  user.stato =u.stato&&  user.cap =u.cap&&  user.userPassword=u.userPassword)  **Context** UserModelDS:: doUpdate(UserBean, String)  **Post:** database.user->exists(u|  user.username =u.username&&  user.nome =u.nome&&  user.cognome =u.cognome&&  user.email =u.email&&  user.telefono =u.telefono&&  user.indirizzo =u.indirizzo&&  user.citta =u.citta&&  user.stato =u.stato&&  user.cap =u.cap&&  user.userPassword=u.userPassword) |
| **invariante** |  |

# **5. Design Pattern**

Nel sistema che abbiamo strutturato è stato utilizzato il Pattern DAO, che ci ha permesso di separare la logica di persistenza dei dati in un livello separato. Utilizzando questo pattern il servizio rimane completamente all’oscuro di come vengono eseguite le operazioni per l’accesso al database.

Principio noto come **separazione delle logica di business.**

Ovviamente l’accesso al database verrà eseguito solamente dalle classi DAO, per garantire sia una maggiore manutenibilità sia una maggiore sicurezza.

Essendo completamente separata la logica di persistenza è stato molto più semplice realizzare Unit-Test per i singoli componenti.

