UNIVERSITÀ DEGLI STUDI DI SALERNO

CORSO DI LAUREA IN INFORMATICA

INGEGNERIA DEL SOFTWARE

ORION

OBJECT DESIGN DOCUMENT

Docente: Andrea De Lucia

Studente: Michelangelo Esposito – 0512104784

INDICE

1. INTRODUZIONE

1.1 TRADE-OFF PER L'OBJECT DESIGN

Considerando gli obiettivi di design individuati nell'SDD, l'implementazione di Orion sarà incentrata verso la minimizzazione dei tempi di risposta delle pagine, soprattutto per quanto riguarda le operazioni di ricerca effettuate sul catalogo delle inserzioni proposte.

1.2 DOCUMENTAZIONE DELLE INTERFACCE

1.2.1 CODICE JAVA

Classi ed interfacce:

- La nomenclatura delle classi dovrà rispettare la notazione UpperCamelCase;
- L'identificativo delle classi non dovrà essere ambiguo e dovrà essere congruo allo scopo della classe;
- L'identificativo delle classi dovrà essere formulato al singolare;
- I nomi delle classi di test dovranno riportare il nome della classe testata, seguita dal suffisso "Test";
- I nomi delle classi Bean dovranno riportare il nome dell'entità di riferimento, seguita dal suffisso "Bean";
- I nomi delle classi Data Access Object dovranno riportare il nome dell'entità di riferimento, seguita dal suffisso "Dao".

Metodi:

- La nomenclatura dei metodi dovrà seguire la notazione lowerCamelCase;
- Gli identificativi dei metodi dovranno iniziare con un verbo (es. doSave(), getIdUtente());
- Per ogni metodo definito in una classe sarà necessario specificare il livello di visibilità;
- Eventuali parametri nella signature di un metodo dovranno seguire le convenzioni, di seguito definite, per le variabili d'istanza.

Variabili d'istanza e variabili locali:

- I nomi delle variabili d'istanza e locali dovranno seguire la notazione lowerCamelCase;
- Per ogni variabile d'istanza definita in una classe sarà necessario specificare il livello di visibilità;

Costanti:

- I nomi delle costanti dovranno essere definiti in maiuscolo;
- La separazione tra diverse parole dovrà essere effettuata mediante il carattere "_";
- Le costanti dovranno essere dichiarate come "static final".

Blocchi e indentazioni:

- Il codice dovrà essere indentato utilizzando un carattere di tabulazione per ogni livello di indentazione;
- Le parentesi graffe per l'inizio di un nuovo blocco dovranno essere riportate sulla stessa riga della definizione del blocco;
- Le parentesi graffe che marcano la fine di un blocco dovranno essere verticalmente allineate con l'inizio della definizione del blocco.

Blocchi eccezionali:

- Ogni blocco try/catch/finally definite all'interno di un metodo dovrà essere indentato secondo le specifiche sopra riportate;
- Le clausole catch dovranno essere riportate in maniera ordinata, dalla più specifica alla più generale.

Annotazioni:

- Le annotazioni previste per una classe o per un metodo dovranno apparire una per riga, subito dopo il blocco di documentazione;
- Eventuali annotazioni prima di una variabile d'istanza dovranno essere definite una per linea, al di sopra della variabile stessa e senza lasciare linee vuote.

Documentazione:

- Ogni classe e metodo dovrà essere corredata da documentazione javadoc adeguata, in cui vengono specificati visibilità, parametri e tipo di ritorno.

Esempio di classe Java accettabile:

```
/**
* A test class
*/
public class MyClass {
        private static final String CONST_VAR = "test";

        /**
        * Main class
        * @param args command-line arguments
        */
        public static void main (String[] args) {
                if(true) {
                        System.out.println(CONST_VAR);
                }
        }
}
```

Esempio non accettabile

```
public class MyClass {
private static final String CONST_VAR = "test";

        /**
        * Main class
*/
        public static void main (String[] args) {
        if(true) {
                        System.out.println(CONST_VAR);
                }
}
```

1.2.2 PAGINE HTML5

- Ogni document dovrà riportare il tag di apertura <!doctype html>;
- Ogni tag aperto nel corpo del documento HTML, a meno dell'utilizzo di tag singoli o JSP, dovrà riportare il rispettivo tag di chiusura;
- La struttura basi di una pagina HTML, head-body, dovrà essere rispettata, a meno dell'utilizzo di JSP;
- Ogni documento HTML dovrà essere indentato utilizzando un carattere di tabulazione per ogni livello di indentazione, corrispondente alla definizione di un tag figlio ad un elemento del Document Object Model.

Esempio di pagina HTML accettabile:

```
<!doctype html>
<html>
	<head>
		<title>Titolo</title>
	</head>
	<body>
		<span>Test<span>
	</body>
</html>
```

Esempio non accettabile:

```
<html>
	<head>
		<title>Titolo</title>
	</head>
	<body>
	<span>Test<span>
</body>
</html>
```

1.2.3 JSP

- Le elaborazioni di codice JSP dovranno produrre codice HTML conforme alle convenzioni specificate;
- Tutto il codice Java all'interno di una JSP dovrà essere disponibile attraverso tag JSTL Core, Expression Language e dai seguenti tag JSP:
  - <%! %>: dichiarazione;
  - <%= %>: espressione;
  - <% %>: scriptlet;
  - <%@ %>: direttiva;
- Ogni blocco di codice Java all'interno di una JSP dovrà essere adeguatamente documentato.

Esempio di pagina HTML accettabile:

```
<%@page contentType="text/html"; charset=utf-8" pageEncoding="utf-8"%>

<!doctype html>
<html>
	<head>
		<title>Titolo</title>
	</head>
	<body>
		<p>${requestScope.testText}</p>
	</body>
</html>
```

Esempio non accettabile:

```
<%@page contentType="text/html"; charset=utf-8" pageEncoding="utf-8"%>

<html>
	<head>
		<title>Titolo</title>
	</head>
	<body>
	<p>${requestScope.testText}</p>
</body>
</html>
```

1.2.4 SCRIPT JAVASCRIPT

- Ogni funzione JavaScript dovrà essere adeguatamente documentata, in stile javadoc;
- I nomi di funzioni, variabili e costanti dovranno rispettare le stesse convenzioni definite per il codice Java.

Esempio di script JavaScript accettabile:

```
/**
* Test function
*/
function test() {
        var test = 'This is a test';
        document.getElementById("testId").innerHTML(test);
}
```

Esempio non accettabile:

```
function test() {
var test = 'This is a test';
        document.getElementById("testId").innerHTML(test);
        }
```

1.2.5 FOGLI DI STILE CSS3

- Ogni regola CSS dovrà iniziare all'inizio di una nuova linea, con la specifica dei selettori di tale regola.
- L'ultimo selettore di una regola CSS dovrà essere seguito dall'apertura del blocco;

Esempio di foglio di stile CSS3 accettabile:

```
.div p {
        background-color: #FFFF;
        display: block;
}
```

Esempio non accettabile:

```
.div p
{
background-color: #FFFF;
        display: block;
  }
```

1.3 DESIGN PATTERN UTILIZZATI

1.3.1 PATTERN MODEL VIEW CONTROLLER (MVC)

Il pattern Model-View-Controller (MVC) è un design pattern architetturale utilizzato per separare il presentation layer dall'application processing layer; è basato sulla separazione dei compiti tra componenti software che interpretano tre ruoli principali:

- Il Model fornisce i metodi per accedere ai dati persistenti;
- Il View visualizza i dati contenuti nel model e si occupa dell'interazione con l'utente;
- Il Controller riceve i comandi dall'utente e li attua modificando lo stato delle altre due componenti.

1.3.2 PATTERN DATA ACCESS OBJECT (DAO)

Il pattern Data Access Object (DAO) è un design pattern architetturale utilizzato per separare i servizi di business dell'application processing layer dalle operazioni di accesso ai dati del data management layer; un DAO implementa il meccanismo di accesso richiesto per lavorare con la sorgente dei dati (in questo caso il DBMS MySQL).

Il Data Access Object nasconde completamente i dettagli dell'interazione con la sorgente dati e l'interfaccia esposta al client non cambia quando l'implementazione dell'origine dati sottostante cambia.

Convenzioni per l'implementazione:

- Gli identificativi dei metodi per le operazioni di inserimento, aggiornamento e rimozione dovranno essere rispettivametne doSave(), doUpdate() e doDelete();
- Gli identificativi dei metodi per il recupero di dati dovranno presentare il prefisso "doRetriveBy" seguito dai nome degli attributi su cui basare il recupero, ad accezione del metodo di recupero tramite chiave primaria, definito come doRetrieveByKey(). Seguendo questa convenzione si ritiene inutile descrivere tali metodi, il cui scopo risulta facilmente intuibile.

1.4 DEFINIZIONI, ACRONIMI ED ABBREVIAZIONI

DAO: Data Access Object;
DBMS: DataBase Management System;
HTTP: HyperText Transfer Protocol;
JDBC: Java DataBase Connectivity;
JSP: Java Server Page;
SQL: Structured Query Language.

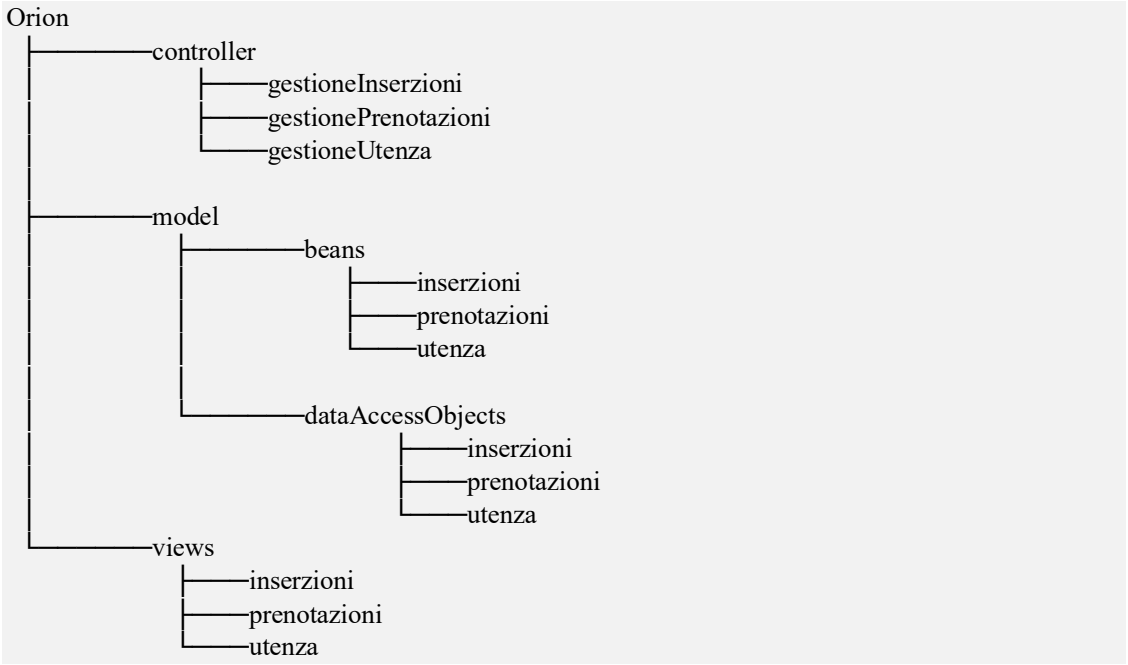
2. PACCHETTI ED INTERFACCE

In questa sezione vengono presentate le gerarchie dei package che compongono Orion ed i dettagli delle classi e delle pagine che li compongono.

Seguendo l'architettura MVC, vi sono tre package principali:

- Controller;
- Model ;
- View.

Ciascuno di questi è poi ulteriormente diviso secondo quelle che sono le tre macro-categorie dei dati della piattaforma: inserzioni, prenotazioni ed utenza.



2.1 MODEL

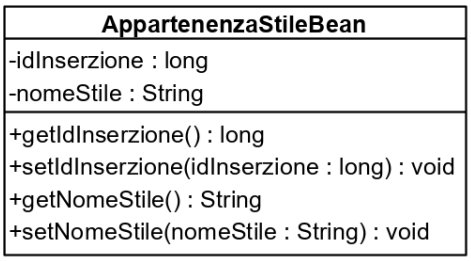
2.1.1 BEANS

Insieme di classi che rappresentano, in memoria, i dati relativi agli oggetti persistenti.

Ogni classe possiede una serie di metodi di accesso e di metodi modificatori per gestire gli attributi, non descritti qui perché elementari.

2.1.1.1 INSERZIONI

AppartenenzaStileBean: racchiude le informazioni relative all'associazione tra un'inserzione ed uno stile.



CommentoBean: racchiude le informazioni relative al commento lasciato dal proprietario sotto una recensione.

CommentoBean
-idRecensione : long -contenuto : String -dataPubblicazione : Date
+getIdRecensione() : long +setIdRecensione(idRecensione : long) : void +getContenuto() : String +setContenuto(contenuto : String) : void +getDataPubblicazione() : Date +setDataPubblicazione(dataPubblicazione : Date) : void

ImmagineBean: racchiude le informazioni relativa ad una delle immagini di un'inserzione.

ImmagineBean
-pathname : String -idInserzione : long
+getPathname() : String +setPathname(pathname : String) : void +getIdInserzione() : long +setIdInserzione(idInserzione : long) : void

IntervalloDisponibilitàBean

IntervalloDisponibilitàBean
-idInserzione : long -dataInizio : Date -dataFine : Date
+getIdInserzione() : long +setIdInserzione(idInserzione : long) : void +getDataInizio() : Date +setDataInizio(dataInizio : Date) : void +getDataFine() : Date +setDataFine(dataFine : Date) : void

InserzioneBean

InserzioneBean
-idInserzione : long -idProprietario : long -stato : String -regione : String -città : String -cap : String -strada : String -descrizione : String -numeroCivico : int -maxNumeroOspiti : int -prezzoGiornaliero : double -metratura : double -dataInserimento : Date -visibilità : boolean
+getIdInserzione() : long +setIdInserzione(idInserzione : long) : void +getIdProprietario() : long +setIdProprietario(idProprietario : long) : void +getStato() : String +setStato(stato : String) : void +getRegione() : String +setRegione(regione : String) : void +getCittà() : String +setCittà(città : String) : void +getCap() : String +setCap(cap : String) : void +getStrada() : String +setStrada(strada : String) : void +getDescrizione() : String +setDescrizione(descrizione : String) : void +getNumeroCivico() : int +setNumeroCivico(numeroCivico : int) : void +getMaxNumeroOspiti() : int +setMaxNumeroOspiti(maxNumeroOspiti : int) : void +getPrezzoGiornaliero() : double +setPrezzoGiornaliero(prezzoGiornaliero : double) : void +getMetratura() : double +setMetratura(metratura : double) : void +getDataInserimento() : Date +setDataInserimento(dataInserimento : Date) : void +getVisibilità() : boolean +setVisibilità(visibilità : boolean) : void

RecensioneBean

RecensioneBean
-idRecensione : long -idCliente : long -idInserzione : long -punteggio : int -titolo : String -contenuto : String -dataPubblicazione : Date
+getIdRecensione() : long +setIdRecensione(idRecensione : long) : void +getIdCliente() : long +setIdCliente(idCliente : long) : void +getIdInserzione() : long +setIdInserzione(idInserzione : long) : void +getPunteggio() : int +setPunteggio(punteggio : int) : void +getTitolo() : String +setTitolo(titolo : String) : void +getContenuto() : String +setContenuto(contenuto : String) : void +getDataPubblicazione() : Date +setDataPubblicazione(dataPubblicazione : Date) : void

StileBean

StileBean
-nomeStile : String -descrizione : String
+getNomeStile() : String +setNomeStile(nomeStile : String) : void +getDescrizione() : String +setDescrizione(descrizione : String) : void

2.1.1.2 PRENOTAZIONI

MetodoPagamentoBean

MetodoPagamentoBean
-numeroCarta : String -idUtente : long -nomeTitolare : String -cognomeTitolare : String -dataScadenza : Date -preferito boolean
+getNumeroCarta() : String +setNumeroCarta(numeroCarta : String) : void +getIdUtente() : long +setIdUtente(idUtente : long) : void +getNomeTitolare() : String +setNomeTitolare(nomeTitolare : String) : void +getCognomeTitolare() : String +setCognomeTitolare(cognomeTitolare : String) : void +getDataScadenza() : Date +setDataScadenza(dataScadenza : Date) : void +getPreferito boolean() +setPreferito boolean(preferito boolean) : void

PrenotazioneArchiviataBean

PrenotazioneArchiviataBean
-idPrenotazione : long -stato : String -regione : String -città : String -cap : String -indirizzo : String -prezzoGiornaliero : float
+getIdPrenotazione() : long +setIdPrenotazione(idPrenotazione : long) : void +getStato() : String +setStato(stato : String) : void +getRegione() : String +setRegione(regione : String) : void +getCittà() : String +setCittà(città : String) : void +getCap() : String +setCap(cap : String) : void +getIndirizzo() : String +setIndirizzo(indirizzo : String) : void +getPrezzoGiornaliero() : float +setPrezzoGiornaliero(prezzoGiornaliero : float) : void

PrenotazioneBean

PrenotazioneBean
-idPrenotazione : long -idCliente : long -idProprietario : long -dataPrenotazione : Date -dataCheckIn : Date -dataCheckOut : Date -totale : float -numeroOspiti : int -stato : boolean
+getIdPrenotazione() : long +setIdPrenotazione(idPrenotazione : long) : void +getIdCliente() : long +setIdCliente(idCliente : long) : void +getIdProprietario() : long +setIdProprietario(idProprietario : long) : void +getDataPrenotazione() : Date +setDataPrenotazione(dataPrenotazione : Date) : void +getDataCheckIn() : Date +setDataCheckIn(dataCheckIn : Date) : void +getDataCheckOut() : Date +setDataCheckOut(dataCheckOut : Date) : void +getTotale() : float +setTotale(totale : float) : void +getNumeroOspiti() : int +setNumeroOspiti(numeroOspiti : int) : void +getStato() : boolean +setStato(stato : boolean) : void

2.1.1.3 UTENZA

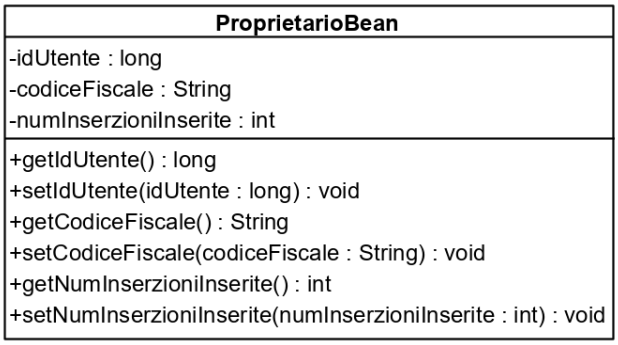
AmministratoreBean: racchiude le informazioni dell'amministratore.

AmministratoreBean
-idUtente : long -numInserzioniRevisionate : int
+getIdUtente() : long +setIdUtente(idUtente : long) : void +getNumInserzioniRevisionate() : int +setNumInserzioniRevisionate(numInserzioniRevisionate : int) : void

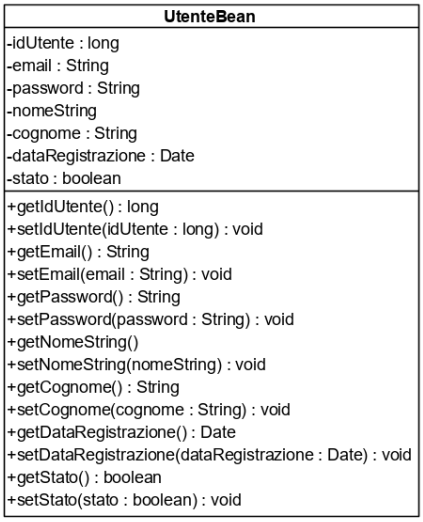
ClienteBean: racchiude le informazioni di un cliente.

ClienteBean
-idUtente : long -dataUltimaPrenotazione : Date
+getIdUtente() : long +setIdUtente(idUtente : long) : void +getDataUltimaPrenotazione() : Date +setDataUltimaPrenotazione(dataUltimaPrenotazione : Date) : void +setAttribute(attribute) : void

ProprietarioBean: racchiude le informazioni di un proprietario.



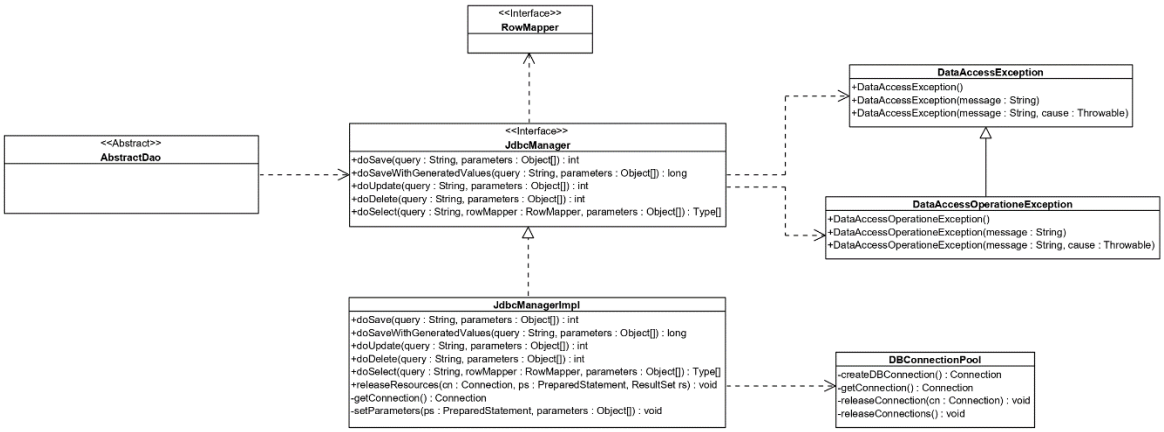
UtenteBean: racchiude le informazioni di un utente registrato.



2.1.2 *DATA ACCESS OBJECTS*

```
model.dataAccessObjects
        ├──inserzioni
        │       ├──AppartenenzaStileDao
        │       ├──AppartenenzaStileDaoImpl
        │       ├──CommentoDao
        │       ├──CommentoDaoImpl
        │       ├──ImmagineDao
        │       ├──ImmagineDaoImpl
        │       ├──InserzioneDao
        │       ├──InserzioneDaoImpl
        │       ├──IntervalloDisponibilitàDao
        │       ├──IntervalloDisponibilitàDaoImpl
        │       ├──RecensioneDao
        │       ├──RecensioneDaoImpl
        │       ├──StileDao
        │       └──StileDaoImpl
        │
        ├──prenotazioni
        │       ├──MetodoPagamentoDao
        │       ├──MetodoPagamentoDaoImpl
        │       ├──PrenotazioneArchiviataDao
        │       ├──PrenotazioneArchiviataDaoImpl
        │       ├──PrenotazioneDao
        │       └──PrenotazioneDaoImpl
        │
        ├──utenza
        │       ├──AmministratoreDao
        │       ├──AmministratoreDaoImpl
        │       ├──ClienteDao
        │       ├──ClienteDaoImpl
        │       ├──ProprietarioDao
        │       ├──ProprietarioDaoImpl
        │       ├──UtenteDao
        │       └──UtenteDaoImpl
        │
        ├──AbstractDao
        ├──DataAccessException
        ├──DataAccessOperationException
        ├──DBConnectionPool
        ├──JdbcManager
        ├──JdbcManagerImpl
        └──RowMapper
```

2.1.2.1 CLASSI DI UTILITÀ GENERALE ED ECCEZIONI



DBConnectionPool

Metodi:

- java.sql.Connection createDBConnection(): crea una nuova connessione e la aggiunge al pool di connessioni aperte;
- java.sql.Connection getConnection(): recupera e restituisce una connessione dal pool o, se questo è vuoto ne crea una con createDBConnection();
- void releaseConnection(Connection c): rilascia la connessione ricevuta come parametro, chiudendola e rimuovendola dal pool;
- void releaseConnections(): rilascia tutte le connessioni aperte, chiudendole e rimuovendole dal pool.

Contratto:

```
context DBConnectionPool::createDBConnection() pre:
  DBConnectionPool.freeDBConnections != null
context DBConnectionPool::createDBConnection() post:
  DBConnectionPool.freeDBConnections += new Connection()
  result = DBConnectionPool.freeDBConnections.get(size)

context DBConnectionPool::getConnection() pre:
  DBConnectionPool.freeDBConnections != null
context DBConnectionPool::getConnection() post:
  if DBConnectionPool.freeDBConnections.size() > 0:
            result = DBConnectionPool.freeDBConnections(0)
  else:
            result = createDBConnection()

context DBConnectionPool::releaseConnection(cn:Connection) pre:
  cn != null and
  DBConnectionPool.freeDBConnections != null
context DBConnectionPool::getConnection() post:
  DBConnectionPool.freeDBConnections.remove(cn)
  cn.close()
context DBConnectionPool::releaseConnections() pre:
  DBConnectionPool.freeDBConnections != null
context DBConnectionPool::getConnection() post:
  for connection cn in DBConnectionPool.freeDBConnections:
            DBConnectionPool.freeDBConnections.remove(cn)
            cn.close()
```

JdbcManager

Metodi:

- int doSave(String query, Object[] parameters) esegue la query parametrica di salvataggio specificata, mediante i parametri ricevuti;
- long doSaveWithGeneratedKeys(String query, Object[] parameters) esegue la query parametrica di salvataggio specificata, mediante i parametri ricevuti e restituisce la chiave primaria auto generata;
- int doUpdate(String query, Object[] parameters) esegue la query parametrica di aggiornamento specificata, mediante i parametri ricevuti;
- int doDelete(String query, Object[] parameters) esegue la query parametrica di rimozione specificata, mediante i parametri ricevuti;
- java.util.List<E> doSelect(String query, RowMapper rowMapper, Object[] parameters);

JdbcManagerImpl

Implementazione di JdbcManager;

Metodi aggiuntivi:

- void releaseResources(Connection cn, PreparedStatement ps, ResultSet rs): rilascia le risorse utilizzate da JDBC per una query;
- java.sql.Connection getConnection(): recupera una connessione aperta dal connection pool;
- void setParameters(PreparedStatement ps, Object[] parameters): si occupa del settaggio dei parametri del tipo appropriato per una query parametrica.

Contratto:

```
context JdbcManagerImpl::doSave(query:String, parameters: Object[]) pre:
  query != null and
  for p in params:
            p != null
  and doSelect(parameters) = null
context JdbcManagerImpl::doSave(query:String, parameters: Object[]) post:
  result = 1 and
  doSelect(parameters) != null

context JdbcManagerImpl::doSaveWithGeneratedKey(query:String, parameters: Object[]) pre:
  query != null and
  for p in params:
            p != null
  and doSelect(parameters) = null
context JdbcManagerImpl::doSaveWithGeneratedKey(query:String, parameters: Object[]) post:
  result = AUTO_GENERATED_KEY and
  doSelect(parameters) != null

context JdbcManagerImpl::doUpdate(query:String, parameters: Object[]) pre:
  query != null and
  for p in params:
            p != null
  and doSelect(parameters) != null
context JdbcManagerImpl::doUpdate(query:String, parameters: Object[]) post:
  result = 1

context JdbcManagerImpl::doDelete(query:String, parameters: Object[]) pre:
```

```
  query != null and
  for p in params:
            p != null
  and doSelect(parameters) != null
context JdbcManagerImpl::doDelete(query:String, parameters: Object[]) post:
  result = 1 and
  doSelect(parameters) = null

context JdbcManagerImpl::doSelect(query:String, rowMapper:RowMapper, parameters: Object[]) pre:
  query != null and
  for p in params:
            p != null
  and rowMapper != null and
context JdbcManagerImpl::doSelect(query:String, rowMapper:RowMapper, parameters: Object[]) post:
  result = ogni entry di tipo specificato nel database che soddisfi parameters


context JdbcManagerImpl::releaseResources(cn:Connection, ps:PreparedStatement, rs:ResultSet) pre:
  cn != null and
  ps != null and
  rs != null
context JdbcManagerImpl::releaseResources(cn:Connection, ps:PreparedStatement, rs:ResultSet) post:
  risorse chiuse correttamente

context JdbcManagerImpl::setParameters(ps:PreparedStatement, parameters:Object[]) pre:
  ps != null and
  for p in params:
            p != null
context JdbcManagerImpl::setParameters(ps:PreparedStatement, parameters:Object[]) post:
```

RowMapper

Interfaccia utilizzata per specificare le modalità di recupero dei dati prelevati da una query JDBC.

Metodi:

- void map(ResultSet rs): metodo che specifica come recuperare i parametri dall'oggetto ResultSet ricevuto.

DataAccessException

Eccezione lanciata a seguito errori generici riscontrati nella gestione delle risorse JDBC.

DataAccessOperationException

Eccezione lanciata a seguito errori riscontrati nell'accesso al database durante una query JDBC.

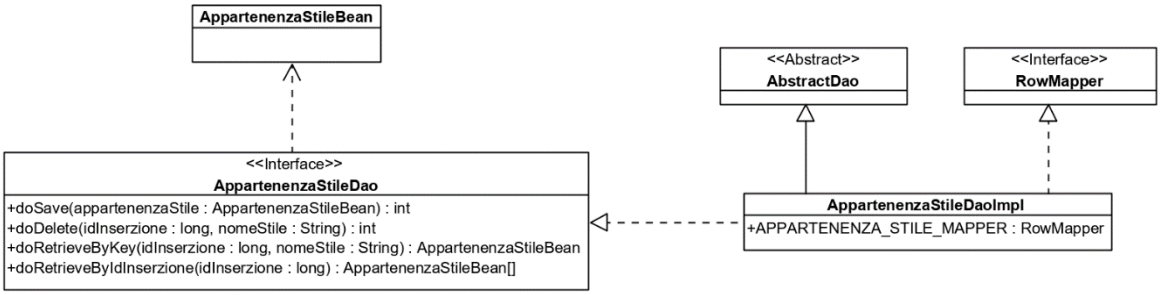
2.1.2.2 INSERZIONI

AppartenenzaStileDao

Metodi:

- int doSave(AppartenenzaStileBean appartenenzaStile);
- int doDelete(long idInserzione, String nomeStile);
- AppartenenzaStileBean doRetrieveByKey(long idInserzione, String nomeStile);
- java.util.List<AppartenenzaStileBean> doRetrieveByIdInserzione(long idInserzione).

Diagramma di classe:



Contratto:

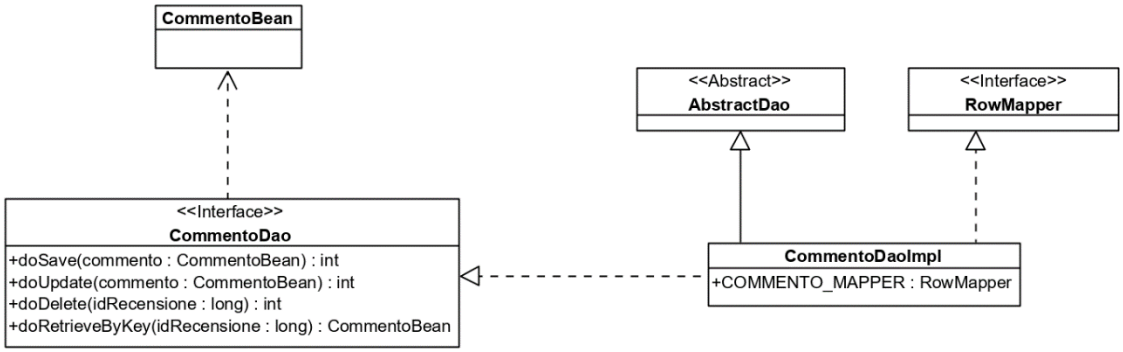
```
context AppartenenzaStileDao::doSave(asb:AppartenenzaStileBean) pre:
  asb != null and
  asb.idInserzione != null and
  asb.nomeStile != null and
  doRetrieveByKey(asb.idInserzione, asb.nomeStile) = null
context AppartenenzaStileDao::doSave(asb:AppartenenzaStileBean) post:
  result = 1 and
  doRetrieveByKey(asb.idInserzione, asb.nomeStile) != null

context AppartenenzaStileDao::doDelete(idInserzione:long, nomeStile:String) pre:
  idInserzione != null and
  nomeStile != null and
  doRetrieveByKey(asb.idInserzione, asb.nomeStile) != null
context AppartenenzaStileDao::doDelete(idInserzione:long, nomeStile:String) post:
  result = 1 and
  doRetrieveByKey(idInserzione, nomeStile) = null

context AppartenenzaStileDao::doRetrieveByKey(idInserzione:long, nomeStile:String) pre:
  idInserzione != null and
  nomeStile != null and
  esiste asb:AppartenenzaStileBean nel database :
            asb.idInserzione = idInserzione and
            adb.nomeStile = nomeStile
context AppartenenzaStileDao::doRetrieveByKey(idInserzione:long, nomeStile:String) post:
  result = asb

context AppartenenzaStileDao::doRetrieveByIdInserzione(idInserzione:long) pre:
  idInserzione != null
context AppartenenzaStileDao::doRetrieveByIdInserzione(idInserzione:long) post:
  result = ogni asb:AppartenenzaStileBean nel database : asb.idInserzione = idInserzione
```

CommentoDao

Metodi:

- int doSave(CommentoBean commento);
- int doUpdate(CommentoBean commento);
- int doDelete(long idRecensione);
- CommentoBean doRetrieveByKey(long idRecensione).
-

Diagramma di classe:



Contratto:

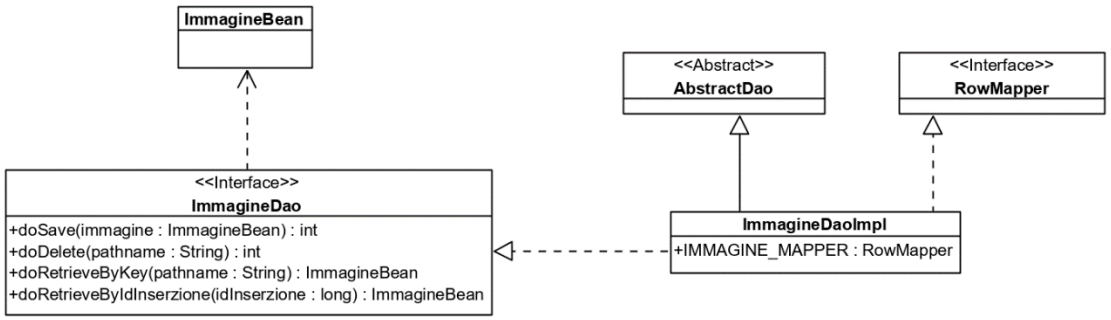
```
context CommentoDao::doSave(cb:CommentoBean) pre:
  cd != null and
  cd.idRecensione != null and
  cd.contenuto != null and
  cd.dataPubblicazione != null and
  doRetrieveByKey(cb.idRecensione)
context CommentoDao::doSave(cb:CommentoBean) post:
  result = 1 and
  doRetrieveByKey(cb.idRecensione) != null

context CommentoDao::doUpdate(cb:CommentoBean) pre:
  cd != null and
  cd.idRecensione != null and
  cd.contenuto != null and
  cd.dataPubblicazione != null and
  doRetrieveByKey(cb.idRecensione) != null
context CommentoDao::doUpdate(cb:CommentoBean) post:
  result = 1

context CommentoDao::doDelete(idRecensione:long) pre:
  idRecensione != null and
  doRetrieveByKey(cb.idRecensione) != null
context CommentoDao::doDelete(idRecensione:long) post:
  result = 1 and
  doRetrieveByKey(cb.idRecensione) = null

context CommentoDao::doRetrieveByKey(idRecensione:long) pre:
  idRecensione != null and
  esiste cb.Commentobean nel database :
            cb.idRecensione =  idRecensione
context CommentoDao::doRetrieveByKey(idRecensione:long) post:
  result = cb
```

ImmagineDao

Metodi:

- int doSave(ImmagineBean immagine);
- int doDelete(String pathname);
- ImmagineBean doRetrieveByKey(String pathname);
- java.util.List<ImmagineBean> doRetrieveByIdInserzione(long idInserzione).

Diagramma di classe:



Contratto:

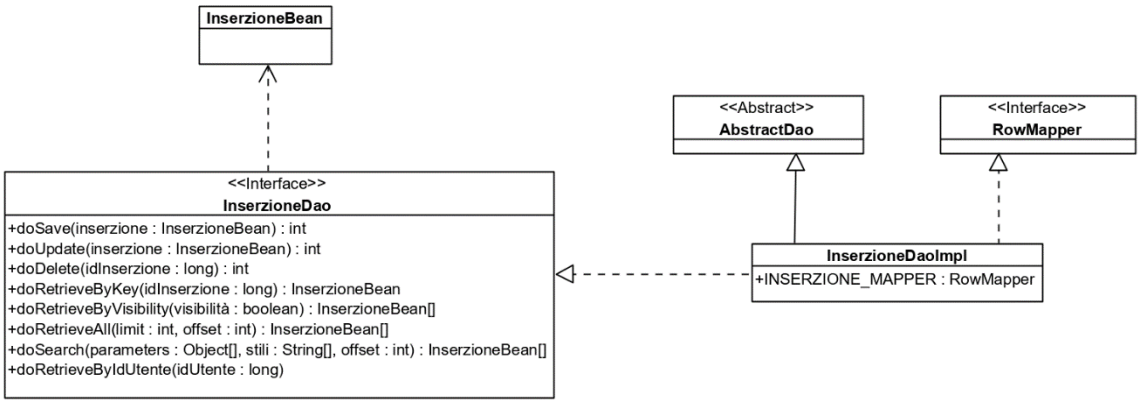
```
context ImmagineDao::doSave(ib:ImmagineBean) pre:
  ib != null and
  ib.pathname != null and
  ib.idInserzione != null and
  doRetrieveByKey(ib.pathname) = null
context ImmagineDao::doSave(ib:ImmagineBean) post:
  result = 1 and
  doRetrieveByKey(ib.pathname) != null

context ImmagineDao::doDelete(pathname:String) pre:
  pathname != null and
  doRetrieveByKey(pathname) != null
context ImmagineDao::doDelete(pathname:String) post:
  result = 1 and
  doRetrieveByKey(pathname) = null

context ImmagineDao::doRetrieveByKey(pathname:String) pre:
  pathname != null and
  esiste ib:ImmagineBean nel database : ib.pathname = pathname
context ImmagineDao::doRetrieveByKey(pathname:String) post:
  result = ib

context ImmagineDao::doRetrieveByIdInserzione(idInserzione:long) pre:
  idInserzione != null
context ImmagineDao::doRetrieveByIdInserzione(idInserzione:long) post:
  result = ogni ib:ImmagineBean nel database : ib.idInserzione = idInserzione
```

InserzioneDao

Metodi:

- long doSave(InserzioneBean inserzione);
- int doUpdate(InserzioneBean inserzione);
- int doDelete(long idInserzione);
- InserzioneBean doRetrieveByKey(long idInserzione);
- java.util.List<InserzioneBean> doRetrieveByVisibility(boolean visibilità);
- java.util.List<InserzioneBean> doRetrieveAll(int limit, int offset);
- java.util.List<InserzioneBean> doSearch(Java.util.List<Object> parameters, String[] stili, int offset).

Diagramma di classe:



Contratto:

```
context InserzioneDao::doSave(ib:InserzioneBean) pre:
  ib != null and
  ib.idInserzione != null and
  ib.idProprietario != null and
  ib.stato != null and
  ib.regione != null and
  ib.città != null and
  ib.cap != null and
  ib.strada != null and
  ib.descrizione != null and
  ib.numeroCivico != null and
  ib.maxNumeroOspiti != null and
  ib.prezzoGiornaliero != null and
  ib.metratura != null and
  ib.dataInserimento != null and
  ib.visibilità != null and
context InserzioneDao::doSave(ib:InserzioneBean) post:
  result = AUTO_GENERATED_KEY and
  doRetrieveByKey(AUTO_GENERATED_KEY) != null

context InserzioneDao::doUpdate(ib:InserzioneBean) pre:
  ib != null and
  ib.idInserzione != null and
  ib.idProprietario != null and
  ib.stato != null and
  ib.regione != null and
```

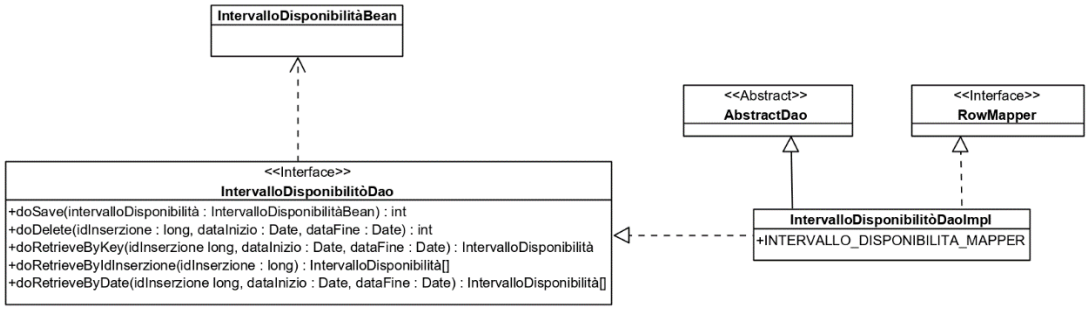
```
  ib.città != null and
  ib.cap != null and
  ib.strada != null and
  ib.descrizione != null and
  ib.numeroCivico != null and
  ib.maxNumeroOspiti != null and
  ib.prezzoGiornaliero != null and
  ib.metratura != null and
  ib.dataInserimento != null and
  ib.visibilità != null and
  doRetrieveByKey(ib.idInserzione) != null
context InserzioneDao::doUpdate(ib:InserzioneBean) post:
  result = 1

context InserzioneDao::doDelete(idInserzione:long) pre:
  idInserzione != null and
  doRetrieveByKey(ib.idInserzione) != null
context InserzioneDao::doDelete(idInserzione:long) post:
  result = 1 and
  doRetrieveByKey(ib.idInserzione) = null

context InserzioneDao::doRetrieveByKey(idInserzione:long) pre:
  idInserzione != null and
  esiste ib:InserzioneBean nel database : ib.idInserzione = idInserzione
context InserzioneDao::doRetrieveByKey(idInserzione:long) post:
  result = ib

context InserzioneDao::doRetrieveByVisibility(visibilità:boolean) pre:
  visibilità != null
context InserzioneDao::doRetrieveByVisibility(visibilità:boolean) post:
  result = ogni ib:InserzioneBean nel database : ib.visibilità = visibilità

context InserzioneDao::doRetrieveAll(limit:int, offset:int) pre:
  limit != null and
  offset != null
context InserzioneDao::doRetrieveAll
  result = i primi limit ib:InserzioneBean nel database a partire da offset

context InserzioneDao::doSearch(parametri:String, stili:StileBean, offset) pre:
  parametri != null and
  stili != null and
  offset != null and
context InserzioneDao::doSearch(parametri:String, stili:StileBean, offset) post:
  result = ogni ib:InserzioneBean nel database : ib.parametri in parametri and
           ib.stili in stili a partire da offset

context InserzioneDao::doRetrieveByIdUtente(idUtente:long) pre:
  idUtente != null
context InserzioneDao::doRetrieveByIdUtente(idUtente:long) post:
  result = ogni ib:InserzioneBean nel database : ib.idProprietario = idUtente
```

IntervalloDisponibilitàDao

Metodi:

- int doSave(IntervalloDisponibilitàBean intervalloDisponibilità);
- int doDelete(long idInserzione, Date dataInizio, Date dataFine);
- IntervalloDisponibilitàBean doRetrieveByKey(long idInserzione, Date dataInizio, Date dataFine);
- java.util.List<IntervalloDisponibilitàBean> doRetrieveByIdInserzione(long idInserzione);
- IntervalloDisponibilitàBean doRetrieveByDate(long idInserzione, Date dataInizio, Date dataFine).

Diagramma di classe:



Contratto:

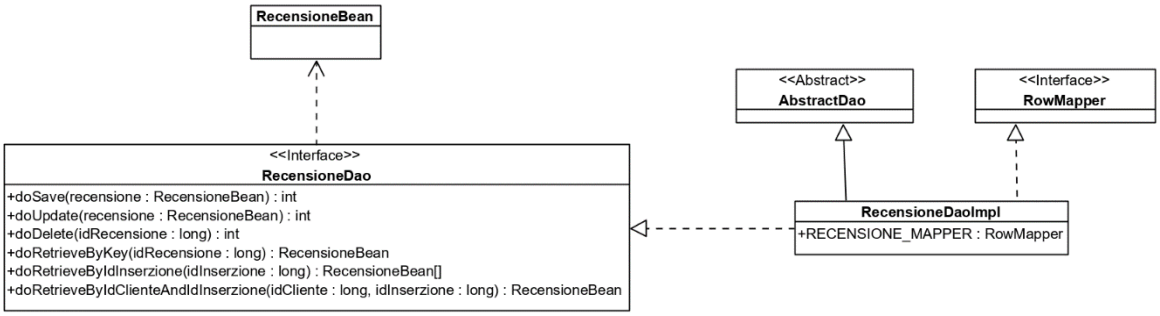
```
context IntervalloDisponibilitàDao::doSave(ib:IntervalloDisponibilitàBean) pre:
  ib != null and
  ib.idInserzione != null and
  ib.dataInizio != null and
  ib.dataFine != null
  doRetireveByKey(ib.idInserzione, ib.dataInizio, ib.dataFine) = null
context IntervalloDisponibilitàDao::doSave(id:IntervalloDisponibilitàBean) post:
  result = 1 and
  doRetireveByKey(ib.idInserzione, ib.dataInizio, ib.dataFine) != null

context IntervalloDisponibilitàDao::doDelete(idInserzione:long, dataInizio:Date, dataFine:Date) pre:
  idInserzione != null and
  dataInizio != null and
  dataFine != null
  doRetireveByKey(idInserzione, dataInizio, dataFine) != null
context IntervalloDisponibilitàDao::doDelete(idInserzione:long, dataInizio:Date, dataFine:Date) post:
  result = 1 and
  doRetireveByKey(idInserzione, dataInizio, dataFine) = null

context IntervalloDisponibilitàDao::doRetrieveByIdInserzione(idInserzione:long) pre:
  idInserzione != null
context IntervalloDisponibilitàDao::doRetrieveByIdInserzione(idInserzione:long) post:
  result = ogni idb:IntervalloDisponibilitàBean nel database : idb.idInserzione = idInsezione

context IntervalloDisponibilitàDao::doRetrieveByDate(dataInizio:Date, dataFine:Date) pre:
  dataInizio != null and
  dataFine != null
  esiste idb:IntervalloDisponibilitàBean nel database :
            idb.dataInizio <= dataInizio and
            idb.dataFine >= dataFine
context IntervalloDisponibilitàDao::doRetrieveByDate(dataInizio:Date, dataFine:Date) post:
  result = idb
```

RecensioneDao

Metodi:

- long doSave(RecensioneBean recensione);
- int doUpdate(RecensioneBean recensione);
- int doDelete(long idRecensione);
- RecensioneBean doRetrieveByKey(long idRecensione);
- java.util.List<RecensioneBean> doRetrieveByIdInserzione(long idInserzione).

Diagramma di classe:



Contratto:

```
context RecensioneDao::doSave(rb:RecensioneBean) pre:
  rb != null and
  rb.idCliente != null and
  rb.idInserzione != null and
  rb.punteggio != null and
  rb.titolo != null and
  rb.contenuto != null and
  rb.dataPubblicazione != null and
context RecensioneDao::doSave(rb:RecensioneBean) post:
  result = AUTO_GENERATED_KEY and
  doRetrieveByKey(AUTO_GENERATED_KEY) != null

context RecensioneDao::doUpdate(rb:RecensioneBean) pre:
  rb != null and
  rb.idRecensione != null and
  rb.idCliente != null and
  rb.idInserzione != null and
  rb.punteggio != null and
  rb.titolo != null and
  rb.contenuto != null and
  rb.dataPubblicazione != null and
  doRetrieveByKey(rb.idRecensione) != null
context RecensioneDao::doUpdate(rb:RecensioneBean) post:
  result = 1

context RecensioneDao::doDelete(idRecensione:long) pre:
  idRecensione != null and
  doRetrieveByKey(rb.idRecensione) != null
context RecensioneDao::doDelete(rb:RecensioneBean) post:
  result = 1 and
  doRetrieveByKey(rb.idRecensione) = null
```

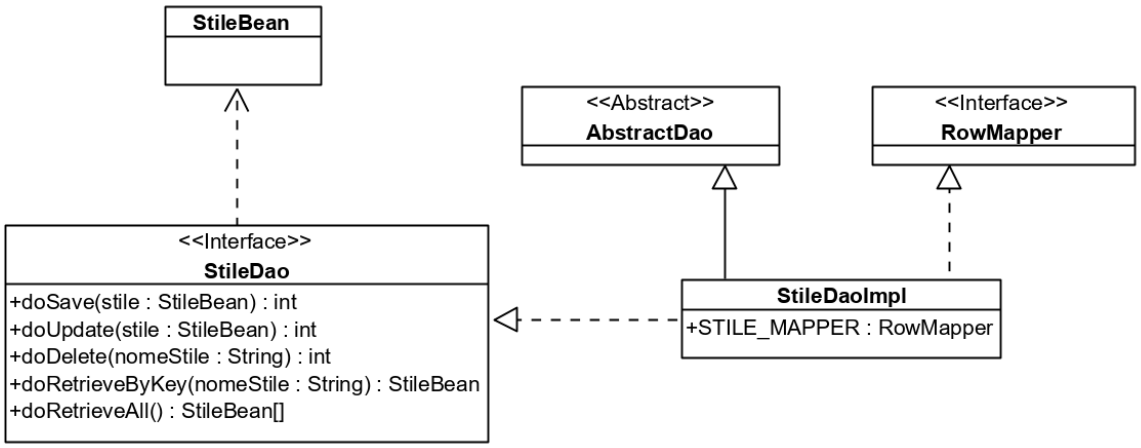
```
context RecensioneDao::doRetrieveByKey(idRecensione:long) pre:
  idRecensione != null and
  esiste rb:RecensioneBean nel database : rb.idRecensione = idRecensione
context RecensioneDao::doRetrieveByKey(rb:RecensioneBean) post:
  result = rb

context RecensioneDao::doRetrieveByIdInsezione(idInserzione:long) pre:
  idInserzione != null
context RecensioneDao::doRetrieveByKey(rb:RecensioneBean) post:
  result = ogni rb:RecensioneBean nel database : rb.idInserzione = idInserzione

context RecensioneDao::doRetrieveByIdClienteAndIdInsezione(idCliente:long, idInserzione:long) pre:
  idCliente != null and
  idInserzione != null
context RecensioneDao::doRetrieveByKey(rb:RecensioneBean) post:
  result = ogni rb:RecensioneBean nel database :
            rb.idCliente = idCliente and
            rb.idInserzione = idInserzione
```

StileDao

Metodi:

- int doSave(StileBean stile);
- int doUpdate(StileBean stile);
- int doDelete(String nomeStile);
- StileBean doRetrieveByKey(String nomeStile);
- java.util.List<StileBean> doRetrieveAll().

Diagramma di classe:



Contratto:

```
context StileDao::doSave(sb:StileBean) pre:
  sb != null and
  sb.nomeStile != null and
  sb.descrizione != null and
  doRetrieveByKey(sb.nomeStile) = null
```

```
context StileDao::doSave(sb:StileBean) post:
  result = 1 and
  doRetrieveByKey(sb.nomeStile) != null

context StileDao::doUpdate(sb:StileBean) pre:
  sbdescrizione != null and
  doRetrieveByKey(sb.nomeStile) != null
context StileDao::doUpdate(descrizione:String) post:
  result = 1

contet StileDao::doDelete(nomeStile:String) pre:
  nomeStile != null and
  doRetrieveByKey(nomeStile) != null
contet StileDao::doDelete(nomeStile:String) post:
  result = 1 and
  doRetrieveByKey(sb.nomeStile) = null

context StileDao::doRetrieveByKey(nomeStile:String) pre:
  nomeStile != null and
  esiste sb:StileBean nel database : sb.nomeStile = nomeStile
context StileDao::doRetrieveByKey(nomeStile:String) post:
  result = sb

context StileDao::doRetrieveAll() pre:
context StileDao::doRetrieveAll() post:
  result = sb:StileBean nel database
```

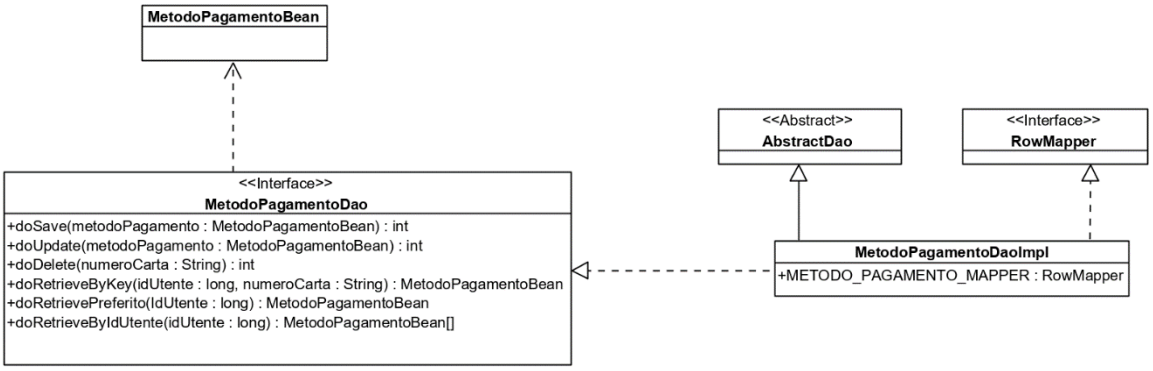
2.1.2.3 PRENOTAZIONI

MetodoPagamentoDao

Metodi:

- int doSave(MetodoPagamentoBean metodoPagamento);
- int doUpdate(MetodoPagamentoBean metodoPagamento);
- int doDelete(String numeroCarta);
- MetodoPagamentoBean doRetrieveByKey(String numeroCarta);
- MetodoPagamentoBean doRetrieveByPreferito(boolean preferito);
- MetodoPagamentoBean doRetrievePreferito(long idUtente);
- java.util.List<MetodoPagamentoBean> doRetrieveByIdUtente(long idUtente).

Diagramma di classe:



Contratto:

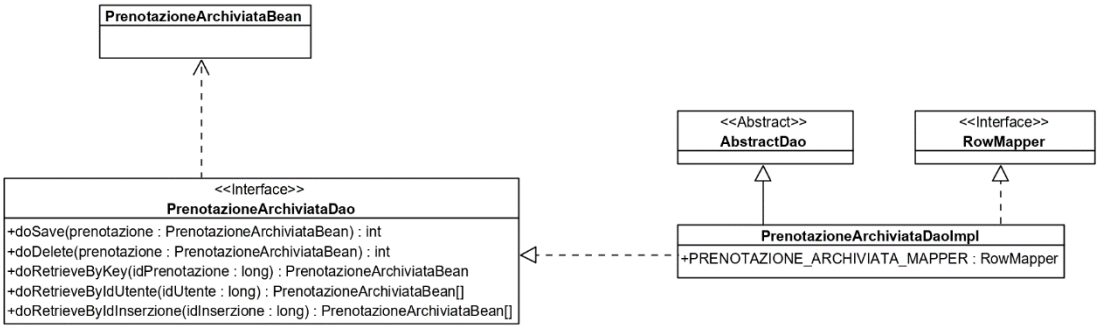
```
context MetodoPagamentoDao::doSave(mpb:MetodoPagamentoBean) pre:
  mpb != null and
  mpb.numeroCarta != null and
  mpb.nomeTitolare != null and
  mpb.cognomeTitolare != null and
  mpb.dataScadenza != null and
  mpb.preferito != null and
  doRetrieveByKey(mpb.numeroCarta) = null
context MetodoPagamentoDao::doSave(mpb:MetodoPagamentoBean) post:
  result = 1 and
  doRetrieveByKey(mpb.numeroCarta) != null

context MetodoPagamentoDao::doUpdate(mpb:MetodoPagamentBean) pre:
  mpb != null and
  mpb.numeroCarta != null and
  mpb.nomeTitolare != null and
  mpb.cognomeTitolare != null and
  mpb.dataScadenza != null and
  mpb.preferito != null and
  doRetrieveByKey(mpb.numeroCarta) != null
context MetodoPagamentoDao::doUpdate(mpb:MetodoPagamentBean) post:
  result = 1

context MetodoPagamentoDao::doDelete(numeroCarta:String) pre:
  numeroCarta != null and
  doRetrieveByKey(mpb.numeroCarta) != null
context MetodoPagamentoDao::doDelete(numeroCarta:String) post:
  result = 1 and
  doRetrieveByKey(mpb.numeroCarta) = null

context MetodoPagamentoDao::doRetrieveByKey(numeroCarta:String) pre:
  numeroCarta != null and
  esiste mpb:MetodoPagamentoBean nel database : mpb.numeroCarta = numeroCarta
context MetodoPagamentoDao::doRetrieveByKey(numeroCarta:String) post:
  result = mpb

context MetodoPagamentoDao::doRetrievePreferito(idUtente:long) pre:
  idUtente != null and
  esiste mpb:MetodoPagamentoBean nel database :
            mpb.idUtente = idUtente and
            mpb.preferito = true
context MetodoPagamentoDao::doRetrievePreferito(idUtente:long) post
  result = mpb

context MetodoPagamentoDao::doRetireveByIdUtente(idUtente:long) pre:
  idUtente != null
context MetodoPagamentoDao::doRetireveByIdUtente(idUtente:long) post:
  result ogni mpb:MetodoPagamentoBean nel database : mpb.idUtente = idUtente
```

PrenotazioneArchiviataDao

Metodi:

- int doSave(PrenotazioneArchiviataBean pab);
- int doUpdate(PrenotazioneArchiviataBean pab);
- PrenotazioneArchiviataBean
- java.util.List< PrenotazioneArchiviataBean> doRetrieveByIdUtente(long idUtente);
- java.util.List< PrenotazioneArchiviataBean> doRetrieveByIdInserzione(long idInserzione).

Diagramma di classe:



Contratto:

```
context PrenotazioneArchiviataDao::doSave(pab:PrenotazioneArchiviataBean) pre:
  pab != null and
  pab.idPrenotazione != null and
  pab.stato != null and
  pab.regione != null and
  pab.città != null and
  pab.cap != null and
  pab.indirizzo != null and
  pab.prezzoGiornaliero != null and
  doRetrieveByKey(pab.idPrenotazione) = null
context PrenotazioneArchiviataDao::doSave(pab:PrenotazioneArchiviataBean) post:
  result = 1 and
  doRetrieveByKey(pab.idPrenotazione) != null

context PrenotazioneArchiviataDao::doUpdate(pab:PrenotazioneArchiviataBean) pre:
  pab.idPrenotazione != null and
  pab.stato != null and
  pab.regione != null and
  pab.città != null and
  pab.cap != null and
  pab.indirizzo != null and
  pab.prezzoGiornaliero != null and
  doRetrieveByKey(pab.idPrenotazione) != null
context PrenotazioneArchiviataDao::doUpdate(pab:PrenotazioneArchiviataBean) post:
  result = 1

context PrenotazioneArchiviataDao::doDelete(idPrenotazione:long) pre:
  idPrenotazione != null
  doRetrieveByKey(pb.getIdPrenotazione) != null
context PrenotazioneArchiviataDao::doDelete(idPrenotazione:long) pre:
```

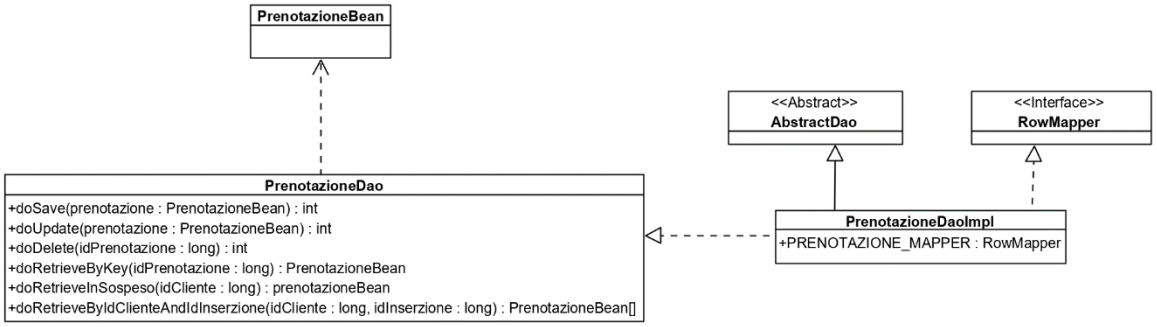
```
  result = 1 and
  doRetrieveByKey(pb.getIdPrenotazione) = null

context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) pre:
  idPrenotazione != null
  esiste pab:PrenotazioneArchiviataBean nel database : pab.idPrenotazione = idPrenotazione
context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) post:
  result = pab

context PrenotazioneArchiviataDao::doRetrieveByIdUtente(idUtente:long) pre:
  idUtente != null
context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) post:
  result ogni pab:PrenotazioneArchiviataBean nel database : pab.idCliente = idUtente

context PrenotazioneArchiviataDao::doRetrieveByIdInserzione(idInserzione:long) pre:
  idInserzione != null
context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) post:
  result ogni pab:PrenotazioneArchiviataBean nel database : pab.idInserzione = idInserzione
```

PrenotazioneDao

Metodi:

- long doSave(PrenotazioneBean prenotazione);
- int doUpdate(PrenotazioneBean prenotazione);
- int doDelete(long idPrenotazione);
- PrenotazioneBean doRetrieveByKey(long idPrenotazione);
- PrenotazioneBean doRetrieveInSospeso(long idCliente).
- java.util.List<PrenotazioneBean> doRetrieveByIdClienteAndIdInserzione(long idCliente, long idInserzione)

Diagramma di classe:



Contratto:

```
context PrenotazioneDao::doSave(pb:PrenotazioneBean) pre:
  pb != null
context PrenotazioneDao::doSave(pb:PrenotazioneBean) post:
  result = AUTO_GENERATED_KEY and
  doRetrieveByKey(AUTO_GENERATED_KEY) != null

context PrenotazioneDao::doSave(pb:PrenotazioneBean) pre:
  pb != null and
```

```
  doRetrieveByKey(pb.getIdPrenotazione) != null
context PrenotazioneDao::doSave(pb:PrenotazioneBean) post:
  result = 1 and

context PrenotazioneDao::doDelete(idPrenotazione:long) pre:
  idPrenotazione != null
  doRetrieveByKey(pb.getIdPrenotazione) != null
context PrenotazioneDao::doDelete(idPrenotazione:long) pre:
  result = 1 and
  doRetrieveByKey(pb.getIdPrenotazione) = null

context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) pre:
  idPrenotazione != null
  esiste pb:PrenotazioneBean nel database : pb.idPrenotazione = idPrenotazione
context PrenotazioneArchiviataDao::doRetrieveByKey(idPrenotazione:long) post:
  result = pb

context PrenotazioneArchiviataDao::doRetrieveInSospeso(idCliente:long) pre:
  idCliente != null
  esiste pb:PrenotazioneBean nel database :
            pb.stato = 0 and
            pb.idCliente = idCliente
context PrenotazioneArchiviataDao::doRetrieveInSospeso(idCliente:long) post:
  result = pb

context PrenotazioneDao::doRetrieveByIdClienteAndIdInserzione(idCliente:long, idInserzione:long)
pre:
  idCliente != null and
  idInserzione != null
context PrenotazioneDao::doRetrieveByIdClienteAndIdInserzione(idCliente:long, idInserzione:long)
post:
  ogni pb:PrenotazioneBean nel database :
            pb.idCliente = idCliente and
            pb.idInserzione = idInserzione
```

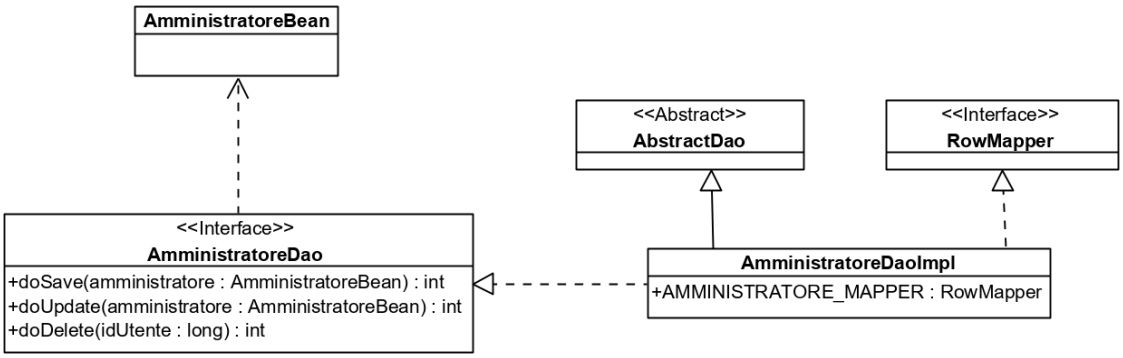
2.1.2.4 UTENZA

AmministratoreDao

Metodi:

- int doSave(AmministratoreBean amministratore);
- int doUpdate(AmministratoreBean amministratore);
- int doDelete(long idUtente).

Diagramma di classe:



Contratto:

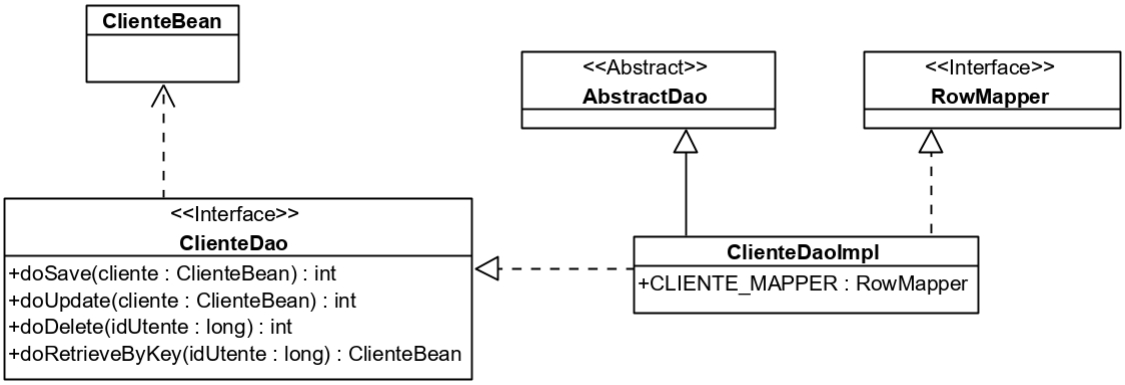
```
context AmministratoreDao::doSave(ab:AmministratoreBean) pre:
  ab != null and
  ab.idUtente != null and
  ab.numInserzioniRevisionate != null
  doRetrieveByKey(ab.idUtente) = null
context AmministratoreDao::doSave(ab:AmministratoreBean) post:
  result = 1 and
  doRetrieveByKey(ab.idUtente) != null

context AmministratoreDao::doUpdate(ab:AmministratoreBean) pre:
  ab != null and
  ab.idUtente != null and
  ab.numInserzioniRevisionate != null
  doRetrieveByKey(ab.idUtente) != null
context AmministratoreDao::doUpdate(ab:AmministratoreBean) post:
  result = 1

context AmministratoreDao::doDelete(idUtente:long) pre:
  idUtente != null and
  doRetrieveByKey(ab.idUtente) != null
  esiste ab:AmministratoreBean nel database : ab.idUtente != idUtente
context AmministratoreDao::doDelete(idUtente) post:
  result = 1 and
  doRetrieveByKey(ab.idUtente) = null
```

ClienteDao

Metodi:

- int doSave(ClienteBean cliente);
- int doUpdate(ClienteBean cliente);
- int doDelete(long idUtente);
- ClienteBean doRetrieveByKey(long idUtente).

Diagramma di classe:



Contratto:

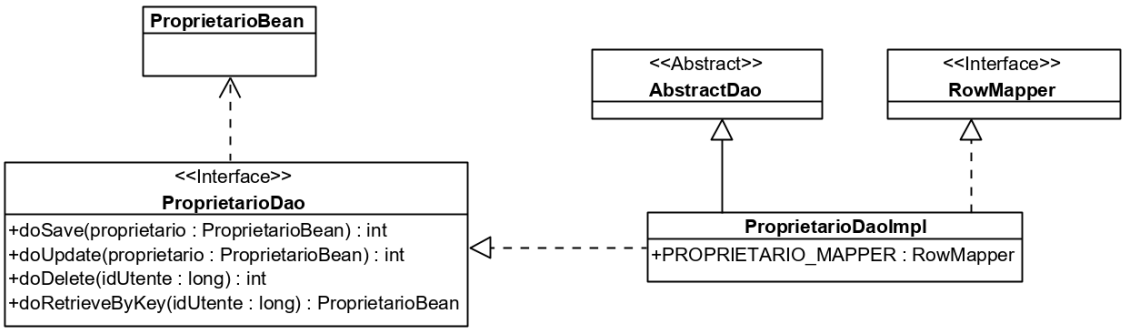
```
context ClienteDao::doSave(cb:ClienteBean) pre:
  cb != null and
  cb.idUtente != null
  cb.dataUltimaPrenotazione != null and
  doRetireveByKey(cb.idUtente) = null
context ClienteDao::doSave(cb:ClienteBean) post:
  result = 1 and
  doRetireveByKey(cb.idUtente) != null

context ClienteDao::doUpdate(cb:ClienteBean) pre:
            cb != null and
  cb.idUtente != null
  cb.dataUltimaPrenotazione != null and
  doRetireveByKey(cb.idUtente) != null
context ClienteDao::doUpdate(cb:ClienteBean) pre:
  result = 1

context ClienteDao::doDelete(idUtente:long) pre:
  idUtente != null and
  doRetireveByKey(idUtente) != null
context ClienteDao::doDelete(idUtente:long) post:
  doRetireveByKey(cb.idUtente) = null

context ClienteDao::doRetrieveByKey(idUtente:long) pre:
  idUtente != null and
  esiste cb:ClienteBean nel database : cb.idUtente = idUtente
context ClienteDao::doRetrieveByKey(idUtente:long) post:
  result = cb
```

ProprietarioDao

Metodi:

- int doSave(ProprietarioBean proprietario);
- int doUpdate(ProprietarioBean proprietario);
- int doDelete(long idUtente);
- ProprietarioBean doRetrieveByKey(long idUtente).

Diagramma di classe:



Contratto:

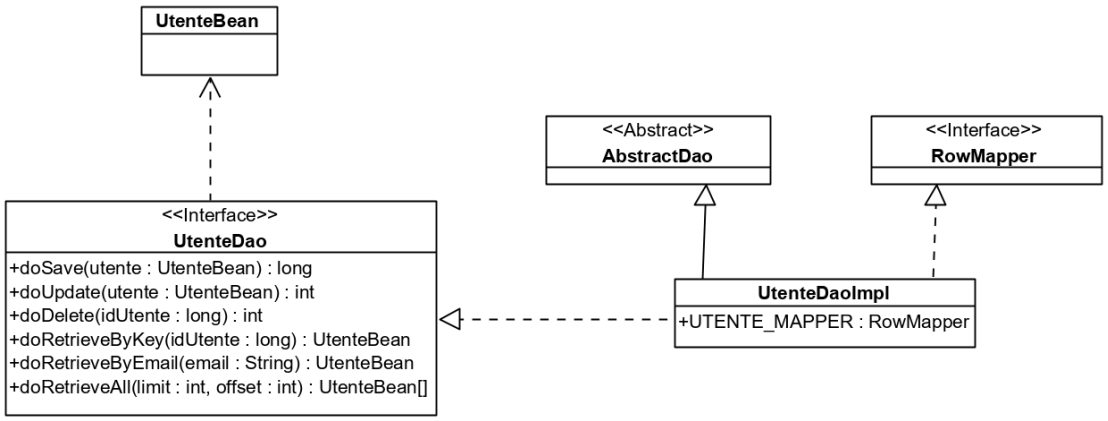
```
context ProprietarioDao::doSave(pb:ProprietarioBean) pre:
  pb.idUtente != null and
  pb.codiceFiscale != null and
  pb.numInserzioniInserite != null and
  doRetrieveByKey(pb.idUtente) = null
context ProprietarioDao::doSave(pb:ProprietarioBean) post:
  result = 1 and
  doRetrieveByKey(pb.idUtente) != null

context ProprietarioBean::doUpdate(pb:ProprietarioBean) pre:
  pb.idUtente != null and
  pb.codiceFiscale != null and
  pb.numInserzioniInserite != null and
  doRetrieveByKey(pb.idUtente) != null
context ProprietarioBean::doUpdate(pb:ProprietarioBean) post:
  result = 1

context ProprietarioDao::doDelete(idUtente:long) pre:
  idUtente != null and
  doRetireveByKey(idUtente) != null
context ProprietarioDao::doDelete(idUtente:long) pre:
  doRetireveByKey(idUtente) = null

context ProprietarioDao::doRetrieveByKey(idUtente:long) pre:
  idUtente != null and
  esiste pb:ProprietarioBean nel database : pb.idUtente = idUtente
context ProprietarioDao::doRetrieveByKey(idUtente:long) pre:
  result = pb
```

UtenteDao

Metodi:

- long doSave(UtenteBean utente);
- int doUpdate(UtenteBean utente);
- int doDelete(long idUtente);
- UtenteBean doRetrieveByKey(long idUtente);
- UtenteBean doRetrieveByEmail(String email);
- java.util.List<UtenteBean> doRetrieveAll(int limit, int offset).

Diagramma di classe:



Contratto:

```
context UtenteDao::doSave(ub:UtenteBean) pre:
  ub != null and
  ub.email != null and
  ub.password != null and
  ub.nome != null and
  ub.cognome != null and
  ub.stato != null and
  doRetrieveByEmail(ub.email) = null
context UtenteDao::doSave(ub:UtenteBean) post:
  result = AUTO_GENERATED_KEY
  doRetrieveByKey(AUTO_GENERATED_KEY) != null

context UtenteDao::doUpdate(ub:UtenteBean) pre:
  ub != null and
  ub.idUtente != null
  ub.email != null and
  ub.password != null and
  ub.nome != null and
  ub.cognome != null and
  ub.stato != null and
  doRetrieveByKey(ub.idUtente) != null
context UtenteDao::doUpdate(ub:UtenteBean) post:
  result = 1

context UtenteDao::doDelete(idUtente:long) pre:
  idUtente != null and
```

```
  doRetrieveByKey(idUtente) != null
context UtenteDao::doDelete(idUtente:long) post:
  doRetrieveByKey(idUtente) = null

context UtenteDao::doRetrieveByKey(idUtente:long) pre:
  idUtente != null and
  esiste UtenteBean ub nel database : ub.idUtente = Utente
context UtenteDao::doRetrieveByKey(id:long) post:
  result = ub

context UtenteDao::doRetrieveByEmail(email:String) pre:
  email != null and
  esiste UtenteBean ub nel database : ub.email = email
context UtenteDao::doRetrieveByEmail(email:String) post:
  result = ub

context UtenteDao::doRetrieveAll(limit:int, offset:int) pre:
  limit != null and
  offset != null
context UtenteDao::doRetrieveAll(limit:int, offset:int) post:
  result = i primi limit ub:UtenteBean nel database a partire da offset
```

2.2 VIEW

Insieme di pagine web dinamiche, in formato JSP, che regolano l'interazione con l'utente.

```
view
├──inserzioni
│       ├──inserimentoInserzione
│       ├──inserzione
│       └──listaInserzioni
│
├──prenotazioni
│       ├──storicoPrenotazioniUtente
│       └──storicoPrenotazioniInserzione
│
├──utenza
│       ├──paginaAmministrazione
│       └──profiloUtente
│
├──error
├──footer
├──header
├──homepage
├──notAuthorized
└──orionException
```

2.2.1 *INSERZIONI*

Pagina	Descrizione
inserimentoInserzione	Pagina per l'inserimento di un'inserzione.
inserzione	Pagina contenente tutte le informazioni su un'inserzione.
listaInserzioni	Pagina contenente tutte le informazioni su una generica lista di inserzioni.

2.2.2 *PRENOTAZIONI*

Pagina	Descrizione
storicoPrenotazioniUtente	Pagina contenente la lista delle prenotazioni effettuate da un cliente.
storicoPrenotazioniInserzione	Pagina contenente le informazioni di tutti i clienti che hanno prenotato un'inserzione

2.2.3 *UTENZA*

Pagina	Descrizione
paginaAmministrazione	Pagina contentente tutte le funzionalità di amministrazione.
profiloUtente	Pagina contente tutte le credenziali utente ed i metodi di pagamento.

2.2.4 *ALTRE PAGINE*

Pagina	Descrizione
error	Pagina di errore generica.
footer	Footer di fine pagina.
header	Header presente in ogni pagina.
homepage	Pagina principale di Orion
notAuthorized	Pagina di errore per eccezioni di sicurezza.
orionException	Pagina di errore per eccezioni generiche.

2.3 *CONTROLLER*

Insieme di classi Java che si occupano di gestire le richieste utente.

```
controller
├──gestioneInserzioni
│       ├──ApprovaInserzione
│       ├──InserimentoCommento
│       ├──InserimentoRecensione
│       ├──InserimentoStile
│       ├──Inserzione
│       ├──InserzioniProrietario
│       ├──ModificaStile
│       ├──RicercaInserzione
│       ├──RimozioneInserzione
│       ├──RimozioneRecensione
│       └──RimozioneStile
│
├──gestionePrenotazioni
│       ├──AnnullamentoPrenotazione
│       ├──AvvioPrenotazione
│       ├──CompletamentoPrenotazione
│       ├──InserimentoMetodoPagamento
│       ├──ModificaMetodoPagamento
│       ├──RimozioneMetodoPagamento
│       ├──StoricoPrenotazioniInserzione
│       └──StoricoPrenotazioniUtente
│
├──gestioneUtenza
│       ├──Accesso
│       ├──AjaxEmail
│       ├──Amministrazione
│       ├──Logout
│       ├──ModificaCredenziali
│       ├──ProfiloUtente
│       ├──Registrazione
│       ├──RiabilitazioneAccount
│       ├──RimozioneAccount
│       └──SospensioneAccount
│
├──BaseServlet
├──Homepage
├──NotAuthorizedException
├──OrionException
└──OrionImages
```

2.3.1 CONTROLLOINSERZIONI

Servlet	Descrizione
ApprovaInserzione	Cambia la visibilità di un'inserzione.
InserimentoCommento	Inserisce un commento relativo ad una recensione.
InserimentoInserzione	Gestisce il procedimento di inserimento di un'inserzione.
InserimentoRecensione	Valuta la possibilità di inserire una recensione e, in caso ciò sia possibile, procede alla validazione delle informazioni inserite dall'utente.
InserimentoStile	Inserisce un nuovo stile.
Inserzione	Recupera le informazioni da mostrare nella pagina di un'inserzione.
InserzioniProprietario	Recupera la lista delle inserzioni inserite da un proprietario.
ModificaStile	Modifica la descrizione associata ad uno stile precedentemente inserito.
RicercaInserzione	Recupera una lista di inserzioni che rispettano i parametri inseriti.
RimozioneInserzione	Rimuove un'inserzione precedentemente inserita, validando i permessi di chi è interessato.
RimozioneRecensione	Rimuove una recensione precedentemente inserita.
RimozioneStile	Rimuove uno stile precedentemente inserito.

2.3.2 CONTROLLOPRENOTAZIONI

Servlet	Descrizione
AnnullamentoPrenotazione	Annulla una prenotazione precedentemente avviata.
AvvioPrenotazione	Avvia la procedura di prenotazione relativa ad un'inserzione.
CompletamentoPrenotazione	Completa una prenotazione precedentemente avviata, controllando la scadenza dei metodi di pagamento di cliente e proprietario e del timer avviato precedentemente.
InserimentoMetodoPagamento	Inserisce un nuovo metodo di pagamento.
ModificaMetodoPagamento	Modifica le informazioni associate ad un metodo di pagamento precedentemente inserito.
RimozioneMetodoPagamento	Rimuove un metodo di pagamento precedentemente inserito, aggiornando eventualmente il metodo preferito. Se un proprietario ha inserito un solo metodo di pagamento, questo non potrà essere rimosso.
StoricoPrenotazioniInserzione	Recupera la lista di tutti i clienti che hanno prenotato un'inserzione.
StoricoPrenotazioniUtente	Recupera tutte le prenotazioni archiviate effettuate da un cliente.

2.3.3 CONTROLLOUTENZA

Servlet	Descrizione
Accesso	Valida le credenziali di accesso e recupera le informazioni riferite all'utente.
AjaxEmail	Gestisce le richieste asincrone per verificare se un'e-mail è disponibile al momento della registrazione.
Amministrazione	Recupera le informazioni da mostrare nella pagina amministratore.
Logout	Rimuove i dati presenti nella sessione dell'utente ed effettua il logout.
ModificaCredenziali	Gestisce la modifica delle credenziali, validando i nuovi parametri inseriti ed aggiornando i vecchi.

ProfiloUtente	Recupera le informazioni da mostrare nel profilo utente.
Registrazione	Gestisce il procedimento di registrazione come cliente o proprietario.
RiabilitazioneAccount	Riabilita un account precedentemente sospeso.
RimozioneAccount	Rimuove permanentemente un account.
SospensioneAccount	Sospende un account fino a che non viene manualmente riabilitato.

2.3.4 ALTRI CONTROLLER ED ECCEZIONI

Servlet	Descrizione
BaseServlet	Servlet generica contenente metodi per la validazione dei dati.
Homepage	Recupera le informazioni da mostrare in homepage.
NotAuthorizedException	Eccezione lanciata per errori di autorizzazione.
OrionException	Eccezione lanciata per errori generici.
OrionImages	Gestisce il caricamento delle immagini sul server.