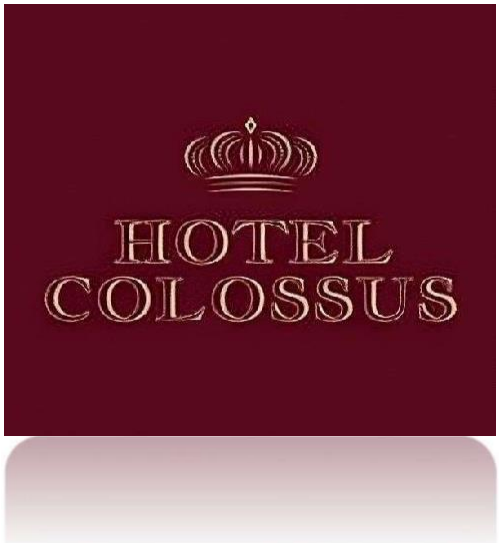
UNIVERSITÀ DEGLI STUDI
DI SALERNO



OBJECT DESIGN

TEAM COLOSSUS
Stefano Santoro – Giovanni Riccardi
-Renato Natale – Samuele Valiante

Coordinatore del progetto:

Nome	Matricola
Stefano Santoro	0512120778

Partecipanti:

Nome	Matricola
Giovanni Riccardi	0512119392
Renato Natale	0512119641
Samuele Valiante	0512119125

Scritto da:	Giovanni Riccardi

Revision History

Data	Versione	Descrizione	Autore
21/12/2025	1.0	Prima versione ODD	Giovanni Riccardi

Sommario

INTRODUCTION

Una volta completata la fase di system design, l'attenzione si s**post**a verso la realizzazione concreta del modello orientato agli oggetti definito durante l'analisi. In questa fase, l'obiettivo principale è raffinare e rendere implementabile quanto pattuito nel modello concettuale, mantenendo la tracciabilità con i requisiti.

Il class diagram, inizialmente introdotto durante l'analisi per descrivere i concetti del dominio e le loro relazioni, viene ora riesaminato e arricchito con dettagli necessari all'implementazione, come attributi concreti, tipi di dato, visibilità delle operazioni e realizzazione delle associazioni. Questo processo consente di verificare la coerenza strutturale del sistema e di individuare eventuali problemi di progettazione prima della codifica.

L'object diagram svolge un ruolo fondamentale nel validare le decisioni di progettazione, poiché chiarisce come le classi interagiscono effettivamente a runtime e supportano il mapping hardware/software di progetto verso il codice e le strutture di dati.

In sintesi, questa fase costituisce il ponte tra analisi e implementazione, trasformando

modelli astratti in una descrizione sufficientemente **pre**cisa da poter essere tradotta in codice sorgente e in strutture dati persistenti, nel rispetto dei vincoli architetturali e dei requisiti funzionali e non funzionali definiti nei documenti **pre**cedenti.

Tecnologie e Strumenti di sviluppo

Per la realizzazione del sistema di gestione dell'hotel sono stati adottati strumenti e tecnologie consolidate, selezionate con l'obiettivo di garantire **manutenibilità, modularità, testabilità e sicurezza** del software.
La gestione del ciclo di build e delle dipendenze è affidata a **Apache Maven**, che consente di automatizzare il processo di compilazione, testing e packaging del sistema, assicurando coerenza tra gli ambienti di sviluppo e facilitando la riproducibilità delle build.
Le attività di testing sono supportate da **JUnit**, utilizzato per la definizione ed esecuzione dei test unitari e di integrazione, e da **Mockito**, impiegato per la creazione di oggetti simulati (*mock*) al fine di isolare i componenti sotto test e verificare il comportamento delle classi in **pre**senza di dipendenze non ancora disponibili o esterne al sistema.
Per quanto riguarda la sicurezza dei dati sensibili, in particolare delle credenziali degli utenti, il sistema utilizza **bcrypt** come algoritmo di hashing per la crittografia delle password. L'adozione di bcrypt consente di proteggere le informazioni sensibili rispettando le buone pratiche di sicurezza nella gestione dell'autenticazione.
La persistenza dei dati è realizzata mediante **JDBC (Java Database Connectivity)**, che fornisce un'interfaccia standard per l'accesso a database relazionali. L'uso di JDBC permette di separare la logica applicativa dalla gestione dei dati persistenti, garantendo portabilità e controllo diretto sulle operazioni di accesso e manipolazione dei dati.

I solution Object più rilevanti con cui si è scelto di implementare una soluzione al problema **post**o in fase di analisi sono:

- Architettura RMI: La scelta di RMI consente di realizzare un **sistema distribuito in un contesto chiuso e controllato**, nel quale client e server sono entrambi implementati in Java. Grazie alla **trasparenza di accesso**, il client può invocare servizi remoti come se fossero locali, senza doversi occupare dei dettagli di comunicazione di rete. Inoltre, l'utilizzo di Java e della JVM garantisce la **portabilità del sistema su diversi ambienti desktop**, permettendo l'esecuzione dell'applicazione su differenti sistemi operativi senza modifiche al codice.
- **DatabaseManager / ConnectionManager**, responsabile della creazione e gestione delle connessioni JDBC

Interface Specification

CLASS UTENTE CREDENZIALI

- INVARIANTE context UtenteCredenziali
  inv: self.username.size() <= 9 and not self.username.matches('[0-9]+')

context UtenteCredenziali:: CheckCredenziali(Username,HashedPassword)
pre Username <> null and Username <> "" and HashedPassword <> null and HashedPassword <> ""

context UtenteCredenziali::CheckCredenziali(Username: String, HashedPassword: String) : Boolean
post: result = (Utente.allInstances()->exists(u | u.Username = Username and u.HashedPassword = HashedPassword))

INTERFACCIA PERSISTENCE

context <T>::doSave(entity: T) : void **pre**: entity <> null

context <T>::doSave(entity: T) : void **post**: T.allInstances()->includes(entity)

context <T>::doDelete(key: String) : Boolean **pre**: key <> null and key <> ""

context <T>::doDelete(key: String) : Boolean
post: result = (T.allInstances()->exists(e | e.key = key) implies
not T.allInstances()->exists(e | e.key = key))

context <T>::doRetrieveByKey(key: String) : T **pre**: key <> null and key <> ""

context <T>::doRetrieveByKey(key: String) : T **post**: (result <> null implies result.key = key) and (result = null implies T.allInstances()->forAll(e | e.key <> key))

context <T>::doRetrieveAll() : ArrayList(T)**pre**: true

context <T>::doRetrieveAll() : ArrayList(T) **post**: result->asSet() = T.allInstances()->asSet()

CLASSE GOVERNANTE

context GovernanteImpl::setStatoInPulizia(c: Camera) : void
pre: c <> null

context GovernanteImpl::setStatoInPulizia(c: Camera) : void
post: c.stato = StatoCamera::InPulizia

CLASSE FRONTDESK

context FrontDeskImpl::registraPrenotazione(dataPrenotazione: LocalDate,tipoDocumento: Enum,noteAggiuntive: String,
Intestatario: String,dataScadenzaDoc: LocalDate,dataRilascioDoc: LocalDate,rm: Camera,c: Cliente) : Prenotazione
pre:
dataPrenotazione <> null and
tipoDocumento <> null and
Intestatario <> null and Intestatario <> "" and
dataRilascioDoc <> null and
dataScadenzaDoc <> null and
rm <> null and
c <> null

context FrontDeskImpl::registraPrenotazione(...) : Prenotazione
post:
result <> null and
Prenotazione.allInstances()->includes(result) and
result.cliente = c and
result.camera = rm

context FrontDeskImpl::banBlackList(c: Cliente) : Boolean

pre: c <> null

context FrontDeskImpl::banBlackList(c: Cliente) : Boolean
post: result = (Blacklist.allInstances()->includes(c))

context FrontDeskImpl::unBanBlackList(c: Cliente) : Boolean
pre: c <> null

context FrontDeskImpl::unBanBlackList(c: Cliente) : Boolean
post: result = (not Blacklist.allInstances()->includes(c))

context FrontDeskImpl::aggiungiServizio(s: Servizio, p: Prenotazione) : Boolean
pre: s <> null and p <> null

context FrontDeskImpl::aggiungiServizio(s: Servizio, p: Prenotazione) : Boolean
post: result = (p.servizi->includes(s))

context FrontDeskImpl::rimuoviServizio(p: Prenotazione) : Boolean
pre: p <> null

context FrontDeskImpl::rimuoviServizio(p: Prenotazione) : Boolean
post: result = (p.servizi->isEmpty())

context FrontDeskImpl::aggiungiCliente(p: Prenotazione) : Boolean
pre: p <> null and p.cliente <> null

context FrontDeskImpl::aggiungiCliente(p: Prenotazione) : Boolean
post: result = (p.cliente <> null)

context FrontDeskImpl::rimuoviCliente(p: Prenotazione) : Boolean
pre: p <> null

context FrontDeskImpl::rimuoviCliente(p**:** Prenotazione) : Boolean
post: result = (p.cliente = null)

context FrontDeskImpl::eliminaPrenotazione(p: Prenotazione) : void
pre: p <> null

context FrontDeskImpl::eliminaPrenotazione(p: Prenotazione) : void
post: not Prenotazione.allInstances()->includes(p)

context FrontDeskImpl::ricercaConFiltro(
  nome: String, cognome: String, nazionalità: String, dataNascita: LocalDate, sesso: Sesso) : Collection(Prenotazione)
pre:
  nome <> null or cognome <> null or nazionalità <> null or dataNascita <> null or sesso <> null

context FrontDeskImpl::ricercaConFiltro(...) : Collection(Prenotazione)
post:
  result->forAll(p |
    (nome = null or p.cliente.nome = nome) and
    (cognome = null or p.cliente.cognome = cognome) and
    (nazionalità = null or p.cliente.nazionalità = nazionalità) and
    (dataNascita = null or p.cliente.dataNascita = dataNascita) and
    (sesso = null or p.cliente.sesso = sesso)
)

context FrontDeskImpl::checkout(p: Prenotazione, metodoPagamento: String) : Ricevuta
pre: p <> null and metodoPagamento <> null and metodoPagamento <> ""

context FrontDeskImpl::checkout(p: Prenotazione, metodoPagamento: String) : Ricevuta
post:
  result <> null and
  result.prenotazione = p and
  result.metodoPagamento = metodoPagamento

CLASSE MANAGER
context ManagerImpl::registraImpiegato(i: Impiegato) : void

pre: i <> null
pre: i <> null and Impiegato.allInstances()->includes(i)

context ManagerImpl::eliminaImpiegato(i: Impiegato) : void
post: not Impiegato.allInstances()->includes(i)

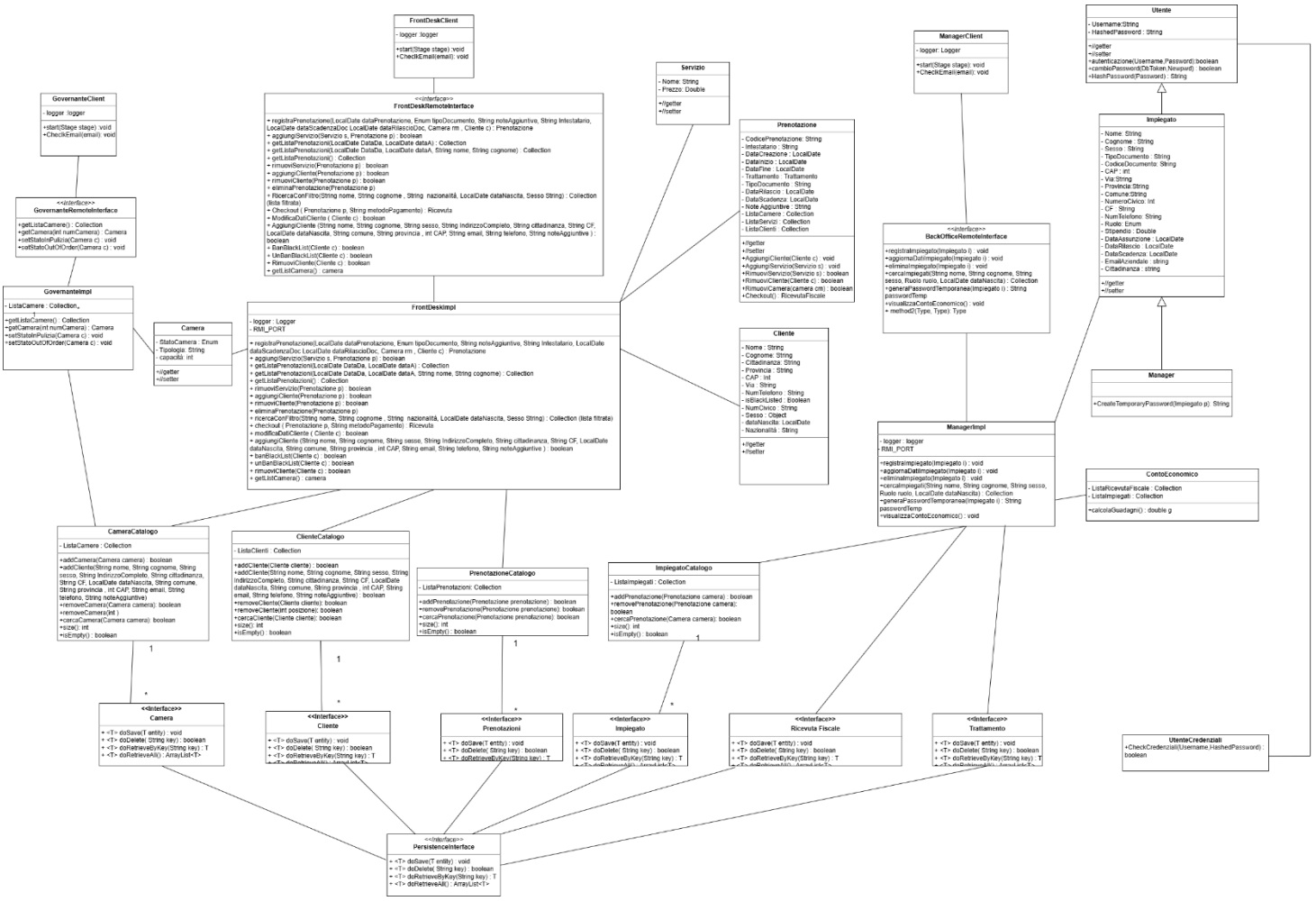
context ManagerImpl::cercaImpiegati(
  nome: String, cognome: String, sesso: String, ruolo: Ruolo, dataNascita: LocalDate) :
Collection(Impiegato)
pre: nome <> null or cognome <> null or sesso <> null or ruolo <> null or dataNascita <> null

context ManagerImpl::cercaImpiegati(...) : Collection(Impiegato)
post: result->forAll(e |
  (nome = null or e.nome = nome) and
  (cognome = null or e.cognome = cognome) and
  (sesso = null or e.sesso = sesso) and
  (ruolo = null or e.ruolo = ruolo) and
  (dataNascita = null or e.dataNascita = dataNascita)
)

context ManagerImpl::generaPasswordTemporanea(i: Impiegato) : String
pre: i <> null and Impiegato.allInstances()->includes(i)

context Manager Impl ::generaPasswordTemporanea(i: Impiegato) : String
post: result <> null and result.size() > 0

Evoluzione del class diagram


FrontDeskClient
- logger :logger
+start(Stage stage): void
+CheckEmail(email): void
GovernanteClient
- logger :logger
+start(Stage stage): void
+CheckEmail(email): void
<<interface>>
GovernanteRemoteInterface
+getListaCamere() : Collection
+getCamera(int numCamera) : Camera
+setStatoInPulizia(Camera c) : void
+setStatoOutOfOrder(Camera c) : void
GovernanteImpl
- ListaCamere : Collection
+getListaCamere() : Collection
+getCamera(int numCamera) : Camera
+setStatoInPulizia(Camera c) : void
+setStatoOutOfOrder(Camera c) : void
Camera
- StatoCamera : Enum
- Tipologia: String
- capacità : int
+/getter
+/setter
<<interface>>
FrontDeskRemoteInterface
+ registraPrenotazione(LocalDate dataPrenotazione, Enum tipoDocumento, String notaAggiuntive, String Intestatario, LocalDate dataScadenzaDoc LocalDate dataRilascioDoc, Camera rm , Cliente c) : Prenotazione
+ aggiungiServizio(Servizio s, Prenotazione p) : boolean
+ getListaPrenotazioni(LocalDate DataDa, LocalDate dataA) : Collection
+ getListaPrenotazioni(LocalDate DataDa, LocalDate dataA, String nome, String cognome) : Collection
+ getListaPrenotazioni() : Collection
+ rimuoviServizio(Prenotazione p) : boolean
+ aggiungiCliente(Prenotazione p) : boolean
+ rimuoviCliente(Prenotazione p) : boolean
+ eliminaPrenotazione(Prenotazione p)
+ RicercaConFiltro(String nome, String cognome , String nazionalità, LocalDate dataNascita, Sesso String) : Collection (lista filtrata)
+ Checkout (Prenotazione p, String metodoPagamento) : Ricevuta
+ ModificaDatiCliente (Cliente c) : boolean
+ AggiungiCliente (String nome, String cognome, String sesso, String indirizzoCompleto, String cittadinanza, String CF, LocalDate dataNascita, String comune, String provincia , int CAP, String email, String telefono, String noteAggiuntive) : boolean
+ BanBlackList(Cliente c) : boolean
+ UnBanBlackList(Cliente c) : boolean
+ RimuoviCliente(Cliente c) : boolean
+ getListCamera() : camera
FrontDeskImpl
- loggar : Logger
- RMI_PORT
+ registraPrenotazione(LocalDate dataPrenotazione, Enum tipoDocumento, String notaAggiuntive, String Intestatario, LocalDate dataScadenzaDoc LocalDate dataRilascioDoc, Camera rm , Cliente c) : Prenotazione
+ aggiungiServizio(Servizio s, Prenotazione p) : boolean
+ getListaPrenotazioni(LocalDate DataDa, LocalDate dataA) : Collection
+ getListaPrenotazioni(LocalDate DataDa, LocalDate dataA, String nome, String cognome) : Collection
+ getListaPrenotazioni() : Collection
+ rimuoviServizio(Prenotazione p) : boolean
+ aggiungiCliente(Prenotazione p) : boolean
+ rimuoviCliente(Prenotazione p) : boolean
+ eliminaPrenotazione(Prenotazione p)
+ ricercaConFiltro(String nome, String cognome , String nazionalità, LocalDate dataNascita, Sesso String) : Collection (lista filtrata)
+ checkout (Prenotazione p, String metodoPagamento) : Ricevuta
+ modificaDatiCliente (Cliente c) : boolean
+ aggiungiCliente (String nome, String cognome, String sesso, String indirizzoCompleto, String cittadinanza, String CF, LocalDate dataNascita, String comune, String provincia , int CAP, String email, String telefono, String noteAggiuntive) : boolean
+ banBlackList(Cliente c) : boolean
+ unBanBlackList(Cliente c) : boolean
+ rimuoviCliente(Cliente c) : boolean
+ getListCamera() : camera
Servizio
- Nome: String
- Prezzo: Double
+/getter
+/setter
Prenotazione
- CodicePrenotazione: String
- Intestatario : String
- DataCreazione : LocalDate
- DataInizio : LocalDate
- DataFine : LocalDate
- Trattamento : Trattamento
- TipoDocumento : String
- DataRilascio : LocalDate
- DataScadenza: LocalDate
- Note Aggiuntive : String
- ListaCamere : Collection
- ListaServizi : Collection
- ListaClienti : Collection
+/getter
+/setter
+AggiungiCliente(Cliente c) : void
+AggiungiServizio(Servizio s) : void
+RimuoviServizio(Servizio s) : boolean
+RimuoviCliente(Cliente c) : boolean
+RimuoviCamera(camera cm) : boolean
+Checkout() : RicevutaFiscale
Cliente
- Nome : String
- Cognome: String
- Cittadinanza: String
- Provincia : String
- CAP : Int
- Via : String
- NumTelefono : String
- isBlackListed : Boolean
- NumCivico : String
- Sesso : Object
- dataNascita: LocalDate
- Nazionalità : String
+/getter
+/setter
ManagerClient
- logger: Logger
+start(Stage stage): void
+CheckEmail(email): void
<<interface>>
BackOfficeRemoteInterface
+registraImpiegato(Impiegato i) : void
+aggiornaDatiImpiegato(Impiegato i) : void
+eliminaImpiegato(Impiegato i) : void
+cercaImpiegati(String nome, String cognome, String sesso, Ruolo ruolo, LocalDate dataNascita) : Collection
+generaPasswordTemporanea(impiegato i) : String passwordTemp
+visualizzaContoEconomico() : void
+ method2(Type, Type) Type
ManagerImpl
- loggar : logger
- RMI_PORT
+registraImpiegato(Impiegato i) : void
+aggiornaDatiImpiegato(Impiegato i) : void
+eliminaImpiegato(Impiegata i) : void
+cercaImpiegati(String nome, String cognome, String sesso, Ruolo ruolo, LocalDate dataNascita) : Collection
+generaPasswordTemporanea(impiegato i) : String passwordTemp
+visualizzaContoEconomico() : void
Utente
- Username:String
- HashedPassword : String
+/getter
+/setter
+autenticazione(Username,Password):boolean
+cambioPassword(DbToken,Newpwd) : boolean
+HashPassword(Password) : String
Impiegato
- Nome : String
- Cognome : String
- Sesso : String
- TipoDocumento : String
- CodiceDocumento: String
- CAP : int
- Via :String
- Provincia:String
- Comune:String
- NumeroCivico: Int
- CF : String
- NumTelefono: String
- Ruolo: Enum
- Stipendio : Double
- DataAssunzione : LocalDate
- DataRilascio : LocalDate
- DataScadenza: LocalDate
- EmailAziendale : string
- Cittadinanza : string
+/getter
+/setter
Manager
+CreateTemporaryPassword(Impiegato p): String
ContoEconomico
- ListaRicevutaFiscale : Collection
- ListaImpiegati : Collection
+calcolaGuadagni() : double g
UtenteCredenziali
+CheckCredenziali(Username,HashedPassword) : boolean
CameraCatalogo
- ListaCamere : Collection
+addCamera(Camera camera) : boolean
+addCliente(String nome, String cognome, String sesso, String IndirizzoCompleto, String cittadinanza, String CF, LocalDate dataNascita, String comune, String provincia , int CAP, String email, String telefono, String noteAggiuntive)
+removeCamera(Camera camera): boolean
+removeCamera(int i)
+cercaCamera(Camera camera): boolean
+size(): int
+isEmpty() : boolean
ClienteCatalogo
- ListaClienti : Collection
+addCliente(Cliente cliente) : boolean
+addCliente(String nome, String cognome, String sesso, String IndirizzoCompleto, String cittadinanza, String CF, LocalDate dataNascita, String comune, String provincia , int CAP, String email, String telefono, String noteAggiuntive) : boolean
+removeCliente(Cliente cliente): boolean
+removeCliente(int posizione): boolean
+cercaCliente(Cliente cliente): boolean
+size(): int
+isEmpty() : boolean
PrenotazioneCatalogo
- ListaPrenotazioni: Collection
+addPrenotazione(Prenotazione prenotazione) : boolean
+removePrenotazione(Prenotazione prenotazione): boolean
+cercaPrenotazione(Prenotazione prenotazione): boolean
+size(): int
+isEmpty() : boolean
ImpiegatoCatalogo
- ListaImpiegati : Collection
+addPrenotazione(Prenotazione camera) : boolean
+removePrenotazione(Prenotazione camera): boolean
+cercaPrenotazione(Camera camera): boolean
+size(): int
+isEmpty() : boolean
<<Interface>>
Camera
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>
<<Interface>>
Cliente
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
<<Interface>>
Prenotazioni
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
<<Interface>>
Impiegato
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
<<Interface>>
Ricevuta Fiscale
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
<<Interface>>
Trattamento
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
<<Interface>>
PersistenceInterface
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>