UNIVERSITÀ DEGLI STUDI
DI SALERNO

OBJECT DESIGN

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
06/02/2026	2.0	Refactor dei metodi	Giovanni Riccardi

Sommario

INTRODUCTION

Una volta completata la fase di system design, l'attenzione si sposta 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 precisa 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 precedenti.

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 presenza 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 posto 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

FRONTDESK

Gestione Prenotazioni: Visualizzazione
FrontDeskInterface
context FrontDeskInterface::getPrenotazioni() : Collection(Prenotazione)
pre: true
context FrontDeskInterface::getPrenotazioni() : Collection(Prenotazione)

post: result <> null

Gestione Prenotazioni: Operazioni CRUD
FrontDeskInterface
context FrontDeskInterface::addPrenotazione(p: Prenotazione) : void
pre: p <> null **context** FrontDeskInterface::addPrenotazione(p: Prenotazione) : void
post: Prenotazione.allInstances()->includes(p)
context FrontDeskInterface::removePrenotazione(p: Prenotazione) : void
pre: p <> null
context FrontDeskInterface::removePrenotazione(p: Prenotazione) : void
post: not Prenotazione.allInstances()->includes(p)
context FrontDeskInterface::updatePrenotazione(p: Prenotazione) : void
pre: p <> null
context FrontDeskInterface::updatePrenotazione(p: Prenotazione) : void
post: Prenotazione.allInstances()→includes(p)

Gestione Prenotazioni: Ricerca
FrontDeskInterface
context FrontDeskInterface::getPrenotazioneById(id: int) : Prenotazione
pre: id <> null
context FrontDeskInterface::getPrenotazioneById(id: int) : Prenotazione
post: result <> null

Gestione Clienti: Operazioni CRUD
FrontDeskInterface
context FrontDeskInterface::addCliente(c: Cliente) : void
pre: c <> null
context FrontDeskInterface::addCliente(c: Cliente) : void
post: Cliente.allInstances()->includes(c)
context FrontDeskInterface::removeCliente(c: Cliente) : void
pre: c <> null

context FrontDeskInterface::removeCliente(c: Cliente) : void
post: not Cliente.allInstances()->includes(c)
context FrontDeskInterface::updateCliente(c: Cliente) : void
pre: c <> null
context FrontDeskInterface::updateCliente(c: Cliente) : void
post: Cliente.allInstances()→includes(c)

Gestione Clienti: Ricerca
FrontDeskInterface
context FrontDeskInterface::getClienteByCf(cf: String) : Cliente
pre: cf <> null
context FrontDeskInterface::getClienteByCf(cf: String) : Cliente
post: result <> null

Gestione Clienti: Blacklist
FrontDeskInterface
context FrontDeskInterface::banCliente(c: Cliente) : void
pre: c <> null
context FrontDeskInterface::banCliente(c: Cliente) : void
post: c.isBlacklisted = true
context FrontDeskInterface::unBanCliente(c: Cliente) : void
pre: c <> null
context FrontDeskInterface::unBanCliente(c: Cliente) : void
post: c.isBlacklisted = false

Gestione Camere: Operazioni Stato
FrontDeskInterface
context FrontDeskInterface::getCamere() : Collection(Camera)
pre: true
context FrontDeskInterface::getCamere() : Collection(Camera)
post: result <> null
context FrontDeskInterface::aggiornaStatoCamera(c: Camera) : boolean

pre: c <> null
context FrontDeskInterface::aggiornaStatoCamera(c: Camera) : boolean
post: result = true or result = false

Autenticazione
FrontDeskInterface
context FrontDeskInterface::authentication(username: String, password: String, pwd2: String) : Impiegato
pre: username <> null and password <> null and pwd2 <> null
context FrontDeskInterface::authentication(username: String, password: String, pwd2: String) : Impiegato
post: result <> null

Operazioni di Sistema: Undo e Redo
FrontDeskInterface
context FrontDeskInterface::undoCommand() : void
pre: true
context FrontDeskInterface::undoCommand() : void
post: true
context FrontDeskInterface::redoCommand() : void
pre: true
context FrontDeskInterface::redoCommand() : void
post: true
GOVERNANTE

Gestione Stato Camere: Visualizzazione
GovernanteInterface
context GovernanteInterface::getListCamere() : Collection(Camera)
pre: true
context GovernanteInterface::getListCamere() : Collection(Camera)
post: result <> null

Gestione Stato Camere: Modifica Stato
GovernanteInterface
context GovernanteInterface::aggiornaStatoCamera(c: Camera) : boolean
pre: c <> null
context GovernanteInterface::aggiornaStatoCamera(c: Camera) : boolean
post: result = true or result = false

MANAGER

Gestione Impiegati: Operazioni CRUD

ManagerInterface
context ManagerInterface::addImpiegato(i: Impiegato) : void
pre: i <> null
context ManagerInterface::addImpiegato(i: Impiegato) : void
post: Impiegato.allInstances()->includes(i)
context ManagerInterface::removeImpiegato(i: Impiegato) : void
pre: i <> null
context ManagerInterface::removeImpiegato(i: Impiegato) : void
post: not Impiegato.allInstances()->includes(i)
context ManagerInterface::updateImpiegato(i: Impiegato) : void
pre: i <> null
context ManagerInterface::updateImpiegato(i: Impiegato) : void
post: Impiegato.allInstances()→includes(i)

Gestione Impiegati: Ricerca e Filtri
ManagerInterface
context ManagerInterface::getImpiegatoByCF(Cf: String) : Impiegato
pre: Cf <> null
context ManagerInterface::getImpiegatoByCF(Cf: String) : Impiegato

post: result <> null
context ManagerInterface::filtroImpiegati(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(Impiegato)
pre: nome <> null or sesso <> null or ruolo <> null or orderBy <> null
context ManagerInterface::filtroImpiegati(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(Impiegato)
post: result->forAll(i | (nome = null or i.nome = nome) and (sesso = null or i.sesso = sesso) and (ruolo = null or i.ruolo = ruolo))

Generazione Password
ManagerInterface
context ManagerInterface::generatePassword() : String
pre: true
context ManagerInterface::generatePassword() : String
post: result <> null and result <> ""

Gestione Conto Economico
ManagerInterface
context ManagerInterface::calcolaContoHotel() : Map
pre: true
context ManagerInterface::calcolaContoHotel() : Map
post: result <> null and result->includes("prenotazioni") and result->includes("camere") and result->includes("servizi") and result->includes("trattamenti") and result→includes("passivita")

Operazioni di Sistema: Undo e Redo
ManagerInterface
context ManagerInterface::undoCommand() : void
pre: true **context** ManagerInterface::undoCommand() : void
post: true

context ManagerInterface::redoCommand() : void
pre: true
context ManagerInterface::redoCommand() : void
post: true

INVARIANTI DI CLASSE

FrontDeskInterface
context FrontDeskInterface
inv: self.oclIsKindOf(ObserverCamereInterface)
context FrontDeskInterface
inv: self.getPrenotazioni()->forAll(p | p <> null)
context FrontDeskInterface
inv: self.getCamere()->forAll(c | c <> null)

GovernanteInterface
context GovernanteInterface
inv: self.oclIsKindOf(ObserverCamereInterface)
context GovernanteInterface
inv: self.getListCamere()->forAll(c | c <> null)

ManagerInterface
context ManagerInterface
inv: self.filtroImpiegati(null, null, null, null)->forAll(i | i <> null)
context ManagerInterface
inv: self.calcolaContoHotel()->keys()->size() = 5
context ManagerInterface
inv: self.calcolaContoHotel()->keys()->includes("prenotazioni") and
self.calcolaContoHotel()->keys()->includes("camere") and
self.calcolaContoHotel()->keys()->includes("servizi") and

self.calcolaContoHotel()->keys()->includes("trattamenti") and self.calcolaContoHotel()->keys()->includes("passivita")

Cataloghi

CatalogoClienti

Gestione Clienti: Operazioni CRUD

CatalogoClienti
context CatalogoClienti::aggiungiCliente(cliente: Cliente) : boolean
pre: cliente <> null and listaClienti <> null and frontDeskStorage <> null
context CatalogoClienti::aggiungiCliente(cliente: Cliente) : boolean
post: result = true implies not listaClienti@pre->includes(cliente) and listaClienti->includes(cliente)
context CatalogoClienti::removeCliente(cliente: Cliente) : boolean
pre: cliente <> null and listaClienti <> null and frontDeskStorage <> null
context CatalogoClienti::removeCliente(cliente: Cliente) : boolean
post: result = true implies listaClienti@pre->includes(cliente) and not listaClienti->includes(cliente) and not listaClientiBannati->includes(cliente)
context CatalogoClienti::updateCliente(cliente: Cliente) : boolean
pre: cliente <> null and listaClienti <> null and frontDeskStorage <> null
context CatalogoClienti::updateCliente(cliente: Cliente) : boolean
post: result = true implies listaClienti->exists(c | c.cf = cliente.cf) and (cliente.isBlacklisted = true implies listaClientiBannati→includes(cliente))

Gestione Clienti: Ricerca e Filtri
CatalogoClienti

context CatalogoClienti::cercaClienti(nome: String, cognome: String, nazionalita: String, dataNascita: LocalDate, sesso: String) : Collection(Cliente)
pre: nome <> null or cognome <> null or nazionalita <> null or dataNascita <> null or sesso <> null
context CatalogoClienti::cercaClienti(nome: String, cognome: String, nazionalita: String, dataNascita: LocalDate, sesso: String) : Collection(Cliente)
post: result->forAll(c | (nome = null or c.nome = nome) and (cognome = null or c.cognome = cognome) and (nazionalita = null or c.nazionalità = nazionalita) and (dataNascita = null or c.dataNascita.isBefore(dataNascita)) and (sesso = null or c.sesso = sesso))
context CatalogoClienti::getCliente(CFCliente: String) : Cliente
pre: CFCliente <> null and CFCliente <> ""
context CatalogoClienti::getCliente(CFCliente: String) : Cliente
post: result = null or result.cf = CFCliente

Gestione Clienti: Validazione
CatalogoClienti
context CatalogoClienti::checkCliente(nome: String, cognome: String, nazionalita: String, dataNascita: LocalDate, blackListed: Boolean) : void
pre: nome <> null or cognome <> null or nazionalita <> null or dataNascita <> null or blackListed <> null
context CatalogoClienti::checkCliente(nome: String, cognome: String, nazionalita: String, dataNascita: LocalDate, blackListed: Boolean) : void
post: true

Gestione Clienti: Dati
CatalogoClienti
context CatalogoClienti::getListaClienti() : Collection(Cliente)
pre: true
context CatalogoClienti::getListaClienti() : Collection(Cliente)

post: result = listaClienti
context CatalogoClienti::setListaClienti(listaClienti1: Collection(Cliente)) : void
pre: listaClienti1 <> null
context CatalogoClienti::setListaClienti(listaClienti1: Collection(Cliente)) : void
post: listaClienti->forAll(c | listaClienti1->includes(c))
context CatalogoClienti::getListaClientiBannati() : Collection(Cliente)
pre: true
context CatalogoClienti::getListaClientiBannati() : Collection(Cliente)
post: result = listaClientiBannati
context CatalogoClienti::setListaClientiBannati(listaClientiBannati1: Collection(Cliente)) : void
pre: listaClientiBannati1 <> null
context CatalogoClienti::setListaClientiBannati(listaClientiBannati1: Collection(Cliente)) : void
post: listaClientiBannati = listaClientiBannati1

Gestione Clienti: Utility
CatalogoClienti
context CatalogoClienti::clienteIsEquals(c: Cliente) : boolean
pre: true
context CatalogoClienti::clienteIsEquals(c: Cliente) : boolean
post: result = false implies (c = null or not listaClienti->includes(c))
context CatalogoClienti::aggiornalista() : boolean
pre: true
context CatalogoClienti::aggiornalista() : boolean
post: result = true implies not listaClienti→isEmpty()

CatalogoImpiegati

Gestione Impiegati: Operazioni CRUD
CatalogoImpiegati
context CatalogoImpiegati::aggiungiImpiegato(imp: Impiegato) : boolean
pre: imp <> null and listaImpiegati <> null and backOfficeStorage <> null
context CatalogoImpiegati::aggiungiImpiegato(imp: Impiegato) : boolean
post: result = true implies not listaImpiegati@pre->includes(imp) and listaImpiegati->includes(imp) and listaImpiegati->size() = listaImpiegati@pre->size() + 1
context CatalogoImpiegati::eliminaImpiegato(imp: Impiegato) : boolean
pre: imp <> null and listaImpiegati <> null and backOfficeStorage <> null
context CatalogoImpiegati::eliminaImpiegato(imp: Impiegato) : boolean
post: result = true implies listaImpiegati@pre->includes(imp) and not listaImpiegati->includes(imp) and listaImpiegati->size() = listaImpiegati@pre->size() - 1
context CatalogoImpiegati::updateImpiegato(imp: Impiegato) : boolean
pre: imp <> null and listaImpiegati <> null and backOfficeStorage <> null
context CatalogoImpiegati::updateImpiegato(imp: Impiegato) : boolean
post: result = true implies listaImpiegati@pre->includes(imp) and listaImpiegati@pre->exists(i | i.codiceFiscale = imp.codiceFiscale and not i.equals(imp)) and listaImpiegati->includes(imp) and not listaImpiegati->exists(i | i.codiceFiscale = imp.codiceFiscale and not i.equals(imp)) and listaImpiegati->size() = listaImpiegati@pre→size()

Gestione Impiegati: Validazione
CatalogoImpiegati
context CatalogoImpiegati::checkImpiegato(impiegato: Impiegato) : void
pre: impiegato <> null
context CatalogoImpiegati::checkImpiegato(impiegato: Impiegato) : void
post: true

Gestione Impiegati: Dati
CatalogoImpiegati
context CatalogoImpiegati::getListaImpiegati() : Collection(Impiegato)
pre: true
context CatalogoImpiegati::getListaImpiegati() : Collection(Impiegato)
post: result = listaImpiegati
context CatalogoImpiegati::setListaImpiegati(listaImpiegati1: Collection(Impiegato)) : void
pre: listaImpiegati1 <> null
context CatalogoImpiegati::setListaImpiegati(listaImpiegati1: Collection(Impiegato)) : void
post: listaImpiegati->forAll(i | listaImpiegati1→includes(i))

Gestione Impiegati: Utility
CatalogoImpiegati
context CatalogoImpiegati::aggiornaLista() : boolean
pre: true
context CatalogoImpiegati::aggiornaLista() : boolean
post: result = true implies listaImpiegati <> null

CatalogoPrenotazioni

Gestione Prenotazioni: Operazioni CRUD
CatalogoPrenotazioni
context CatalogoPrenotazioni::addPrenotazioni(prenotazione: Prenotazione) : boolean
pre: listaPrenotazioni1 <> null
context CatalogoPrenotazioni::addPrenotazioni(prenotazione: Prenotazione) : boolean

post: result = true implies listaPrenotazioni->includes(prenotazione)
context CatalogoPrenotazioni::removePrenotazioni(prenotazione: Prenotazione) : boolean
pre: prenotazione = null or listaPrenotazioni <> null
context CatalogoPrenotazioni::removePrenotazioni(prenotazione: Prenotazione) : boolean
post: (prenotazione = null implies result = false) and (not listaPrenotazioni@pre->includes(prenotazione) implies result = false) and (result = true implies listaPrenotazioni@pre->includes(prenotazione) and not listaPrenotazioni->includes(prenotazione) and listaPrenotazioni->size() = listaPrenotazioni@pre->size() - 1)
context CatalogoPrenotazioni::UpdatePrenotazioni(prenotazione: Prenotazione) : boolean
pre: prenotazione = null or listaPrenotazioni <> null and fds <> null
context CatalogoPrenotazioni::UpdatePrenotazioni(prenotazione: Prenotazione) : boolean
post: (prenotazione = null implies result = false) and (not listaPrenotazioni@pre->includes(prenotazione) implies result = false) and (result = true implies listaPrenotazioni@pre->exists(p | p.IDPrenotazione = prenotazione.IDPrenotazione) and listaPrenotazioni->includes(prenotazione) and not listaPrenotazioni->exists(p | p.IDPrenotazione = prenotazione.IDPrenotazione and not p.equals(prenotazione)) and listaPrenotazioni->size() = listaPrenotazioni@pre→size())null

- **Gestione Prenotazioni: Validazione**

CatalogoPrenotazioni
context CatalogoPrenotazioni::checkPrenotazione(prenotazione: Prenotazione) : void
pre: prenotazione <> null

context CatalogoPrenotazioni::checkPrenotazione(prenotazione: Prenotazione) : void
post: true

- **Gestione Prenotazioni: Dati**

Prenotazione
context Prenotazione::getIDPrenotazione() : Integer
pre: true

Gestione Prenotazioni: Utility
CatalogoPrenotazioni
context CatalogoPrenotazioni::aggiornalista() : boolean
pre: listaPrenotazioni <> null
context CatalogoPrenotazioni::aggiornalista() : boolean
post: result = true implies fds <> null and listaPrenotazioni <> null

CatalogoCamere

Gestione Camere: Operazioni Stato
CatalogoCamere
context CatalogoCamere::aggiornaStatoCamera(c: Camera) : boolean
pre: c <> null and c.numeroCamera > 0
context CatalogoCamere::aggiornaStatoCamera(c: Camera) : boolean
post: result = true implies camereList->exists(cam | cam.numeroCamera = c.numeroCamera and cam.statoCamera = c.statoCamera) and lastModified = c

Gestione Camere: Visualizzazione
CatalogoCamere
context CatalogoCamere::getListaCamere() : Collection(Camera)
pre: true

context CatalogoCamere::getListaCamere() : Collection(Camera)
post: result <> null
context CatalogoCamere::getCamera(numeroCamera: int) : Camera
pre: numeroCamera > 0
context CatalogoCamere::getCamera(numeroCamera: int) : Camera
post: (result <> null implies camereList->exists(camera | camera.numeroCamera = numeroCamera)) and (result <> null implies result <> camereList->any(camera | camera.numeroCamera = numeroCamera))
context CatalogoCamere::getLastModified() : Camera
pre: true
context CatalogoCamere::getLastModified() : Camera
post: result = lastModified

Gestione Camere: Operazioni CRUD
CatalogoCamere
context CatalogoCamere::addCamere(camere: Collection(Camera)) : void
pre: camere <> null and camere->size() > 0
context CatalogoCamere::addCamere(camere: Collection(Camera)) : void
post: camere->forAll(camera | camereList->exists(c | c.numeroCamera = camera.numeroCamera))

Gestione Camere: Observer Pattern
CatalogoCamere
context CatalogoCamere::attach(observer: ObserverCamereInterface) : void
pre: observer <> null
context CatalogoCamere::attach(observer:

ObserverCamereInterface) : void
post: observers->includes(observer)
context CatalogoCamere::detach(observer: ObserverCamereInterface) : void
pre: observer <> null
context CatalogoCamere::detach(observer: ObserverCamereInterface) : void
post: not observers->includes(observer)
context CatalogoCamere::notifyObservers() : void
pre: true
context CatalogoCamere::notifyObservers() : void
post: observers->forAll(observer | observer.update() invoked)

Gestione Camere: Utility
CatalogoCamere
context CatalogoCamere::cameraIsEquals(c: Cliente) : boolean
pre: true
context CatalogoCamere::cameraIsEquals(c: Cliente) : boolean
post: result = false implies (c = null or not camereList→includes(c))

CatalogueUtils

Utilità: Validazione
CatalogueUtils
context CatalogueUtils::checkNull(oggetto: T) : void
pre: oggetto <> null
context CatalogueUtils::checkNull(oggetto: T) : void
post: true

INVARIANTI DI CLASSE

CatalogoClienti
context CatalogoClienti
inv: listaClienti <> null
context CatalogoClienti
inv: listaClientiBannati <> null
context CatalogoClienti
inv: frontDeskStorage <> null
context CatalogoClienti
inv: listaClientiBannati->forAll(c | c.isBlacklisted = true)
context CatalogoClienti
inv: listaClienti->forAll(c | c <> null)
contextCatalogoClienti
inv: listaClientiBannati->forAll(c | listaClienti->includes(c))

context CatalogoClienti
inv: listaClienti->forAll(c1, c2 | c1 <> c2 implies c1.cf <> c2.cf)

CatalogoImpiegati

contextCatalogoImpiegati
inv: listaImpiegati <> null

context CatalogoImpiegati
inv: backOfficeStorage <> null

context CatalogoImpiegati
inv: listaImpiegati->forAll(i | i <> null)
context CatalogoImpiegati
inv: listaImpiegati->forAll(i1, i2 | i1 <> i2 implies i1.codiceFiscale <>

i2.codiceFiscale)
context CatalogoImpiegati
inv: listaImpiegati->forAll(i | i.stipendio > 0)
context CatalogoImpiegati**inv:** listaImpiegati->forAll(i | i.ruolo <> null)

CatalogoPrenotazioni
context CatalogoPrenotazioni
inv: listaPrenotazioni <> null
context CatalogoPrenotazioni
inv: fds <> null
context CatalogoPrenotazioni
inv: listaPrenotazioni->forAll(p | p <> null)

context CatalogoPrenotazioni
inv: listaPrenotazioni->forAll(p1, p2 | p1 <> p2 implies p1.IDPrenotazione <> p2.IDPrenotazione)
context CatalogoPrenotazioni **inv:** listaPrenotazioni->forAll(p | p.dataInizio.isBefore(p.dataFine) or p.dataInizio.isEqual(p.dataFine) = false)
context CatalogoPrenotazioni
inv: listaPrenotazioni->forAll(p | p.listaClienti->size() > 0)

CatalogoCamere
Context CatalogoCamere **inv:** camereList <> null
context CatalogoCamere **inv:** fds <> null
context CatalogoCamere **inv:** observers <> null
context CatalogoCamere **inv:** camereList->forAll(c | c <> null)
context CatalogoCamere**inv:** camereList->forAll(c1, c2 | c1 <> c2 implies c1.numeroCamera <> c2.numeroCamera)

context CatalogoCamere **inv:** camereList->forAll(c | c.numeroCamera > 0)

context CatalogoCamere **inv:** camereList->forAll(c | c.numeroMaxOccupanti > 0)
context CatalogoCamere **inv:** camereList->forAll(c | c.prezzoCamera >= 0)
context CatalogoCamere **inv:** self.oclIsKindOf(SubjectCamereInterface)

CatalogueUtils
context CatalogueUtils **inv:** true
post: result->forAll(c | listaCamere->includes(c))

Comandi Prenotazione

GESTIONE PRENOTAZIONI – COMANDI

AddPrenotazioneCommand

AddPrenotazioneCommand
context AddPrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
pre: true
context AddPrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
post: result = catalogue
context AddPrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
pre: catalogue <> null
context AddPrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
post: self.catalogue = catalogue
context AddPrenotazioneCommand::getPrenotazione() : Prenotazione
pre: true
context AddPrenotazioneCommand::getPrenotazione() : Prenotazione
post: result = prenotazione
context AddPrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void

pre: prenotazione <> null
context AddPrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void
post: self.prenotazione = prenotazione
context AddPrenotazioneCommand::execute() : void
pre: true
context AddPrenotazioneCommand::execute() : void
post: CatalogoPrenotazioni.listaPrenotazioni->includes(prenotazione)
context AddPrenotazioneCommand::undo() : void
pre: true

context AddPrenotazioneCommand::undo() : void
post: not CatalogoPrenotazioni.listaPrenotazioni→includes(prenotazione)

RemovePrenotazioneCommand
RemovePrenotazioneCommand
context RemovePrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
pre: true
context RemovePrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
post: result = catalogue
context RemovePrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
pre: catalogue <> null
context RemovePrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
post: self.catalogue = catalogue
context RemovePrenotazioneCommand::getPrenotazioni() : Prenotazione
pre: true
context RemovePrenotazioneCommand::getPrenotazioni() : Prenotazione
post: result = prenotazione
context RemovePrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void

pre: prenotazione <> null
context RemovePrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void
post: self.prenotazione = prenotazione
context RemovePrenotazioneCommand::execute() : void
pre: true
context RemovePrenotazioneCommand::execute() : void
post: not CatalogoPrenotazioni.listaPrenotazioni->includes(prenotazione)
context RemovePrenotazioneCommand::undo() : void
pre: true
context RemovePrenotazioneCommand::undo() : void
post: CatalogoPrenotazioni.listaPrenotazioni→includes(prenotazione)

UpdatePrenotazioneCommand
UpdatePrenotazioneCommand
context UpdatePrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
pre: true
context UpdatePrenotazioneCommand::getCatalogue() : CatalogoPrenotazioni
post: result = catalogue
context UpdatePrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
pre: catalogue <> null
context UpdatePrenotazioneCommand::setCatalogue(catalogue: CatalogoPrenotazioni) : void
post: self.catalogue = catalogue
context UpdatePrenotazioneCommand::getPrenotazione() : Prenotazione
pre: true
context UpdatePrenotazioneCommand::getPrenotazione() : Prenotazione
post: result = prenotazione
context UpdatePrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void
pre: prenotazione <> null

context UpdatePrenotazioneCommand::setPrenotazione(prenotazione: Prenotazione) : void
post: self.prenotazione = prenotazione
context UpdatePrenotazioneCommand::execute() : void
pre: true
context UpdatePrenotazioneCommand::execute() : void
post: CatalogoPrenotazioni.listaPrenotazioni->exists(p | p.IDPrenotazione = prenotazione.IDPrenotazione)
context UpdatePrenotazioneCommand::undo() : void
pre: true
context UpdatePrenotazioneCommand::undo() : void
post: CatalogoPrenotazioni.listaPrenotazioni->exists(p | p.IDPrenotazione = prenotazioneNonModificata.IDPrenotazione)

INVARIANTI DI CLASSE

AddPrenotazioneCommand
context AddPrenotazioneCommand
inv: catalogue <> null
context AddPrenotazioneCommand
inv: prenotazione <> null
context AddPrenotazioneCommand
inv: self.oclIsKindOf(Command)

RemovePrenotazioneCommand
context RemovePrenotazioneCommand
inv: catalogue <> null
context RemovePrenotazioneCommand
inv: prenotazione <> null
context RemovePrenotazioneCommand
inv: self.oclIsKindOf(Command)

UpdatePrenotazioneCommand
context UpdatePrenotazioneCommand
inv: catalogue <> null
context UpdatePrenotazioneCommand
inv: prenotazione <> null
context UpdatePrenotazioneCommand
inv: self.oclIsKindOf(Command)
context UpdatePrenotazioneCommand
inv: prenotazioneNonModificata <> null implies prenotazioneNonModificata.IDPrenotazione = prenotazione.IDPrenotazione

Comandi Impiegato

GESTIONE IMPIEGATI – COMANDI

AddImpiegatoCommand

AddImpiegatoCommand
context AddImpiegatoCommand::getCatalogue() : CatalogoImpiegati
pre: true
context AddImpiegatoCommand::getCatalogue() : CatalogoImpiegati
post: result = catalogue
context AddImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
pre: catalogue <> null
context AddImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
post: self.catalogue = catalogue
context AddImpiegatoCommand::getImpiegato() : Impiegato
pre: true
context AddImpiegatoCommand::getImpiegato() : Impiegato
post: result = impiegato
context AddImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
pre: impiegato <> null
context AddImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
post: self.impiegato = impiegato
context AddImpiegatoCommand::execute() : void
pre: true
context AddImpiegatoCommand::execute() : void
post: CatalogoImpiegati.listaImpiegati->includes(impiegato)
context AddImpiegatoCommand::undo() : void
pre: true
context AddImpiegatoCommand::undo() : void
post: not CatalogoImpiegati.listaImpiegati→includes(impiegato)

RemoveImpiegatoCommand

RemoveImpiegatoCommand
context RemoveImpiegatoCommand::getCatalogue() : CatalogoImpiegati

pre: true
context RemoveImpiegatoCommand::getCatalogue() : CatalogoImpiegati
post: result = catalogue
context RemoveImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
pre: catalogue <> null
context RemoveImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
post: self.catalogue = catalogue
context RemoveImpiegatoCommand::getImpiegato() : Impiegato
pre: true
context RemoveImpiegatoCommand::getImpiegato() : Impiegato
post: result = impiegato
context RemoveImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
pre: impiegato <> null
context RemoveImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
post: self.impiegato = impiegato
context RemoveImpiegatoCommand::execute() : void
pre: true
context RemoveImpiegatoCommand::execute() : void
post: not CatalogoImpiegati.listaImpiegati->includes(impiegato)
context RemoveImpiegatoCommand::undo() : void
pre: true
context RemoveImpiegatoCommand::undo() : void
post: CatalogoImpiegati.listaImpiegati→includes(impiegato)

UpdateImpiegatoCommand

UpdateImpiegatoCommand
context UpdateImpiegatoCommand::getCatalogue() : CatalogoImpiegati
pre: true
context UpdateImpiegatoCommand::getCatalogue() : CatalogoImpiegati

post: result = catalogue
context UpdateImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
pre: catalogue <> null
context UpdateImpiegatoCommand::setCatalogue(catalogue: CatalogoImpiegati) : void
post: self.catalogue = catalogue
context UpdateImpiegatoCommand::getImpiegato() : Impiegato
pre: true
context UpdateImpiegatoCommand::getImpiegato() : Impiegato
post: result = impiegato
context UpdateImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
pre: impiegato <> null
context UpdateImpiegatoCommand::setImpiegato(impiegato: Impiegato) : void
post: self.impiegato = impiegato
context UpdateImpiegatoCommand::execute() : void
pre: true
context UpdateImpiegatoCommand::execute() : void
post: CatalogoImpiegati.listaImpiegati->exists(i | i.codiceFiscale = impiegato.codiceFiscale)
context UpdateImpiegatoCommand::undo() : void
pre: true
context UpdateImpiegatoCommand::undo() : void
post: CatalogoImpiegati.listaImpiegati->exists(i | i.codiceFiscale = impiegatoNonModificato.codiceFiscale)

INVARIANTI DI CLASSE

AddImpiegatoCommand
context AddImpiegatoCommand
inv: catalogue <> null
context AddImpiegatoCommand

inv: impiegato <> null
context AddImpiegatoCommand
inv: self.oclIsKindOf(Command)

RemoveImpiegatoCommand
context RemoveImpiegatoCommand
inv: catalogue <> null
context RemoveImpiegatoCommand
inv: impiegato <> null
context RemoveImpiegatoCommand
inv: self.oclIsKindOf(Command)

UpdateImpiegatoCommand
context UpdateImpiegatoCommand
inv: catalogue <> null
context UpdateImpiegatoCommand
inv: impiegato <> null
context UpdateImpiegatoCommand
inv: self.oclIsKindOf(Command)
context UpdateImpiegatoCommand
inv: impiegatoNonModificato <> null implies impiegatoNonModificato.codiceFiscale = impiegato.codiceFiscale

Pattern Command

PATTERN E UTILITÀ DI SISTEMA

Command Pattern - Interfaccia Base

Command
context Command::execute() : void
pre: true
context Command::execute() : void
post: true
context Command::undo() : void
pre: true
context Command::undo() : void
post: true

Command Pattern - Invoker
Invoker
context Invoker::executeCommand(c: Command) : void
pre: c <> null
context Invoker::executeCommand(c: Command) : void
post: undoStack->includes(c) and redoStack->isEmpty()
context Invoker::undoCommand() : void
pre: not undoStack→isEmpty()
context Invoker::undoCommand() : void
post: redoStack->size() = redoStack@pre->size() + 1
context Invoker::redo() : void

pre: not redoStack→isEmpty()
context Invoker::redo() : void
post: undoStack->size() = undoStack@pre->size() + 1

INVARIANTI DI CLASSE

Command
context Command
inv: true

Invoker
context Invoker
inv: undoStack <> null
context Invoker
inv: redoStack <> null
context Invoker
inv: undoStack->forAll(c | c <> null)
context Invoker
inv: redoStack->forAll(c | c <> null)
context Invoker
inv: undoStack->forAll(c | c.oclIsKindOf(Command))
context Invoker
inv: redoStack->forAll(c | c.oclIsKindOf(Command))

Autenticazione

COMUNE A TUTTI I RUOLI

Autenticazione: Gestione Credenziali

CredenzialiUtils
context CredenzialiUtils::HashPassword(password: String) : String
pre: password <> null and password <> ""
context CredenzialiUtils::HashPassword password: String) : String
post: result <> null and result <> "" and result <> password
context CredenzialiUtils::checkPassword(password: String, HashedPassword: String) : boolean
pre: password <> null and password <> "" and HashedPassword <> null and HashedPassword <> ""
context CredenzialiUtils::checkPassword(password: String, HashedPassword: String) : boolean
post: result = true or result = false

Autenticazione: Generazione Token

TokenGenerator
context TokenGenerator::generateToken() : String
pre: true
context TokenGenerator::generateToken() : String
post: result <> null and result <> "" and result.startsWith("PWD-TMP-")
context TokenGenerator::isExpired(expiresAt: Instant) : boolean
pre: expiresAt <> null
context TokenGenerator::isExpired(expiresAt: Instant) : boolean
post: result <> null
context TokenGenerator::getToken() : String
pre: true

context TokenGenerator::getToken() : String
post: result <> null and result <> ""
context TokenGenerator::getExpiresAt() : Instant
pre: true
context TokenGenerator::getExpiresAt() : Instant
post: result <> null

Autenticazione: Verifica Account

Autentication
context Autentication::checkaccount(username: String, password: String, pwd2: String) : boolean
pre: username <> null and password <> null
context Autentication::checkaccount(username: String, password: String, pwd2: String) : boolean
post: result = true or result = false
context Autentication::getImpiegato() : Impiegato
pre: true
context Autentication::getImpiegato() : Impiegato
post: result <> null implies result <> impiegato

INVARIANTI DI CLASSE

CredenzialiUtils
context CredenzialiUtils
inv: true

TokenGenerator
context TokenGenerator
inv: token <> null and token <> ""
context TokenGenerator
inv: expiresAt <> null

context TokenGenerator
inv: token.startsWith("PWD-TMP-")
context TokenGenerator
inv: token.size() > 8

Autentication
context Autentication
inv: password <> null
context Autentication
inv: userName <> null
context Autentication
inv: userName.matches("^(Reception|Manager|Governante)\d+$") implies userName.size() > 9
context Autentication
inv: impiegato <> null implies impiegato.username = userName

Observer Pattern e Utility

PATTERN E UTILITÀ DI SISTEMA

Observer Pattern – Camere

ObserverCamereInterface
context ObserverCamereInterface::update() : Camera
pre: true
context ObserverCamereInterface::update() : Camera
post: result <> null

Subject Pattern – Camere

SubjectCamereInterface
context SubjectCamereInterface::attach(observer: ObserverCamereInterface) : void
pre: observer <> null
context SubjectCamereInterface::attach(observer: ObserverCamereInterface) : void
post: observers->includes(observer)
context SubjectCamereInterface::detach(observer: ObserverCamereInterface) : void
pre: observer <> null
context SubjectCamereInterface::detach(observer: ObserverCamereInterface) : void
post: not observers->includes(observer)
context SubjectCamereInterface::notifyObservers() : void

pre: true
context SubjectCamereInterface::notifyObservers() : void
post: true

INVARIANTI DI CLASSE

ObserverCamereInterface
context ObserverCamereInterface
inv: self.oclIsKindOf(Remote)

SubjectCamereInterface
context SubjectCamereInterface
inv: observers <> null
context SubjectCamereInterface
inv: observers->forAll(o | o <> null)
context SubjectCamereInterface
inv: observers->forAll(o | o.oclIsKindOf(ObserverCamereInterface))

Gestione Conto Economico

MANAGER

Gestione Conto Economico

ManagerInterface
context ManagerInterface::calcolaContoHotel() : Map
pre: true
context ManagerInterface::calcolaContoHotel() : Map
post: result <> null and result->includes("prenotazioni") and result->includes("camere") and result->includes("servizi") and result->includes("trattamenti") and result→includes("passivita")

Generazione Password
ManagerInterface
context ManagerInterface::generatePassword() : String
pre: true
context ManagerInterface::generatePassword() : String
post: result <> null and result <> ""

PATTERN COMPOSITE - CONTO ECONOMICO

ContoEconomicoComponentAbstract

ContoEconomicoComponentAbstract
context ContoEconomicoComponentAbstract::getNomeComponente() : String
pre: true
context ContoEconomicoComponentAbstract::getNomeComponente() : String
post: result = nomevoce
context ContoEconomicoComponentAbstract::getPrezzo() : double
pre: true
context ContoEconomicoComponentAbstract::getPrezzo() : double
post: result = prezzo
context ContoEconomicoComponentAbstract::setPrezzo(prezzo: double) : void
pre: prezzo <> null
context ContoEconomicoComponentAbstract::setPrezzo(prezzo: double) : void
post: self.prezzo = prezzo
context ContoEconomicoComponentAbstract::addChild(child: ContoEconomicoComponentAbstract) : void
pre: child <> null
context ContoEconomicoComponentAbstract::addChild(child: ContoEconomicoComponentAbstract) : void
post: figli->includes(child)
context ContoEconomicoComponentAbstract::removeChild(child: ContoEconomicoComponentAbstract) : void
pre: child <> null
context ContoEconomicoComponentAbstract::removeChild(child: ContoEconomicoComponentAbstract) : void
post: not figli→includes(child)

ContoEconomicoLeaf

ContoEconomicoLeaf
context ContoEconomicoLeaf::getImportoTotale() : double
pre: true
context ContoEconomicoLeaf::getImportoTotale() : double
post: result = importo
context ContoEconomicoLeaf::getTotalePerTipo(t: TipoVoce) : double
pre: t <> null
context ContoEconomicoLeaf::getTotalePerTipo(t: TipoVoce) : double
post: result = (tipo = t implies result = importo) and (tipo <> t implies result = 0)
context ContoEconomicoLeaf::stampaAlbero(indent: String, ultimo: boolean) : void
pre: indent <> null
context ContoEconomicoLeaf::stampaAlbero(indent: String, ultimo: boolean) : void
post: true

ContoEconomicoComposite

ContoEconomicoComposite
context ContoEconomicoComposite::addChild(child: ContoEconomicoComponentAbstract) : void
pre: child <> null
context ContoEconomicoComposite::addChild(child: ContoEconomicoComponentAbstract) : void
post: figli->includes(child)
context ContoEconomicoComposite::removeChild(child: ContoEconomicoComponentAbstract) : void
pre: child <> null
context ContoEconomicoComposite::removeChild(child: ContoEconomicoComponentAbstract) : void
post: not figli->includes(child)
context ContoEconomicoComposite::getImportoTotale() : double
pre: true

context ContoEconomicoComposite::getImportoTotale() : double
post: result = figli->iterate(c; acc: double = 0 | acc + c.getImportoTotale())
context ContoEconomicoComposite::getFigli() : Collection(ContoEconomicoComponentAbstract)
pre: true
context ContoEconomicoComposite::getFigli() : Collection(ContoEconomicoComponentAbstract)
post: result = figli
context ContoEconomicoComposite::stampaAlbero(indent: String, ultimo: boolean) : void
pre: indent <> null
context ContoEconomicoComposite::stampaAlbero(indent: String, ultimo: boolean) : void
post: true
context ContoEconomicoComposite::getTotalePerTipo(tipo: TipoVoce) : double
pre: tipo <> null
context ContoEconomicoComposite::getTotalePerTipo(tipo: TipoVoce) : double
post: result = figli->iterate(c; acc: double = 0 | acc + c.getTotalePerTipo(tipo))

INVARIANTI DI CLASSE

ContoEconomicoComponentAbstract
context ContoEconomicoComponentAbstract
inv: nomevoce <> null
context ContoEconomicoComponentAbstract
inv: figli <> null
context ContoEconomicoComponentAbstract
inv: prezzo >= 0
context ContoEconomicoComponentAbstract
inv: figli->forAll(f | f <> null)

ContoEconomicoLeaf

context ContoEconomicoLeaf
inv: importo >= 0
context ContoEconomicoLeaf
inv: tipo <> null
context ContoEconomicoLeaf
inv: tipo.oclIsKindOf(TipoVoce)
context ContoEconomicoLeaf
inv: figli→isEmpty()

ContoEconomicoComposite
context ContoEconomicoComposite
inv: figli <> null
context ContoEconomicoComposite
inv: figli->forAll(f | f <> null)
context ContoEconomicoComposite
inv: figli->forAll(f | f.oclIsKindOf(ContoEconomicoComponentAbstract))
context ContoEconomicoComposite
inv: getImportoTotale() = figli->iterate(c; acc: double = 0 | acc + c.getImportoTotale())

TipoVoce
context TipoVoce
inv: self = TipoVoce::CAMERA or self = TipoVoce::SERVIZIO or self = TipoVoce::TRATTAMENTO or self = TipoVoce::ALTRO or self = TipoVoce::STIPENDI

Storage Interfaces
STORAGE
FrontDeskStorage
FrontDeskStorage
context FrontDeskStorage::doSave(o: T) : void
pre: o <> null
context FrontDeskStorage::doSave(o: T) : void
post: true
context FrontDeskStorage::doSaveAll(list: Collection(T)) : void
pre: list <> null
context FrontDeskStorage::doSaveAll(list: Collection(T)) : void
post: true
context FrontDeskStorage::doDelete(o: T) : void
pre: o <> null
context FrontDeskStorage::doDelete(o: T) : void
post: true
context FrontDeskStorage::doRetriveByKey(oggetto: Object) : T
pre: oggetto <> null
context FrontDeskStorage::doRetriveByKey(oggetto: Object) : T
post: result = null or result <> null
context FrontDeskStorage::doRetriveAll(order: String) : Collection(T)
pre: order <> null and order <> ""
context FrontDeskStorage::doRetriveAll(order: String) : Collection(T)
post: result <> null

context FrontDeskStorage::doUpdate(o: T) : void
pre: o <> null
context FrontDeskStorage::doUpdate(o: T) : void
post: true
context FrontDeskStorage::doRetriveByAttribute(attribute: String, value: Object) : Collection(T)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context FrontDeskStorage::doRetriveByAttribute(attribute: String, value: Object) : Collection(T)
post: result <> null

GovernanteStorage
context GovernanteStorage::doSave(o: T) : void
pre: o <> null
context GovernanteStorage::doSave(o: T) : void
post: true
context GovernanteStorage::doDelete(o: T) : void
pre: o <> null
context GovernanteStorage::doDelete(o: T) : void
post: true
context GovernanteStorage::doRetriveAll(order: String) : Collection(T)
pre: order <> null and order <> ""
context GovernanteStorage::doRetriveAll(order: String) : Collection(T)
post: result <> null
context GovernanteStorage::doUpdate(o: T) : void
pre: o <> null
context GovernanteStorage::doUpdate(o: T) : void
post: true
context GovernanteStorage::doRetriveByAttribute(attribute: String, value: String) : Collection(T)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context GovernanteStorage::doRetriveByAttribute(attribute: String, value: String) :

Collection(T)
post: result <> null

BackofficeStorage
context BackofficeStorage::doSave(o: T) : void
pre: o <> null and o.codiceFiscale <> null
context BackofficeStorage::doSave(o: T) : void
post: true
context BackofficeStorage::doDelete(o: T) : void
pre: o <> null and o.codiceFiscale <> null
context BackofficeStorage::doDelete(o: T) : void
post: true
context BackofficeStorage::doRetriveByKey(index: Object) : T
pre: index <> null and (index.oclIsKindOf(String) or index.oclIsKindOf(Integer))
context BackofficeStorage::doRetriveByKey(index: Object) : T
post: result <> null and (result.codiceFiscale = index or result.id = index)
context BackofficeStorage::doRetriveAll(order: String) : Collection(T)
pre: order = null or whitelist→includes(order)
context BackofficeStorage::doRetriveAll(order: String) : Collection(T)
post: result <> null and result->size() >= 0
context BackofficeStorage::doUpdate(o: T) : void
pre: o <> null and o.codiceFiscale <> null
context BackofficeStorage::doUpdate(o: T) : void
post: true
context BackofficeStorage::doRetriveByAttribute(attribute: String, value: Object) : Collection(T)
pre: attribute <> null and attribute <> "" and value <> null
context BackofficeStorage::doRetriveByAttribute(attribute: String, value: Object) : Collection(T)
post: result <> null
context BackofficeStorage::doFilter(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(T)

pre: nome <> null or sesso <> null or ruolo <> null or orderBy <> null
context BackofficeStorage::doFilter(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(T)
post: result <> null

INVARIANTI DI CLASSE

FrontDeskStorage
context FrontDeskStorage
inv: true

GovernanteStorage
context GovernanteStorage
inv: true

BackofficeStorage
context BackofficeStorage

inv: true

Data Access Objects (DAO)

STORAGE

PrenotazioneDAO
context PrenotazioneDAO::doSave(p: Prenotazione) : void
pre: p <> null
context PrenotazioneDAO::doSave(p: Prenotazione) : void
post: true
context PrenotazioneDAO::doSaveAll(list: Collection(Prenotazione)) : void
pre: list <> null
context PrenotazioneDAO::doSaveAll(list: Collection(Prenotazione)) : void
post: true
context PrenotazioneDAO::doDelete(p: Prenotazione) : void
pre: p <> null

context PrenotazioneDAO::doDelete(p: Prenotazione) : void
post: true
context PrenotazioneDAO::doRetriveByKey(codicePrenotazione: Object) : Prenotazione
pre: codicePrenotazione <> null
context PrenotazioneDAO::doRetriveByKey(codicePrenotazione: Object) : Prenotazione
post: result = null or result <> null
context PrenotazioneDAO::doRetriveAll(order: String) : Collection(Prenotazione)
pre: order <> null
context PrenotazioneDAO::doRetriveAll(order: String) : Collection(Prenotazione)
post: result <> null
context PrenotazioneDAO::doUpdate(p: Prenotazione) : void
pre: p <> null and p.IDPrenotazione <> null
context PrenotazioneDAO::doUpdate(p: Prenotazione) : void
post: true
context PrenotazioneDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Prenotazione)
pre: attribute <> null and attribute <> "" and value <> null
context PrenotazioneDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Prenotazione)
post: result <> null
context PrenotazioneDAO::createView() : void
pre: true
context PrenotazioneDAO::createView() : void
post: true

TrattamentoDAO
context TrattamentoDAO::doSave(trattamento: Trattamento) : void
pre: o <> null
context TrattamentoDAO::doSave(trattamento: Trattamento) : void
post: true

context TrattamentoDAO::doSaveAll(list: Collection(Trattamento)) : void
pre: list <> null
context TrattamentoDAO::doSaveAll(list: Collection(Trattamento)) : void
post: true
context TrattamentoDAO::doDelete(trattamento: Trattamento) : void
pre: o <> null
context TrattamentoDAO::doDelete(trattamento: Trattamento) : void
post: true
context TrattamentoDAO::doRetriveByKey(nome: Object) : Trattamento
pre: nome <> null
context TrattamentoDAO::doRetriveByKey(nome: Object) : Trattamento
post: result = null or result <> null
context TrattamentoDAO::doRetriveAll(order: String) : Collection(Trattamento)
pre: order <> null and order <> ""
context TrattamentoDAO::doRetriveAll(order: String) : Collection(Trattamento)
post: result <> null
context TrattamentoDAO::doUpdate(trattamento: Trattamento) : void
pre: trattamento <> null and trattamento.getNome() <> null and trattamento.getPrezzo() >= 0
context TrattamentoDAO::doUpdate(trattamento: Trattamento) : void
post: true
context TrattamentoDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Trattamento)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context TrattamentoDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Trattamento)
post: result <> null

ServizioDAO
context ServizioDAO::doSave(servizio: Servizio) : void
pre: o <> null
context ServizioDAO::doSave(servizio: Servizio) : void

post: true
context ServizioDAO::doSaveAll(list: Collection(Servizio)) : void
pre: list <> null
context ServizioDAO::doSaveAll(list: Collection(Servizio)) : void
post: true
context ServizioDAO::doDelete(servizio: Servizio) : void
pre: o <> null
context ServizioDAO::doDelete(servizio: Servizio) : void
post: true
context ServizioDAO::doRetriveByKey(nome: Object) : Servizio
pre: oggetto <> null
context ServizioDAO::doRetriveByKey(nome: Object) : Servizio
post: result = null or result <> null
context ServizioDAO::doRetriveAll(order: String) : Collection(Servizio)
pre: order <> null and order <> ""
context ServizioDAO::doRetriveAll(order: String) : Collection(Servizio)
post: result <> null
context ServizioDAO::doUpdate(servizio: Servizio) : void
pre: servizio <> null and servizio.getNome() <> null and servizio.getPrezzo() >= 0
context ServizioDAO::doUpdate(servizio: Servizio) : void
post: true
context ServizioDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Servizio)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context ServizioDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Servizio)
post: result <> null

CameraDAO
context CameraDAO::doDelete(o: Camera) : void
pre: o <> null
context CameraDAO::doDelete(o: Camera) : void

post: true
context CameraDAO::doRetriveByKey(oggetto: Object) : Camera
pre: oggetto <> null
context CameraDAO::doRetriveByKey(oggetto: Object) : Camera
post: result = null or (result <> null and result.numeroCamera = oggetto)
context CameraDAO::doSave(o: Camera) : void
pre: o <> null
context CameraDAO::doSave(o: Camera) : void
post: true
context CameraDAO::doSaveAll(listCamera: Collection(Camera)) : void
pre: listCamera <> null and listCamera->size() > 0
context CameraDAO::doSaveAll(listCamera: Collection(Camera)) : void
post: true
context CameraDAO::doRetriveAll(order: String) : Collection(Camera)
pre: order <> null and order <> ""
context CameraDAO::doRetriveAll(order: String) : Collection(Camera)
post: result <> null
context CameraDAO::doUpdate(o: Camera) : void
pre: o <> null and o.getNumeroCamera() <> null and o.getStatoCamera() <> null and o.getCapacità() > 0 and o.getPrezzoCamera() >= 0
context CameraDAO::doUpdate(o: Camera) : void
post: true
context CameraDAO::doRetriveByAttribute(attribute: String, value: String) : Collection(Camera)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context CameraDAO::doRetriveByAttribute(attribute: String, value: String) : Collection(Camera)
post: result <> null
context CameraDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Camera)
pre: attribute <> null and attribute <> "" and value <> null and value <> ""
context CameraDAO::doRetriveByAttribute(attribute: String, value: Object) :

Collection(Camera)
post: result <> null

ImpiegatoDAO
context ImpiegatoDAO::doSave(impiegato: Impiegato) : void
pre: impiegato <> null and impiegato.codiceFiscale <> null
context ImpiegatoDAO::doSave(impiegato: Impiegato) : void
post: true
context ImpiegatoDAO::doDelete(impiegato: Impiegato) : void
pre: impiegato <> null and impiegato.codiceFiscale <> null
context ImpiegatoDAO::doDelete(impiegato: Impiegato) : void
post: true
context ImpiegatoDAO::doRetriveByKey(index: Object) : Impiegato
pre: index <> null and (index.oclIsKindOf(String) or index.oclIsKindOf(Integer))
context ImpiegatoDAO::doRetriveByKey(index: Object) : Impiegato
post: result <> null and (result.codiceFiscale = index or result.id = index)
context ImpiegatoDAO::doRetriveAll(order: String) : Collection(Impiegato)
pre: order = null or whitelist→includes(order)
context ImpiegatoDAO::doRetriveAll(order: String) : Collection(Impiegato)
post: result <> null and result->size() >= 0
context ImpiegatoDAO::doUpdate(impiegato: Impiegato) : void
pre: impiegato <> null and impiegato.codiceFiscale <> null
context ImpiegatoDAO::doUpdate(impiegato: Impiegato) : void
post: true
context ImpiegatoDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Impiegato)
pre: attribute <> null and attribute <> "" and value <> null
context ImpiegatoDAO::doRetriveByAttribute(attribute: String, value: Object) : Collection(Impiegato)
post: result <> null
context ImpiegatoDAO::doFilter(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(Impiegato)

pre: nome <> null or sesso <> null or ruolo <> null or orderBy <> null
context ImpiegatoDAO::doFilter(nome: String, sesso: String, ruolo: Ruolo, orderBy: String) : Collection(Impiegato)
post: result <> null

DaoUtils
context DaoUtils::checkWhitelist(whitelist: Collection(String), suspect: String) : boolean
pre: whitelist <> null and suspect <> null
context DaoUtils::checkWhitelist(whitelist: Collection(String), suspect: String) : boolean
post: result = true or result = false

INVARIANTI DI CLASSE

PrenotazioneDAO
context PrenotazioneDAO
inv: VIEW_TABLE_NAME = "prenotazioneClientiCamereView"
context PrenotazioneDAO
inv: whitelist <> null and whitelist->size() = 12
context PrenotazioneDAO

inv: draList <> null and draList->size() = 1
context PrenotazioneDAO
inv: clienteDAO <> null
context PrenotazioneDAO
inv: self.oclIsKindOf(FrontDeskStorage)

TrattamentoDAO
context TrattamentoDAO
inv: TABLE_NAME = "Trattamento"
context TrattamentoDAO
inv: whitelist <> null and whitelist->size() = 2
context TrattamentoDAO
inv: self.oclIsKindOf(FrontDeskStorage)

ServizioDAO
context ServizioDAO
inv: TABLE_NAME = "Servizio"
context ServizioDAO
inv: whitelist <> null and whitelist->size() = 2
context ServizioDAO
inv: self.oclIsKindOf(FrontDeskStorage)

CameraDAO
context CameraDAO
inv: self.oclIsKindOf(FrontDeskStorage)
context CameraDAO
inv: self.oclIsKindOf(GovernanteStorage)

ImpiegatoDAO
context ImpiegatoDAO
inv: TABLE_NAME = "Impiegato"
context ImpiegatoDAO

inv: whitelist <> null
context ImpiegatoDAO
inv: self.oclIsKindOf(BackofficeStorage)

DaoUtils
context DaoUtils
inv: true
Connection Storage

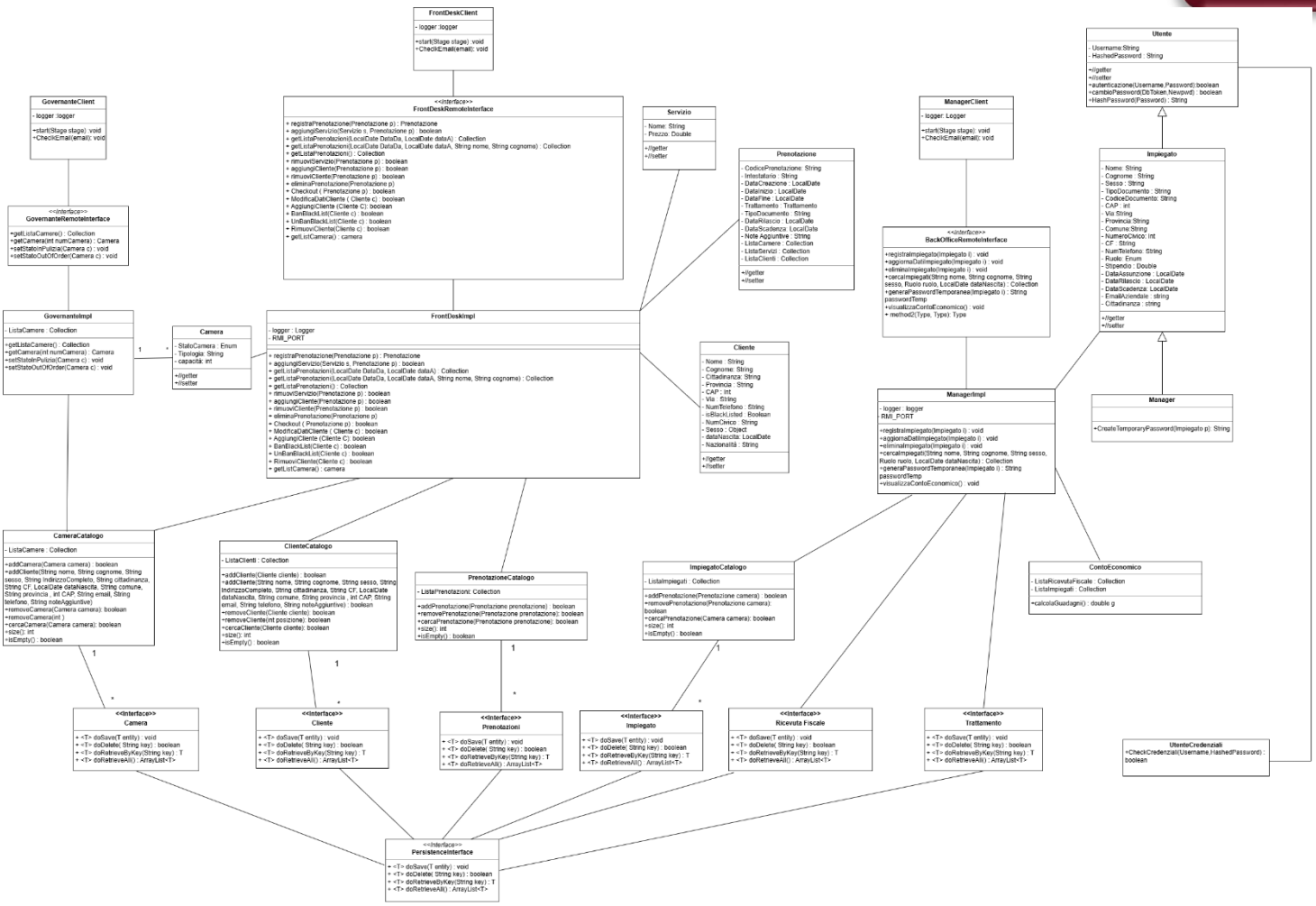
STORAGE

ConnectionStorage
context ConnectionStorage::createDBConnection() : Connection
pre: true
context ConnectionStorage::createDBConnection() : Connection
post: result <> null
context ConnectionStorage::getConnection() : Connection
pre: true
context ConnectionStorage::getConnection() : Connection
post: result <> null and result.isClosed() = false
context ConnectionStorage::releaseConnection(connection: Connection) : void
pre: connection <> null
context ConnectionStorage::releaseConnection(connection: Connection) : void
post: freeDbConnections->includes(connection) or connection.isClosed() = true
context ConnectionStorage::shutdown() : void
pre: true
context ConnectionStorage::shutdown() : void
post: freeDbConnections->isEmpty() and freeDbConnections->forAll(c | c.isClosed() = true)

INVARIANTI DI CLASSE

ConnectionStorage
context ConnectionStorage
inv: freeDbConnections <> null
context ConnectionStorage
inv: freeDbConnections->forAll(c | c <> null)
context ConnectionStorage
inv: freeDbConnections->size() <= 60
context ConnectionStorage
inv: freeDbConnections->forAll(c | c.isClosed() = false implies freeDbConnections→includes(c))

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
<<interface>>
FrontDeskRemoteInterface
+ registraPrenotazione(Prenotazione p) : 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)
+ Checkout (Prenotazione p) : boolean
+ ModificaDatiCliente (Cliente c) : boolean
+ AggiungiCliente (Cliente C): 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
ManagerClient
- logger: Logger
+start(Stage stage): void
+CheckEmail(email): 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
<<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
+generalPasswordTemporanea(Impiegato i) : String passwordTemp
+visualizzaContoEconomico() : void
+ method2(Type, Type): Type
GovernanteImpl
- ListaCamere : Collection
+getListaCamere() : Collection
+getCamera(int numCamera) : Camera
+setStatoInPulizia(Camera c) : void
+setStatoOutOfOrder(Camera c) : void
1
*
Camera
- StatoCamera : Enum
- Tipologia: String
- capacita: int
+/-getter
+/-setter
FrontDeskImpl
- logger : Logger
- RMI_PORT
+ registraPrenotazione(Prenotazione p) : 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)
+ Checkout (Prenotazione p) : boolean
+ ModificaDatiCliente (Cliente c) : boolean
+ AggiungiCliente (Cliente C): boolean
+ BanBlackList(Cliente c) : boolean
+ UnBanBlackList(Cliente c) : boolean
+ RimuoviCliente(Cliente c) : boolean
+ getListCamera() : camera
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
ManagerImpl
- logger : logger
- RMI_PORT
+registraImpiegato(Impiegato i) : void
+aggiornaDatiImpiegato(Impiegato i) : void
+eliminaImpiegato(Impiegato i) : void
+cercaImpiegati(String nome, String cognome, String sesso, Ruolo ruolo, LocalDate dataNascita) : Collection
+generalPasswordTemporanea(Impiegato i) : String passwordTemp
+visualizzaContoEconomico() : void
Manager
+CreateTemporaryPassword(Impiegato p): String
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)
+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
ContoEconomico
- ListaRicevutaFiscale : Collection
- ListaImpiegati : Collection
+calcolaGuadagni() : double g
<<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
+ <T> doRetrieveAll() : ArrayList<T>
<<Interface>>
Prenotazioni
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>
<<Interface>>
Impiegato
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>
<<Interface>>
Ricevuta Fiscale
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>
<<Interface>>
Trattamento
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>
UtenteCredenziali
+CheckCredenziali(Username,HashedPassword) : boolean
<<Interface>>
PersistenceInterface
+ <T> doSave(T entity) : void
+ <T> doDelete(String key) : boolean
+ <T> doRetrieveByKey(String key) : T
+ <T> doRetrieveAll() : ArrayList<T>