Università degli Studi di Salerno
Corso di Ingegneria del Software

Dream&Fly
Object Design Document
Versione 0.3

Data: 06/06/2025

Torna all'indice

Torna all’indice

Coordinatore del progetto:

Nome	Matricola
Singh Amandeep	0512113476
Paolillo Valentina	0512114820

Partecipanti:

Nome	Matricola
Paolillo Valentina	0512114820
Singh Amandeep	0512113476

Scritto da:	Singh Amandeep, Paolillo Valentina

Revision History

Data	Versione	Descrizione	Autore
01/02/2024	0.1	Prima stesura documento	Singh Amandeep, Paolillo Valentina
02/02/2024	0.2	Stesura sezione 2. Packages	Singh Amandeep, Paolillo Valentina
06/06/2025	0.3	Completamento stesura documento e revisione	Paolillo Valentina, Singh Amandeep

Indice

1. INTRODUCTION 5
1.1. Object design trade-offs 5
1.1.1. Tempo di rilascio VS Funzionalità 5
1.1.2. Velocità VS Memoria 5
1.1.3. Costruire VS Comprare 5
1.2. Off-the-shelf components 5
1.3. Interface documentation guidelines 5
1.4. Definitions, acronyms and abbreviations 6
1.5. References 7
2. PACKAGES 7
2.1. Subsystem decomposition 7
2.1.1. Interface 8
2.1.2. Application logic 8
2.1.3. Storage 9
3. CLASS INTERFACES 9
3.1. AccountUserDao 9
3.2. CapsulaDao 11
3.3. FasciaOrariaDao 12
3.4. PrenotabileDao 13
3.5. PrenotazioneDao 16

1. INTRODUCTION

1.1. Object design trade-offs

1.1.1. Tempo di rilascio VS Funzionalità

Per garantire un'esperienza dotata di tutte le funzionalità essenziali per suscitare un impatto positivo sui primi clienti, abbiamo scelto di mettere in secondo piano il tempo di rilascio per prioritizzare la qualità del sistema.

1.1.2. Velocità VS Memoria

Utilizzare più memoria per ottimizzare l'accesso veloce ai dati può aumentare le prestazioni, anche se potrebbe richiedere risorse aggiuntive.

1.1.3. Costruire VS Comprare

Sebbene utilizzare software già realizzato da altri permetta, ad esempio, l'utilizzo di funzionalità già complete oppure una minore quantità di lavoro per gli sviluppatori, è stato deciso di realizzare la maggior parte del Sistema partendo da zero, utilizzando componenti esterne soltanto in alcuni casi.

Trade-Off	
Tempo di rilascio	**Funzionalità**
Velocità	Memoria
Costruire	Comprare

1.2. Off-the-shelf components

Il Sistema utilizzerà i seguenti componenti off-the-shelf:

1. **JQuery:** framework ampiamente utilizzato per semplificare la scrittura di codice JavaScript, facilitando l'interazione con il DOM (Document Object Model) e semplificando le operazioni comuni.
2. **MySQL (v.8.0):** sistema open source di gestione di database relazionali SQL sviluppato e supportato da Oracle.
3. **MySQL Connector/J (JDBC driver):** driver JDBC ufficiale che consente e semplifica la comunicazione tra applicazioni Java e il database MySQL.
4. **Tomcat (v.9.X):** Web Server con annesso application container per applicazioni scritte in Java; utilizzato per eseguire il back-end dell'applicazione.

1.3. Interface documentation guidelines

Tali linee guida includono una lista di regole che gli sviluppatori devono seguire durante la progettazione del sistema. Il progetto Dream&Fly è realizzato con l'IDE di sviluppo EclipseIDE ed è strutturato nel seguente modo:

- **Package:** il nome del pacchetto deve essere sempre in minuscolo.
- **Classi**: i nomi delle classi devono essere descrittivi e scritti in UpperCamelCase.
- **Interfacce:** i nomi delle interfacce devono essere scritte in UpperCamelCase.

- **Metodi**: i metodi devono essere scritti in forma camelCase.
- **Variabili:** il nome delle variabili deve essere descrittivo, si evita quindi di utilizzare variabili con nomi composti da una sola lettera se non per variabili temporanee.
- **Indentazione**:
  - Le istruzioni racchiuse all'interno di un blocco (esempio: for), devono essere indentate di un'unità all'interno dell'istruzione composta.
  - La parentesi di apertura del blocco deve trovarsi alla fine della riga dell'istruzione composta.
  - La parentesi di chiusura del blocco deve trovarsi allo stesso livello di indentazione dell'istruzione composta.
- **HTML:** convenzione usata per definire le linee guida su html:
  HTML Style Guide and Coding Conventions - W3School
- **CSS:** tutti gli stili non in-line devono essere collocati in fogli di stile separati.
- **JavaScript**:
  - Gli script devono essere collocati in file appositi.
  - Il codice JavaScript deve seguire le stesse convenzioni per il layout e i nomi del codice Java.
- **DataBase SQL:** i nomi delle tabelle e dei campi devono seguire le seguenti regole:
  - Devono essere costituiti da sole lettere minuscole.
  - Se il nome è costituito da più parole, è previsto l'uso di underscore (_).

1.4. Definitions, acronyms and abbreviations

- **ODD:** object design document
- **JQuery:** framework utilizzato per semplificare la scrittura di codice JavaScript
- **MySQL**: sistema di gestione di database relazionali SQL
- **MySQL Connector/J (JDBC driver):** driver JDBC ufficiale che consente e semplifica la comunicazione tra applicazioni Java e il database MySQL.
- **SQL**: Structured Query Language
- **JDBC:** Java DataBase Connectivity
- **HTML:** HyperText Markup Language
- **CSS:** Cascading Style Sheets
- **Maven:** strumento di build automation
- **GUI:** interfaccia grafica
- **DAO:** Data Access Object

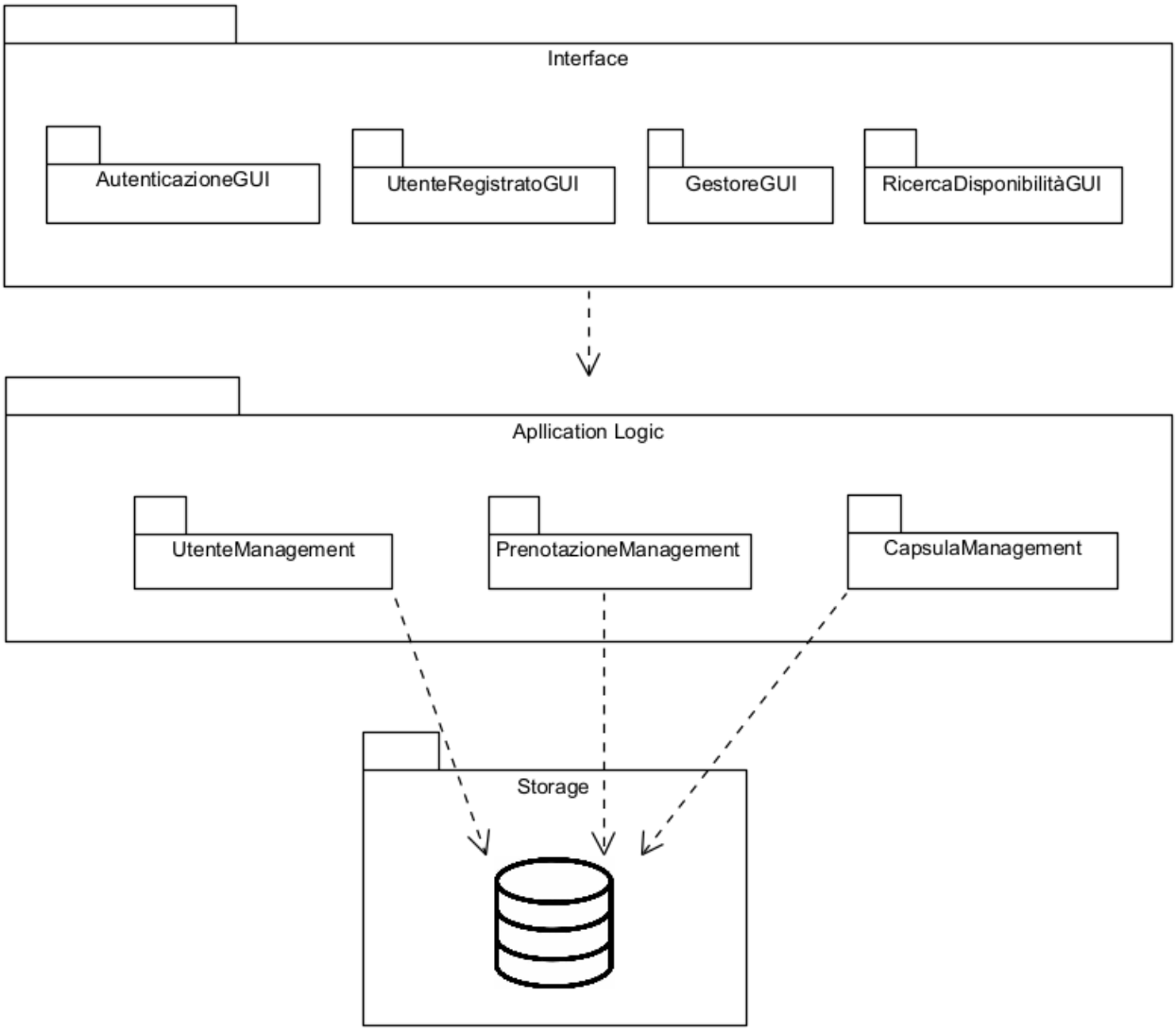
1.5. References

Per stilare la presente documentazione, si è preso come riferimento il libro di testo “Object-Oriented Software Engineering Using UML, Patterns and Java: Third Edition, di Bernd Bruegge ed Allen H. Dutoit”.

2. PACKAGES

In questa sezione viene mostrata la suddivisione del Sistema in package, in base a quanto definito nel documento di System Design. Tale suddivisione è motivata dalle scelte architetturali prese e sottolinea la struttura di directory standard definita da Maven.

2.1. Subsystem decomposition



2.1.1. Interface

Questo package contiene i seguenti sub-package e le seguenti classi:

- Package AutenticazioneGUI
  - InserisciCodice
  - InserisciEmail
  - Login
  - Registrati
  - Reimposta password
- Package UtenteRegistratoGUI
  - AreaUtente
  - ConfermaPrenotazione
  - LeMiePrenotazioni
  - Pagamento
- Package GestoreGUI
  - GestoreAccount
    - AreaRiservataGestoreAccount
    - RegistraNuovoAccountGestore
    - Visualizza-EliminaAccount
  - GestoreCapsule
    - AreaRiservataGestoreCapsule
    - ModificaDisponibilità
    - ModificaPrezzo
    - ProlungaDisponibilità
    - RegistraCapsula
    - VisualizzaCapsula
  - GestorePrenotazioni
    - AreaRiservataGestorePrenotazioni
    - VisualizzaPrenotazioniGestore
- Package RicercaDisponibilitàGUI
  - CapsuleDisponibili

Inoltre, vi si presentano anche le seguenti interfacce: HomePage, Header e Footer.

2.1.2. Application logic

Questo package contiene i seguenti sub-package e le seguenti classi:

- Package utenteManagement
  - EliminaAccountServlet
  - EmailDisponibility
  - InserisciCodiceServlet
  - InserisciEmailServlet
  - LeMiePrenotazioniServlet
  - LoginServlet
  - LogoutServlet
  - ModificaDatiServlet
  - RegistraNuovoAccountGestoreServlet
  - RegistratiServlet
  - RicercaAccountServlet
  - VisualizzaAccountServlet

- Package prenotazioneManagement
  - PagamentoServlet
  - VisualizzaPrenotazioneGestoreFiltriServlet
  - VisualizzaPrenotazioneGestoreServlet

- Package capsulaManagement
  - CapsuleDisponibiliServlet
  - GetFasceOrarieServlet
  - ModificaDisponibilitaServlet
  - ModificaPrezzoServlet
  - NumberDisponibilityServlet
  - ProlungaDisponibilitàServlet
  - RegistraCapsulaServlet
  - RicercaDisponibilitàServlet
  - VisualizzaCapsuleServlet

2.1.3. Storage

Questo package contiene le seguenti classi:

- AccountUser
- AccountUserDao
- Capsula
- CapsulaDao
- FasciaOraria
- FasciaOrariaDao
- Prenotabile
- PrenotabileDao
- Prenotazione
- PrenotazioneDao

3. CLASS INTERFACES

3.1. AccountUserDao

NOME CLASSE	**AccountUserDao**
DESCRIZIONE	Si occupa dell'accesso e la gestione degli account nel database.
doRetrieveByKey	Metodo per recuperare un account tramite e-mail. **context** AccountUserDao::doRetrieveByKey(email: String) : AccountUser **pre**: email <> null and email.trim() <> '' **post**: (result.email = email and result.password <> null and result.name <> null and result.surname <> null

	and result.number <> null and result.ruolo >= 0) or (result.email = null and result.password = null and result.name = null and result.surname = null and result.number = null and result.ruolo = 0)
doUpdateNumber	Metodo per modificare il numero di cellulare associato a un account. **context** AccountUserDao::doUpdateNumber(email: String, cellulare: String) **pre**: email <> null and cellulare <> null and cellulare.matches('\\+?[0-9]+') **post**: self.doRetrieveByKey(email).number = cellulare
doUpdatePassword	Metodo per modificare la password associata a un account. **context** AccountUserDao::doUpdatePassword(email: String, password: String) **pre**: email <> null and password <> null and password.size() >= 8 **post**: self.doRetrieveByKey(email).password = password
doSave	Metodo per registrare un nuovo account. **context** AccountUserDao::doSave(user: AccountUser) **pre**: user <> null and user.email <> null and user.password <> null and user.name <> null and user.surname <> null and user.number <> null **post**: self.doRetrieveByKey(user.email).email = user.email and self.doRetrieveByKey(user.email).password = user.password and self.doRetrieveByKey(user.email).name = user.name and self.doRetrieveByKey(user.email).surname = user.surname and self.doRetrieveByKey(user.email).number = user.number

doSaveGestore	Metodo per registrare un nuovo account gestore **context** AccountUserDao::doSaveGestore(user: AccountUser) **pre**: user <> null and user.email <> null and user.password <> null and user.name <> null and user.surname <> null and user.number <> null and user.ruolo = 1 **post**: self.doRetrieveByKey(user.email).email = user.email and self.doRetrieveByKey(user.email).password = user.password and self.doRetrieveByKey(user.email).name = user.name and self.doRetrieveByKey(user.email).surname = user.surname and self.doRetrieveByKey(user.email).number = user.number and self.doRetrieveByKey(user.email).ruolo = 1
doRetrieveAll	Metodo che restituisce una lista contenete tutti gli account. **context** AccountUserDao::doRetrieveAll(): Collection(AccountUser) **pre:** true **post**: result->forAll(u | u.email <> null and u.name <> null and u.surname <> null and u.number <> null)
doDelete	Metodo per eliminare un account. **context** AccountUserDao::doDelete(email: String) **pre**: email <> null **post**: AccountUser.allInstances()->forAll(u | u.email <> email)

3.2. CapsulaDao

NOME CLASSE	**CapsulaDao**
DESCRIZIONE	Si occupa dell'accesso e la gestione delle capsule nel database.
doRetrieveByKey	Metodo per recuperare una capsula tramite id. **context** CapsulaDao::doRetrieveByKey(id: Integer) **pre**: id <> null **post**: (result.id = id and result.tipologia <> null and result.prezzo_orario >= 0) or

	(result.id = null and result.tipologia = null and result.prezzo_orario = 0)
doSave	Metodo per salvare una nuova capsula. **context** CapsulaDao::doSave(capsula: Capsula) **pre**: capsula <> null and capsula.id <> null and capsula.tipologia <> null and capsula.prezzo_orario >= 0 **post**: self.doRetrieveByKey(capsula.id).id = capsula.id and self.doRetrieveByKey(capsula.id).tipologia = capsula.tipologia and self.doRetrieveByKey(capsula.id).prezzo_orario = capsula.prezzo_orario
doUpdatePrezzoOrario	Metodo per modificare il prezzo orario di una capsula. **context** CapsulaDao::doUpdatePrezzoOrario(id: Integer, prezzo_orario: Real) **pre**: id <> null and prezzo_orario >= 0 **post:** self.doRetrieveByKey(id).prezzo_orario = prezzo_orario
doRetrieveAll()	Metodo per recuperare una lista di tutte le capsule. **context** CapsulaDao::doRetrieveAll() **pre**: true **post**: result->forAll(c | c.id <> null and c.tipologia <> null and c.prezzo_orario >= 0)

3.3. FasciaOrariaDao

NOME CLASSE	**FasciaOrariaDao**
DESCRIZIONE	Si occupa dell'accesso alla tabella `fascia_oraria` del database.
doRetrieveByKey	Metodo per recuperare una fascia oraria tramite numero. **context** FasciaOrariaDao::doRetrieveByKey(numero: Integer) **pre**: numero <> null **post:** (result.numero = numero and result.orarioInizio <> null and result.orarioFine <> null) or (result.numero = 0 and result.orarioInizio = null and result.orarioFine = null)
doRetrieveAll()	Metodo che restituisce tutte le fasce orarie presenti nel database. **context** FasciaOrariaDao::doRetrieveAll() **pre**: true **post:** result->forAll(f | f.numero <> 0 and f.orarioInizio <> null and

	f.orarioFine <> null)
doRetrieveByOrarioInizio	Metodo che restituisce il numero della fascia oraria il cui orario di inizio corrisponde a `orario`. **context** FasciaOrariaDao::doRetrieveByOrarioInizio(orario: String) **pre:** orario <> null **post:** result > 0
doRetrieveByOrarioFine	Metodo che restituisce il numero della fascia oraria il cui orario di fine corrisponde a `orario`. **context** FasciaOrariaDao::doRetrieveByOrarioFine(orario: String) **pre:** orario <> null **post:** result > 0

3.4.PrenotabileDao

NOME CLASSE	**PrenotabileDao**
DESCRIZIONE	Si occupa dell'accesso e della gestione della tabella `e_prenotabile` del database.
doRetrieveAll()	Metodo per recuperare tutti gli oggetti prenotabili nel database. **context** PrenotabileDao::doRetrieveAll() **pre:** true **post:** result->forAll(p | p.dataPrenotabile <> null and p.capsulaId <> null and p.fasciaOrariaNumero <> null)
doSave	Metodo per salvare una nuova prenotazione. **context** PrenotabileDao::doSave(prenotabile: Prenotabile) **pre:** prenotabile <> null and prenotabile.dataPrenotabile <> null and prenotabile.capsulaId <> 0 and prenotabile.fasciaOrariaNumero <> 0 post: self.doRetrieveById(prenotabile.capsulaId)->exists(p | p.dataPrenotabile = prenotabile.dataPrenotabile and p.fasciaOrariaNumero = prenotabile.fasciaOrariaNumero)
doDelete	Metodo per eliminare una prenotazione specifica. **context** PrenotabileDao::doDelete(data: String, capsula_id: Integer, fasciaOraria: Integer) **pre:** data <> null and capsula_id > 0 and fasciaOraria > 0 **post:** not self.doRetrieveById(capsula_id)->exists(p |

	p.dataPrenotabile = data and p.fasciaOrariaNumero = fasciaOraria)
doRetrieveLastDateById	Metodo per ottenere l'ultima data disponibile per una capsula. **context** PrenotabileDao::doRetrieveLastDateById(id: Integer) **pre:** id > 0 **post:** result.capsulaId = id and self.doRetrieveById(id)->forAll(p | p.dataPrenotabile <= result.dataPrenotabile)
doRetrieveIdByDataInizioDataFine	Metodo per ottenere gli ID delle capsule presenti sia nella data di inizio che nella data di fine. **context** PrenotabileDao::doRetrieveIdByDataInizioDataFine(dataInizio: String, dataFine: String) **pre:** dataInizio <> null and dataFine <> null **post:** result->forAll(id | self.doRetrieveById(id)->exists(p | p.dataPrenotabile = dataInizio) and self.doRetrieveById(id)->exists(p | p.dataPrenotabile = dataFine))
doRetrieveByIdAndDate	Metodo per verificare se esiste una prenotazione per una certa data e capsula. **context** PrenotabileDao::doRetrieveByIdAndDate(capsula_id: Integer, data: String) **pre:** capsula_id <> null and data <> null **post:** result = self.doRetrieveById(capsula_id)->exists(p | p.dataPrenotabile = data)
doRetrieveByIdAndFasciaOrariaAndDate	Metodo per verificare se esiste una prenotazione per una certa capsula, fascia oraria e data. **context** PrenotabileDao::doRetrieveByIdAndFasciaOrariaAndDate(capsula_id: Integer, fascia_oraria: Integer, data: String) **pre:** capsula_id <> null and fascia_oraria <> null and data <> null **post:** result = self.doRetrieveById(capsula_id)->exists(p | p.fasciaOrariaNumero = fascia_oraria and

	p.dataPrenotabile = data)
doRetrieveById	Metodo per recuperare tutte le prenotazioni per una capsula. **context** PrenotabileDao::doRetrieveById(id: Integer): **pre:** id <> null **post:** result->forAll(p | p.capsulaId = id)
doRetrieveByDataInizio	Metodo per ottenere tutte le prenotazioni a partire da una certa data. **context** PrenotabileDao::doRetrieveByDataInizio(data: String): **pre:** data <> null **post:** result->forAll(p | p.dataPrenotabile >= data)
doRetrievePrenotabiliByCapsulaAndDataInizio	Metodo per ottenere tutte le prenotazioni di una capsula a partire da una certa data. **context** PrenotabileDao::doRetrievePrenotabiliByCapsulaAndDataInizio(capsulaId: Integer, dataInizio: String) **pre:** capsulaId <> null and dataInizio <> null **post:** result->forAll(p | p.capsulaId = capsulaId and p.dataPrenotabile >= dataInizio)
doRetrievePrenotabiliByCapsulaIdAndDataFine	Metodo per ottenere tutte le prenotazioni di una capsula fino a una certa data. **context** PrenotabileDao::doRetrievePrenotabiliByCapsulaIdAndDataFine(capsula_id: Integer, dataFine: String) **pre:** capsula_id <> null and dataFine <> null **post:** result->forAll(p | p.capsulaId = capsula_id and p.dataPrenotabile <= dataFine)
doRetrieveByDataFine	Metodo che restituisce tutte le prenotazioni con data inferiore o uguale a quella fornita. **context** PrenotabileDao::doRetrieveByDataFine(data: String) **pre:** data <> null **post:** result->forAll(p | p.dataPrenotabile <= data)

doRetrieveByDataInizioAndDataFine	Metodo che restituisce tutte le prenotazioni tra due date. **context** PrenotabileDao::doRetrieveByDataInizioAndDataFine(dataInizio: String, dataFine: String) **pre:** dataInizio <> null and dataFine <> null post: result->forAll(p | p.dataPrenotabile >= dataInizio and p.dataPrenotabile <= dataFine)
doRetrievePrenotabileByCapsulaIdAndDataInizioAndDataFine	Metodo che restituisce tutte le prenotazioni per una capsula in un certo intervallo di date. **context** PrenotabileDao::doRetrievePrenotabileByCapsulaIdAndDataInizioAndDataFine(capsula_id: Integer, dataInizio: String, dataFine: String) **pre:** capsula_id <> null and dataInizio <> null and dataFine <> null **post:** result->forAll(p | p.capsulaId = capsula_id and p.dataPrenotabile >= dataInizio and p.dataPrenotabile <= dataFine)

3.5.PrenotazioneDao

NOME CLASSE	**PrenotazioneDao**
DESCRIZIONE	Si occupa dell'accesso e della gestione dei dati relativi alle prenotazioni in un database.
doRetrieveByKey	Metodo per recuperare una prenotazione tramite il codice. **context** PrenotazioneDao::doRetrieveByKey(codice: Integer) **pre**: codice <> null **post**: (result.codiceDiAccesso = codice and result.prezzoTotale >= 0 and result.userAccountEmail <> null and result.capsulaId > 0) or (result.codiceDiAccesso = null and result.userAccountEmail = null and result.capsulaId = 0)

doSave	Metodo per salvare una prenotazione **context** PrenotazioneDao::doSave(p: Prenotazione) **pre**: p <> null and p.orarioInizio <> null and p.orarioFine <> null and p.dataInizio <> null and p.dataFine <> null and p.prezzoTotale >= 0 and p.dataEffettuazione <> null and p.userAccountEmail <> null and p.capsulaId > 0 **post**: let codice = result in self.doRetrieveByKey(codice).orarioInizio = p.orarioInizio and self.doRetrieveByKey(codice).orarioFine = p.orarioFine and self.doRetrieveByKey(codice).dataInizio = p.dataInizio and self.doRetrieveByKey(codice).dataFine = p.dataFine and self.doRetrieveByKey(codice).prezzoTotale = p.prezzoTotale and self.doRetrieveByKey(codice).dataEffettuazione = p.dataEffettuazione and self.doRetrieveByKey(codice).userAccountEmail = p.userAccountEmail and self.doRetrieveByKey(codice).capsulaId = p.capsulaId
doUpdateValidita	Metodo per modificare la validità della prenotazione. **context** PrenotazioneDao::doUpdateValidita(codice: Integer, valid: Boolean) **pre**: codice <> null **post**: self.doRetrieveByKey(codice).validita = valid
doUpdateRimborso	Metodo per determinare il rimborso della prenotazione. **context** PrenotazioneDao::doUpdateRimborso(codice: Integer, rimborso: Real) **pre**: codice <> null and rimborso >= 0 **post**: self.doRetrieveByKey(codice).rimborso = rimborso
doRetrieveAll	Metodo per ricavare tutte le prenotazioni presenti nel database. **context** PrenotazioneDao::doRetrieveAll() **pre**: true **post**: result->forAll(p | p.codiceDiAccesso <> null and p.userAccountEmail <> null and p.capsulaId > 0 and

	p.prezzoTotale >= 0)
doRetrieveByEmail	Metodo per ricavare le prenotazioni di uno specifico account tramite l'e-mail. **context** PrenotazioneDao::doRetrieveByEmail(e-mail: String, chiamante: Integer) **pre**: e-mail <> null **post**: result->forAll(w | w.prenotazione.userAccountEmail = email)
doRetrieveEmailWithPrenotazione	Questo metodo si occupa di recuperare l'indirizzo e-mail dell'utente associato a una determinata prenotazione **context** PrenotazioneDao::doRetrieveEmailWithPrenotazione() **pre**: true **post**: result->forAll(p | p.userAccountEmail <> null and p.codiceDiAccesso.oclIsUndefined())
doRetrievePrenotazioniByNumeroCapsulaAll	Metodo per ricavare le prenotazioni riguardanti una specifica capsula, identificata tramite ID. **context** PrenotazioneDao::doRetrievePrenotazioniByNumeroCapsulaAll(capsId: Integer) **pre**: capsId <> null **post**: result->forAll(p | p.capsulaId = capsId)
doRetrievePrenotazioniByAccount	Metodo per ricavare le prenotazioni riguardante uno specifico account **context** PrenotazioneDao::doRetrievePrenotazioniByAccount(e-mail: String) **pre**: email <> null **post**: result->forAll(p | p.userAccountEmail = email)
doRetrievePrenotazioniByDataInizio	Metodo per ricavare le prenotazioni a partire da una certa data. **context** PrenotazioneDao::doRetrievePrenotazioniByDataInizio(dataInizio: String) **pre**: dataInizio <> null **post**: result->forAll(p | p.dataInizio >= dataInizio)

doRetrievePrenotazioniByDataFine	Metodo per ricavare le prenotazioni entro una certa data **context** PrenotazioneDao::doRetrievePrenotazioniByDataFine(dataFine : String) **pre**: dataFine <> null **post**: result->forAll(p | p.dataFine <= dataFine)
doRetrievePrenotazioniByDataInizioAndFine	Metodo per ricavare le prenotazioni da una certa data fino a un'altra data. **context** PrenotazioneDao::doRetrievePrenotazioniByDataInizioAndFine (dIn: String, dFi: String) **pre**: dIn <> null and dFi <> null **post**: result->forAll(p | p.dataInizio >= dIn and p.dataFine <= dFi)
doRetrievePrenotazioniByDataInizioAndAccount	Metodo per ricavare le prenotazioni a partire da una data di inizio e di uno specifico account. **context** PrenotazioneDao::doRetrievePrenotazioniByDataInizioAndAccount(dIn: String, email: String) **pre**: dIn <> null and email <> null **post**: result->forAll(p | p.dataInizio >= dIn and p.userAccountEmail = email)
doRetrieveByDataFineAndAccount	Metodo per filtrare le prenotazioni fino alla data fine e di uno specifico account. **context** PrenotazioneDao::doRetrieveByDataFineAndAccount(dFi: String, email: String) **pre**: dFi <> null and email <> null **post**: result->forAll(p | p.dataFine <= dFi and p.userAccountEmail = email)
doRetrievePrenotazioniByNumeroCapsulaAndAccount	Metodo per ricavare le prenotazioni di una specifica capsula e di uno specifico account. **context** PrenotazioneDao::doRetrievePrenotazioniByNumeroCapsulaAndAccount(capsId: Integer, email: String) **pre**: capsId <> null and email <> null **post**: result->forAll(p | p.capsulaId = capsId and p.userAccountEmail = email)

doRetrievePrenotazioniByAll	Metodo per ricavare le prenotazioni secondo le seguenti specifiche: account, data inizio e data fine. **context** PrenotazioneDao::doRetrievePrenotazioniByAll(capsId: Integer, email: String, dIn: String, dFi: String) **pre**: capsId <> null and email <> null and dIn <> null and dFi <> null **post**: result->forAll(p | p.capsulaId = capsId and p.userAccountEmail = email and p.dataInizio >= dIn and p.dataFine <= dFi)
doRetrievePrenotazioniByAccountAndIdAndDataInizio	Metodo per ricavare le prenotazioni in base alla capsula, all'account, e alla data di inizio. **context** PrenotazioneDao::doRetrievePrenotazioniByAccountAndIdAndDataInizio(capsId: Integer, email: String, dIn: String) : Collection(Prenotazione) **pre**: capsId <> null and email <> null and dIn <> null **post**: result->forAll(p | p.capsulaId = capsId and p.userAccountEmail = email and p.dataInizio >= dIn)
doRetrievePrenotazioniByAccountAndIdAndDataFine	Metodo per ricavare le prenotazioni in base all'account,alla capsula e alla data di fine. **context** PrenotazioneDao::doRetrievePrenotazioniByAccountAndIdAndDataFine(capsId: Integer, email: String, dFi: String) **pre**: capsId <> null and email <> null and dFi <> null **post**: result->forAll(p | p.capsulaId = capsId and p.userAccountEmail = email and p.dataFine <= dFi)
doRetrievePrenotazioniByCapsulaAndDataInizio	Metodo per ricavare le prenotazioni in base alla capsula e alla data di inizio. **context** PrenotazioneDao::doRetrievePrenotazioneByCapsulaAndDataInizio(capsId: Integer, dIn: String) : Collection(Prenotazione) **pre**: capsId <> null and dIn <> null **post**: result->forAll(p | p.capsulaId = capsId and p.dataInizio >= dIn)

doRetrievePrenotazioniByCapsulaAndDataFine	Metodo per ricavare le prenotazioni fino alla data di fine indicata e di una specifica capsula. **context** PrenotazioneDao::doRetrievePrenotazioneByCapsulaAndDataInizioAndDataFine(capsId: Integer, dIn: String, dFi: String) **pre**: capsId <> null and dIn <> null and dFi <> null **post**: result->forAll(p | p.capsulaId = capsId and p.dataInizio >= dIn and p.dataFine <= dFi)
doRetrievePrenotazioniByAccountAndDataInizioAndDataFine	Metodo per ricavare le prenotazioni in base all'account e dalla data di inizio alla data di fine. **context** PrenotazioneDao::doRetrievePrenotazioniByAccountAndDataInizioAndDataFine(dIn: String, email: String, dFi: String) **pre**: dIn <> null and e-mail <> null and dFi <> null **post**: result->forAll(p | p.userAccountEmail = email and p.dataInizio >= dIn and p.dataFine <= dFi)