

Object Design Document

Version 1.1

Top Manager

Prof. Andrea De Lucia

Project Manager

Giuseppe Garreffa

Sommario

[Introduzione 5](#_Toc61987553)

[Trade-offs di Object Design 5](#_Toc61987554)

[Funzionalità vs Usabilità 5](#_Toc61987555)

[Sviluppo rapido vs Funzionalità 5](#_Toc61987556)

[Manutenibilità vs Performance 5](#_Toc61987557)

[Linee guida per la documentazione dell’interfaccia 5](#_Toc61987558)

[Definizioni, acronimi e abbreviazioni 5](#_Toc61987559)

[Packages 6](#_Toc61987560)

[view 6](#_Toc61987561)

[control.gestioneAutenticazione 6](#_Toc61987562)

[control.gestionePaziente 6](#_Toc61987563)

[control.gestioneMedico 7](#_Toc61987564)

[control.gestioneLaboratorio 7](#_Toc61987565)

[control.gestioneAsl 8](#_Toc61987566)

[model.entity 8](#_Toc61987567)

[model.dao 9](#_Toc61987568)

[Interfaccia delle classi 10](#_Toc61987569)

[Utente 10](#_Toc61987570)

[Paziente 11](#_Toc61987571)

[Medico 13](#_Toc61987572)

[Laboratorio 15](#_Toc61987573)

[Operatore 17](#_Toc61987574)

[Referto 18](#_Toc61987575)

[Prenotazione 19](#_Toc61987576)

[Prestazione 20](#_Toc61987577)

[UtenteManager 21](#_Toc61987578)

[PazienteManager 22](#_Toc61987579)

[MedicoManager 23](#_Toc61987580)

[RefertoManager 23](#_Toc61987581)

[PrestazioneManager 24](#_Toc61987582)

[Bridge Pattern 25](#_Toc61987583)

Partecipanti

|  |  |
| --- | --- |
| **Nome** | **Matricola** |
| Giuseppe Cirillo | 0512102644 |
| Giuseppe Garreffa | 0512103512 |
| Cristian Alfano | 0512102872 |
| Anna Santoro | 0512103458 |

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versione** | **Descrizione** | **Autore** |
| 05/12/2020 | 0.1 | Bozza | Cristian Alfano |
| 09/12/2020 | 0.2 | Inserimento classe Utente, UtenteManager, metodo di registrazione per PazienteManager, control.gestioneAutenticazione | Cristian Alfano |
| 10/12/2020 | 0.3 | Inserimento classe Laboratorio, classe Referto e manager, classe Prenotazione e manager | Giuseppe Garreffa |
| 13/12/2020 | 0.4 | Inserimento Classe Paziente e manager, Classe OperatoreASL e manager | Anna Santoro |
| 15/12/2020 | 0.5 | Inserimento Classe Medico e manager, Classe Prestazione e manager | Giuseppe Cirillo |
| 12/01/2021 | 0.6 | Modifica Classe Laboratorio inserimento attributo e metodi, modifica prestazione e prenotazione manager | Giuseppe Cirillo |
| 12/01/2021 | 0.7 | Inserimento Design Pattern | Cristian Alfano |
| 14/01/2021 | 1.0 | Revisione | Cristian Alfano  Giuseppe Cirillo |
| 18/01/2021 | 1.1 | Revisione generale e accettazione | Cristian Alfano  Giuseppe Garreffa |

**Introduzione**

Trade-offs di Object Design

### Funzionalità vs Usabilità

Si è deciso di preferire l’usabilità laddove fosse necessario scegliere tra le due categorie citate in quanto l’uso del sito deve essere intuitivo per l’utente finale e le funzionalità non implementate possono essere successivamente sviluppate e messe in funzione in maniera incrementale.

### Sviluppo rapido vs Funzionalità

Si è deciso di non implementare alcune funzionalità a causa delle scadenze fissate e di preferire lo sviluppo e il testing delle funzionalità ad alta priorità ed alcune di media priorità in modo da rendere fruibile il sistema almeno per i Pazienti e i Laboratori.

### Manutenibilità vs Performance

Si è deciso di preferire le performance per quanto riguarda la gestione dei dati persistenti. I file dei referti saranno memorizzati direttamente su disco rigido e i loro percorsi saranno memorizzati nella colonna “File” della tabella “Referto”.

Il Database è stato ristrutturato in modo da unire le categorie di utenza in un’unica tabella in modo da rendere la ricerca più veloce e di separare i recapiti e gli orari di attività in tabelle separate in quanto il potenziale numero di pazienti che accedono al Sistema rendendo non necessario l’accesso a queste informazioni è molto più elevato rispetto alle altre categorie.

Linee guida per la documentazione dell’interfaccia

Affinché il codice sia facilmente leggibile e manutenibile, si è deciso di adottare la notazione [CamelCase](https://en.wikipedia.org/wiki/Camel_case) in quanto è la più familiare per gli sviluppatori coinvolti.

Esempio di nome per una variabile: usernamePaziente

Esempio di nome per una classe: RefertoManager

È raccomandato l’uso della seguente forma d’uso delle parentesi graffe per la scrittura di metodi, classi, cicli e istruzioni di controllo:

NomeCampo {

// Codice

}

Il codice dovrà essere opportunamente indentato e l’ambiente di sviluppo Eclipse permette l’indentazione automatica delle istruzioni.

Definizioni, acronimi e abbreviazioni

|  |  |
| --- | --- |
| Nome | Descrizione |
| RandomControl | Le classi Control gestiscono le richieste HTTP e inviano le risposte al client |
| RandomManager | Le classi Manager gestiscono la Business Logic e l’interazione con i file e il database. |
| JSP | [JavaServer Pages](https://www.oracle.com/java/technologies/jspt.html) |

**Packages**

## view

|  |  |
| --- | --- |
| Nome | Descrizione |
| index.jsp | Home del sistema |
| paziente/homepaziente.jsp | Home del paziente |
| paziente/prenota.jsp | Pagina per le prenotazioni |
| laboratorio/homelaboratorio.jsp | Home del laboratorio |
| laboratorio/CaricaPrestazione.jsp | Pagina per l’inserimento di una nuova prestazione |
| laboratorio/InserisciReferto.jsp | Pagina per l’inserimento di un nuovo referto |
| registrazione1.jsp | Pagina per la prima parte di registrazione del Paziente |
| registrazione2.jsp | Pagina per la seconda parte di registrazione del Paziente |

## control.gestioneAutenticazione

|  |  |
| --- | --- |
| Nome | Descrizione |
| LoginControl | Gestisce il login |
| LogoutControl | Gestisce il logout |
| FirstFormControl | Gestisce il primo form della registrazione |
| SecondFormControl | Gestisce il secondo form della registrazione |

## control.gestionePaziente

|  |  |
| --- | --- |
| Nome | Descrizione |
| PrenotazioneControl | Gestisce i campi inseriti per la prenotazione e ne controlla il formato |
| ProvinceControl | Gestisce la provincia inserita e ne controlla il formato |
| DistrettoControl | Gestisce il distretto inserito e ne controlla il formato |
| DataControl | Gestisce la data inserita e ne controlla il formato |
| OrarioControl | Gestisce l’orario inserito e ne controlla il formato |
| PrenotazioneControl | Gestisce la prenotazione effettuata e ne controlla il formato |
| MedicoControl | Permette la visualizzazione del medico |
| updateControl | Gestisce i dati da modificare e ne controlla il formato |
| RicercaPrenotazioneControl | Gestisce i campi inseriti e ne effettua un controllo |
| RicercaRefertoControl | Gestisce i campi inseriti e ne effettua il controllo |
| RefertoControl | Effettua un controllo sul referto |

## control.gestioneMedico

|  |  |
| --- | --- |
| Nome | Descrizione |
| buttonModificaControl | Abilita la modifica dei dati di un medico |
| updateControl | Gestisce la procedura di modifica dei dati con i relativi controlli e messaggi all’utente |
| pazienteFormControl | Gestisce la lista di pazienti |
| ricercaFormControl | Gestisce la ricerca |
| refertoButtonControl | Gestisce i referti di un paziente |
| RefertiTableControl | Gestisce la ricerca dei referti di un paziente |
| PazientiControl | Gestisce la procedura di richiesta dei pazienti |

## control.gestioneLaboratorio

|  |  |
| --- | --- |
| Nome | Descrizione |
| refertoControl | Gestisce la procedura di richiesta dei referti |
| caricamentoControl | Gestisce la procedura di caricamento di un nuovo referto con i relativi controlli e messaggi all’utente |
| searchControl | Gestisce la ricerca dei referti |
| modificaControl | Gestisce la procedura di richiesta dei referti durante la fase di modifica |
| updateControl | Gestisce la procedura di modifica di un referto con i relativi controlli e messaggi all’utente |
| deleteControl | Gestisce la procedura di cancellazione di un referto con i relativi controlli e messaggi all’utente |
| prestazioniControl | Gestisce la procedura di richiesta delle prestazioni o dell’invio dei dati al Model |
| editControl | Gestisce la procedura di modifica di un referto con i relativi controlli e messaggi all’utente |

## control.gestioneAsl

|  |  |
| --- | --- |
| Nome | Descrizione |
| InsertOperatoreControl | Gestisce l’inserimento di un personale medico o laboratorio |
| TabMedicoControl | Gestisce la richiesta e la visualizzazione dei medici presenti sulla piattaforma |
| TabRiassuntoControl | Gestisce la richiesta e visualizzazione dei medici e pazienti scelti |
| TabPazienteControl | Gestisce la richiesta e la visualizzazione dei pazienti presenti sulla piattaforma |
| collegamentoControl | Gestisce il collegamento tra medico e pazienti scelti |
| personaleControl | Gestisce la visualizzazione e il controllo del personale |
| ricercaControl | Gestisce la ricerca del personale |

## model.entity

|  |  |
| --- | --- |
| Nome | Descrizione |
| Utente | Rappresenta un utente |
| Paziente | Rappresenta un paziente |
| Laboratorio | Rappresenta un laboratorio |
| Medico | Rappresenta un medico |
| Operatore | Rappresenta il gestore dei medici e dei laboratori |
| Referto | Rappresenta il referto di un paziente |
| Prenotazione | Rappresenta la prenotazione di un paziente |
| Prestazione | Rappresenta la prestazione offerta da quel determinato laboratorio |

## model.dao

|  |  |
| --- | --- |
| Nome | Descrizione |
| UtenteManager | Modella specifiche interazioni degli utenti con il DataBase |
| PazienteManager | Modella le interazioni del Paziente con il DataBase |
| MedicoManager | Modella le interazioni del Medico con il DataBase |
| RefertoManager | Modella le interazioni dei referti con il DataBase |
| PrestazioneManager | Modella le interazioni del prestazioni con il DataBase |

**Interfaccia delle classi**

Utente

|  |  |
| --- | --- |
| Nome | Utente |
| Descrizione | Rappresenta un utente registrato nel sistema |
| Attributi | -username: String  -password: String  -email: String  -cellulare: String  -telefono: String  -ruolo: enum |
| Signature dei metodi | +getUsername()  +setUsername(String username)  +getPassword()  +setPassword(String password)  +getEmail()  +setEmail(String email)  +getRuolo()  +getCellulare()  +setCellulare(String cellulare)  +getTelefono()  +setTelefono(String telefono) |
| Pre-condizioni | **context** Utente::setPassword(p) **pre**:  p.matches(‘^(?=.\*[0-9])(?=.\*[a-zA-Z])([a-zA-Z0-9]+){8,64}$’) |
| **context** Utente::setUsername(u) **pre**:  u.matches(‘^[A-Za-z0-9]{​​6,25}​​$’) |
| **context** Utente::setCellulare(c) **pre**:  c.matches(‘^[0-9]{8,16}$’) |
| **context** Utente::setTelefono(t) **pre**:  t.matches(‘^[0-9]{8,16}$’) |
| **context** Utente::setEmail(e) **pre**:  e.matches(‘\S+@\S+\.\S+’) |
| Post-condizioni |  |
| Invarianti |  |

## Paziente

|  |  |
| --- | --- |
| Nome | Paziente |
| Descrizione | Rappresenta un paziente registrato |
| Attributi | -Nome: String  -Cognome: String  -CF: String  -LuogoNascita: String  -DataNascita: Date  -Stato: String  -Città: String  -Provincia: String  -Cap: Intenger  -Indirizzo: String |
| Signature dei metodi | +getNome()  +setNome(String nome)  +getCognome()  +setCognome(String cognome)  +getCF()  +setCF(String cf)  +getLuognoNascita()  +setLuogoNascita(String luogoNascita)  +getDataNascita()  +setDataNascita(Date dataNascita)  +getStato  +setStato(String stato)  +getCittà()  +setCittà(String città)  +getProvincia()  +setProvincia(String provincia)  +getCap()  +setCap(int cap)  +getIndirizzo()  +setIndirizzo(String indirizzo) |
| Pre-condizioni | **context** Paziente::setNome(n) **pre**:  n.matches(‘^[A-Za-z ]+$’) |
| **context** Paziente:: setCognome(c) **pre**:  c.matches(‘^[A-Za-z ]+$’) |
| **context** Paziente:: setCF(cf) **pre**:  cf.length = 16 and cf.matches(‘^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$’) |
| **context** Paziente:: setLuogoNascita(l) **pre**:  l.matches(‘^[A-Za-z ]+$’) |
| **context** Paziente:: setDataNascita(d) **pre**:  d.length > 8 and d.length < 16 and d.matches(‘^(0?[1-9]|[12][0-9]|3[01])[/](0?[1-9]|1[012])[/]\d{4}$’) and d < now() |
| **context** Paziente:: setStato(s) **pre:**  s.matches(‘^[A-Za-z- ]+$’) |
| **context** Paziente:: setCittà(t) **pre**:  t.matches(‘^[A-Za-z- ]+$’) |
| **context** Paziente:: setProvincia(p) **pre**:  t.matches(‘^[A-Za-z- ]+$’) |
| **context** Paziente:: setCap(a) **pre**:  a.length = 5 and a.matches(‘^[0-9]+$’) |
| **context** Paziente:: setIndirizzo(i) **pre**:  i.matches(‘^[a-zA-Z0-9-, ]\*$’) |
| Post-condizioni |  |
| Invarianti |  |

## Medico

|  |  |
| --- | --- |
| Nome | Medico |
| Descrizione | Rappresenta un Medico registrato al sistema |
| Attributi | * CF: String * denominazione * città: String * provincia: String * cap: Integer * indirizzo: String * orariApertura: Time orariChiusura: Time * giorniApertura: Date * giorniChiusura: Date |
| Signature dei metodi | * getCF() * setCF(string CF) * getDenominazione() * setDenominazione(String denominazione) * getCitta() * setCitta(String citta) * getProvincia() * setProvincia(String provincia) * getCap() * setCap(int cap) * getOrariApertura() * setOrariApertura(List<Time> orari) * getOrariChiusura() * setOrariChiusura(List<Time> orari) * getGiorniApertura() * setGiorniApertura(List<Date> giorni) * getGiorniChiusura() * setGiorniChiusura(List<Time> giorni) * getIndirizzo() * setIndirizzo(String indirizzo) |
| Pre-condizioni | **context** Medico:: setDenominazione(d) **pre**:  d.matches(‘^[A-Za-z]+$’) |
| **context** Medico:: setCF(cf) **pre**:  cf.length = 16 and cf.matches(‘^[a-zA-Z]{6}[0-9]{2}[abcdehlmprstABCDEHLMPRST]{1}[0-9]{2}([a-zA-Z]{1}[0-9]{3})[a-zA-Z]{1}$’) |
| **context** Medico:: setCitta(c) **pre**:  c.matches(‘^[A-Za-z]+$’) |
| **context** Medico:: setProvincia(p) **pre**:  p.matches(‘^[A-Za-z]+$’) |
| **context** Medico:: setCap(ca) **pre**:  ca.matches(‘^[0-9]{5}$’) |
| **context** Paziente:: setIndirizzo(i) **pre**:  i.matches(‘^[a-zA-Z0-9-, ]\*$’) |
| Post-condizioni |  |
| Invarianti |  |

## Laboratorio

|  |  |
| --- | --- |
| Nome | Laboratorio |
| Descrizione | Rappresenta un laboratorio che offre prestazioni mediche ed è registrato alla piattaforma |
| Attributi | * PIVA: Int * nome: String * città: String * provincia: String * cap: Int * indirizzo: String * orariApertura: Time * orariChiusura: Time * giorniApertura: Date * giorniChiusura: Date |
| Signature dei metodi | * getPIVA() * setPIVA(int PIVA) * getDenominazione() * setDenominazione(String denominazione) * getCitta() * setCitta(String citta) * getProvincia() * setProvincia(String provincia) * getCap() * setCap(int cap) * getOrariApertura() * setOrariApertura(List<Time> orari) * getOrariChiusura() * setOrariChiusura(List<Time> orari) * getGiorniApertura() * setGiorniApertura(List<Date> giorni) * getGiorniChiusura() * setGiorniChiusura(List<Time> giorni) * getIndirizzo() * setIndirizzo(String indirizzo) |
| Pre-condizioni | **context** Laboratorio:: setDenominazione(d) **pre**:  d.matches(‘^[A-Za-z]+$’) |
| **context** Laboratorio:: setPIVA(pi) **pre**:  pi.matches(‘^[0-9]{11}$’) |
| **context** Laboratorio:: setCitta(c) **pre**:  c.matches(‘^[A-Za-z]+$’) |
| **context** Laboratorio:: setProvincia(p) **pre**:  p.matches(‘^[A-Za-z]+$’) |
| **context** Laboratorio:: setCap(ca) **pre**:  ca.matches(‘^[0-9]{5}$’) |
| **context** Paziente:: setIndirizzo(i) **pre**:  i.matches(‘^[a-zA-Z0-9-, ]\*$’) |
| Post-condizioni |  |
| Invarianti |  |

## Operatore

|  |  |
| --- | --- |
| Nome | Operatore |
| Descrizione | Rappresenta il gestore dei medici e dei laboratori |
| Attributi | * nome: String * cognome: String * CF: String |
| Signature dei metodi | * getNome() * getCognome() * getCF() |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

## Referto

|  |  |
| --- | --- |
| Nome | Referto |
| Descrizione | Rappresenta il referto registrato sulla piattaforma di un Paziente |
| Attributi | * ID: String * note: String * file: File * Prestazione: String * Data : date * Paziente : String |
| Signature dei metodi | * getID() * setId(String ID) * getNote() * setNote(String nota) * getFile() * setFile(File file) * getPrestazione() * setPrestazione(String prestazione) * getPaziente() * setPaziente(String paziente) * getData() * setData(Date data) |
| Pre-condizioni | **Context** Referto:: setNote(n) **pre**: n.matches(‘^[A-Za-z0-9]+$’) |
| **Context** Referto:: setId(i) **pre**: i.matches (‘^[A-Za-z0-9]+$’) |
| Post-condizioni |  |
| Invarianti |  |

## Prenotazione

|  |  |
| --- | --- |
| Nome | Prenotazione |
| Descrizione | Rappresenta la prenotazione che il paziente effettua |
| Attributi | -ID: int  -luogo: String  -dataPrenotazione: Date  -orario: Time  -stato: annullato, Referto |
| Signature dei metodi | +getID()  +setID(int id)  +getLuogo()  +setLuogo(String luogo)  +getDataPrenotazio()  +setDataPrenotazione(Date dataPrenotazione)  +getOrario()  +setOrario(Time orario)  +getStato() |
| Pre-condizioni | **Context** Prenotazione:: setId(i) **pre**: i.matches (‘^[0-9]+$’) |
| **Context** Prenotazione:: setLuogo(l) **pre**: l.matches (‘^[A-Za-z]+$’) |
|  | **Context** Prenotazione:: setDataPrenotazione(d) **pre**: d.matches(‘[^(0?[1-9]|[12][0-9]|3[01])[\/\-](0?[1-9]|1[012])[\/\-]\d{4}$]’) |
|  | **Context** Prenotazione:: setOrario(o) **pre**: o.matches(‘^([0-9]|0[0-9]|1[0-9]|2[0-3]):[0-5][0-9]$’) |
| Post-condizioni |  |
| Invarianti |  |

## Prestazione

|  |  |
| --- | --- |
| Nome | Prestazione |
| Descrizione | Rappresenta la prestazione |
| Attributi | -codice: String  -descrizione: String  -prezzo: Float |
| Signature dei metodi | +getCodice()  +setCodice(String codice)  +getDescrizione()  +setDescrizione(String descrizione)  +getPrezzo()  +setPrezzo(float prezzo) |
| Pre-condizioni | **Context** Prestazione:: setCodice(c) **pre**: c.matches (‘^[A-Za-z0-9]+$’) |
| **Context** Prestazione:: setDescrizione(d) **pre**: d.matches (‘^[A-Za-z]+$’) |
| **Context** Prestazione:: setPrezzo(p) **pre**: p.lenght = 6 and p.matches(‘^[0-9]+$’) |
| Post-condizioni |  |
| Invarianti |  |

UtenteManager

|  |  |
| --- | --- |
| Nome | UtenteManager |
| Descrizione | Modella le interazioni dell’Utente col Database |
| Attributi | -ds: DataSource  -TABLE\_NAME: String |
| Signature dei metodi | +retrieve(String username, String password)  +containsIdentificativo(String identificativo) |
| Pre-condizioni |  |
| Post-condizioni |  |
| Invarianti |  |

## PazienteManager

|  |  |
| --- | --- |
| Nome | PazienteManager |
| Descrizione | Modella le interazioni del Paziente con il DataBase |
| Attributi | -ds: DataSource  -TABLE\_NAME: String |
| Signature dei metodi | +save(Paziente p)  +check(String username, String codiceFiscale) |
| Pre-condizioni | **context** PazienteManager::save(p) **pre**:  PazienteManager.check(p.getUsername(username), p.getPassword(password)) = false |
| Post-condizioni | **context** PazienteManager::save(p) **post**: p.check(p.getUsername(username), p.getPassword(password)) = true |
| Invarianti |  |

## MedicoManager

|  |  |
| --- | --- |
| Nome | MedicoManager |
| Descrizione | Modella le interazioni del Medico con il DataBase |
| Attributi | ds: DataSource  -TABLE\_NAME: String |
| Signature dei metodi | + getMedico(Paziente p) |
| Pre-condizioni | **context** MedicoManager::getMedico(p) **pre**: p != null |
| Post-condizioni |  |
| Invarianti |  |

## RefertoManager

|  |  |
| --- | --- |
| Nome | RefertoManager |
| Descrizione | Permette di effettuare la ricerca del referto |
| Attributi | -ds: DataSource  -TABLE\_NAME: String |
| Signature dei metodi | +save(Referto r, String usernameLaboratorio, String codPrestazione, String CF)  +retrieve(String codReferto)  +getRefertiPaziente(String usernamePaziente)  +getRefertiLaboratorio(String usernameLaboratorio)  +delete(String codReferto) |
| Pre-condizioni | **context** RefertoManager::save(r,usernameLaboratorio,codPrestazione, CF) **pre**: RefertoManager.retreive(r.getID) = null |
| **context** RefertoManager::delete(codReferto) **pre**: RefertoManager.retreive(codReferto) != null |
| Post-condizioni | **context** RefertoManager::save(r,usernameLaboratorio,codPrestazione, CF) **pre**: RefertoManager.retreive(r.getID) != null |
| **context** RefertoManager::delete(c) **post**: RefertoManager.retrieve(codReferto) = null |
| Invarianti |  |

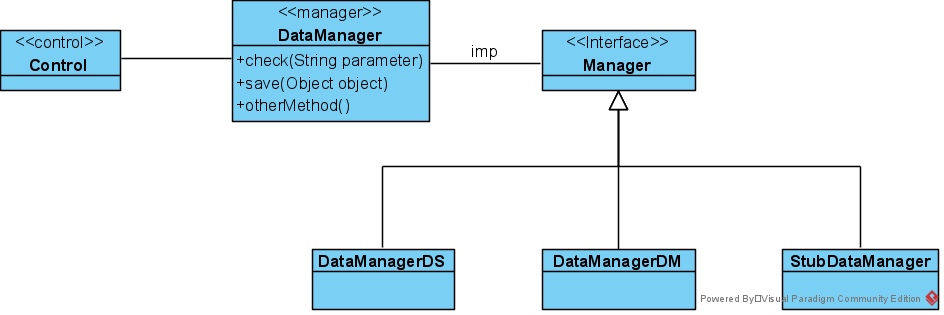
## PrestazioneManager

|  |  |
| --- | --- |
| Nome | PrestazioneManager |
| Descrizione | Gestisce le informazioni relative alle prestazioni |
| Attributi | -ds: DataSource  -TABLE\_NAME: String |
| Signature dei metodi | +check(String codPrestazione, String usernameLaboratorio)  +save(Prestazione prestazione)  +delete(String codPrestazione, String usernameLaboratorio)  +getPrestazioni() |
| Pre-condizioni | **context** PrestazioneManager::save(codPrestazione, usernameLaboratorio) **pre**: PrestazioneManager.check(codPrestazione) = null |
| **context** PrestazioneManager::delete(codPrestazione, usernameLaboratorio) **pre**: PrestazioneManager.check(codPrestazione) != null |
| Post-condizioni | **context** PrestazioneManager::save(codPrestazione, usernameLaboratorio) **post**: PrestazioneManager.check(codPrestazione) != null |
| **context** PrestazioneManager::delete(codPrestazione, usernameLaboratorio) **post**: PrestazioneManager.check(codPrestazione) = null |
| Invarianti |  |

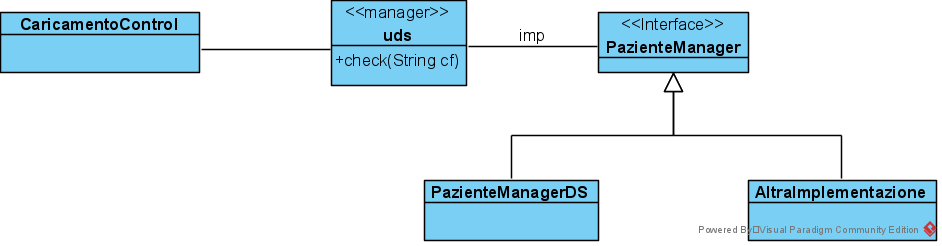
**Design Patterns**

## Bridge Pattern

Il Bridge Pattern permette di separare l’interfaccia di una classe dalla sua implementazione. Risulta particolarmente utile l’utilizzo di questo Design Pattern anche se si predilige l’utilizzo del solo DataSource come implementazione in quanto si potrebbero voler recuperare i dati da altri database o proprio accedere a sistemi che gestiscono la persistenza dei dati in maniera diversa.



Un possibile esempio riferito al progetto è questo:



*CaricamentoControl* invia un messaggio a *uds* chiedendo di verificare la presenza del Paziente nel sistema attraverso il codice fiscale. Si può dunque decidere di usare un’implementazione piuttosto che un’altra, in particolare si è deciso di usare l’implementazione con DataSource in modo che la factory di connessioni sia gestita automaticamente.

Laddove fosse necessario usare sistemi legacy per il recupero dei dati, sarà possibile combinare l’implementazione con un Adapter design.