OBJECT DESIGN

DOCUMENT

Nefrapp

Riferimento	Nefrapp_RAD_Vers.1.8, Nefrapp_SDD_Vers.1.0
Versione	1.0
Data	11/112/2019
Destinatario	Professoressa F.Ferrucci
Proposto da	Eugenio Corbisiero, Sara Corrente, Silvio Di Martino, Antonio Donnarumma, Luca Esposito, Matteo Falco, Domenico Musone, Davide Benedetto Strianese.
Approvato da	Antonio Fasulo, Francesco Garofalo

Project Manager

Nome	Cognome	Matricola
Francesco	Garofalo	0522500615
Antonio	Fasulo	0522500627

Partecipanti

Nome	Cognome	Matricola
Eugenio	Corbisiero	0512105449
Sara	Corrente	0512105695
Silvio	Di Martino	0512105629
Antonio	Donnarumma	0512105083
Luca	Esposito	0512105123
Matteo	Falco	0512109201
Domenico	Musone	0512105689
Davide Benedetto	Strianese	0512105257

Revision History

Data	Versione	Descrizione	Autori
26/11/2019	0.1	Definizione punti salienti	Sara Corrente, Luca Esposito
26/11/2019	0.2	Introduzione	Luca Esposito
05/12/2019	0.3	Package Interface	Luca Esposito
05/12/2019	0.4	Package generale	Eugenio Corbisiero
07/12/2019	0.5	Package Gestioni	Sara Corrente, Domenico Musone
07/12/2019	0.6	Design Pattern	Sara Corrente
09/12/2019	0.7	Package Storage	Davide Benedetto Strianese
09/12/2019	0.8	Package Entity	Luca Esposito
11/12/2019	0.9	Revisione e controllo qualità	Domenico Musone
18/12/2019	1.0	Schemi Design Patterns	Sara Corrente
18/12/2019	1.1	Riformulazione trade-off	Domenico Musone
13/01/2020	1.2	Aggiornato Storage	Antonio Donnarumma
14/01/2020	1.3	Aggiornato Entity	Luca Esposito
14/01/2020	1.4	Aggiornati Design Patterns e package gestioni	Sara Corrente Domenico Musone Silvio Di Martino
14/01/2020	1.5	Interfacce delle classi	Sara Corrente Luca Esposito
14/01/2020	1.6	Revisione Totale	Luca Esposito

Sommario

1.Introduzione

1.1 Object design trade-offs

1.1.1 Accoppiamento vs leggibilità del codice

Qualora determinate operazioni diventassero fortemente ricorrenti in alcune porzioni di codice, sarà valutato l'incapsulamento di tali operazioni in componenti d'utilità. Queste incrementeranno il livello di accoppiamento del sistema, determinando dipendenze aggiuntive, ma garantiranno codice più leggibile nelle componenti che descrivono la logica del sistema.
Sarà inserito un package "Utility" comprendente unità che si occupano di crittografia, di controllo degli accessi alle funzionalità e di conversioni da oggetti prelevati dal database a oggetti utilizzabili nel codice. L'accoppiamento aggiuntivo risultante dall'aggiunta di questo package sarà preferito rispetto alle ripetizioni di codice e alle violazioni di responsibilità che si avrebbero se le operazioni citate fossero svolte direttamente nei package che ne necessitano.

1.1.2 Tempo di sviluppo vs sicurezza

Fin dalla prima messa in funzione, il sistema dovrà gestire dati sensibili. Ciò implica che codifiche, controlli di sicurezza e protocolli di autenticazione non potranno essere oggetto di compromessi nel caso si necessiti di effettuare tagli per rispettare le scadenze. L'attenzione verso la sicurezza del sistema prevarrà dunque sull'esigenza di svilupparne le funzionalità in tempi brevi.

1.2 Descrizione componenti off-the-shelf

Saranno usate componenti ***off-the-shelf***, ossia componenti software preesistenti disponibili sul mercato, che saranno adoperati per facilitare la creazione del progetto.
Per l'interfaccia grafica é previsto l'uso di ***Bootstrap***, un framework open source che contiene una raccolta di strumenti liberi per la creazione di siti e applicazioni per il Web. Questo contiene modelli di progettazione basati su HTML e CSS, per le varie componenti dell'interfaccia, come moduli, bottoni e navigazione, così come alcune estensioni opzionali di JavaScript.

È stato scelto questo template per la realizzazione e implementazione dell'interfaccia grafica, disponibile gratuitamente:

1- https://startbootstrap.com/themes/sb-admin-2/

Per effettuare il download e upload di file sul sistema, sarà utilizzata la libreria FileUpload di Apache. https://commons.apache.org/proper/commons-fileupload.

È infine previsto l'uso delle librerie comprese nel kit di sviluppo di Java SE 8, della libreria standard di tag JSP (JSTL), della libreria GSON per la gestione dei documenti in formato JSON all'interno del codice Java, e di MongoDB come DBMS, corredato dalla libreria necessaria per l'integrazione Java.

1.3 Linee guida per la documentazione dell'interfaccia

Per rendere il codice più estensibile e manutenibile, prima dell'implementazione della logica del sistema è opportuno stabilire delle convenzioni da seguire nel corso dell'implementazione. È stato scelto di utilizzare Checkstyle per verificare se il codice è conforme alle regole di codifica.

Commenti

Si farà uso di commenti per aggiungere informazioni accessorie che facilitino la comprensione del codice, eventualmente giustificando decisioni particolari o determinate scelte algoritmiche. Il lavoro di documentazione sarà coadiuvato dall'utilizzo di Javadoc, con l'ausilio della funzione di autogenerazione di Eclipse (Generate Javadoc).

I metodi dovranno dunque essere sempre preceduti da opportuna documentazione, riportando l'obiettivo del metodo stesso e il nome/i dell'autore/i.

Dichiarazioni

Le dichiarazioni delle variabili dovranno essere posizionate all'inizio del blocco di codice in cui le si utilizza. Dichiarare le variabili solo al momento del loro primo uso può infatti ostacolare la comprensione del codice.

Indentazione

L'indentazione dovrà essere effettuata con TAB e, a prescindere dal linguaggio usato per la produzione del codice, ogni istruzione dovrà essere opportunamente indentata.

Es.

```
<html>
        <head>
        </head>

        <body>
        </body>
</html>
```

Parentesi

Si riporterà il blocco di istruzioni che seguono un IF, un FOR o un WHILE tra parentesi graffe, anche nel caso vi fosse una sola istruzione.

Script Javascript

Nel caso vi fosse la necessità di corredare le pagine con degli script in linguaggio Javascript per ampliarne le funzionalità, tali script saranno collocati in file separati per agevolare la documentazione e il riuso del codice.

Funzioni e oggetti Javascript dovranno essere preceduti da un commento in stile Javadoc che li descriva e li documenti.

1.4 Definizioni, acronimi e abbreviazioni

Acronimi	Descrizione
HTML	Linguaggio di mark-up per pagine web.
CSS	Linguaggio usato per definire la formattazione di pagine web.
GUI	Graphic User Interface.
ODD	Object Design Document.

1.5 Riferimenti

- Documentazione di progetto (Nefrapp_RAD_Vers.1.8, Nefrapp_SDD_Vers.1.0)
- Bruegge, Dutoit, Object-Oriented Software Engineering.

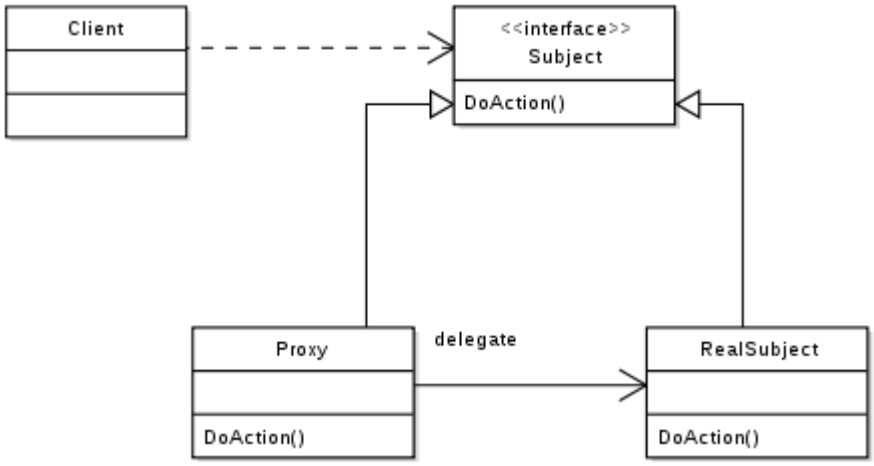
2. Design Pattern

2.1 Proxy

Questo pattern ha lo scopo di gestire la visualizzazione di messaggi e di annunci da parte degli utenti Medico e Paziente.

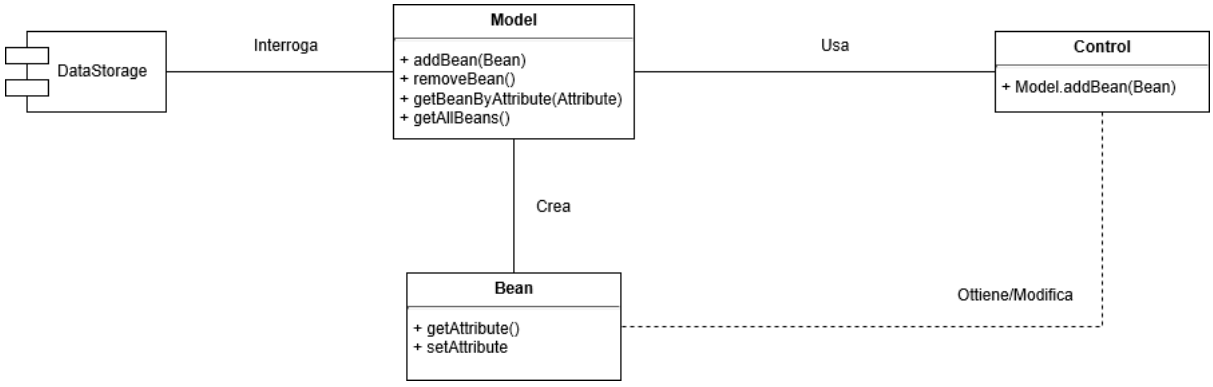
Il Proxy si compone di una interfaccia, chiamata Subject, che viene implementata dalla classe Proxy che agisce come un sostituto del Subject.. La classe Proxy mantiene una referenza all'oggetto sostituto (RealSubject) in modo tale da poter effettuare le operazioni richieste.

Il Proxy verrà utilizzato per la Gestione Messaggi e la Gestione Annunci..



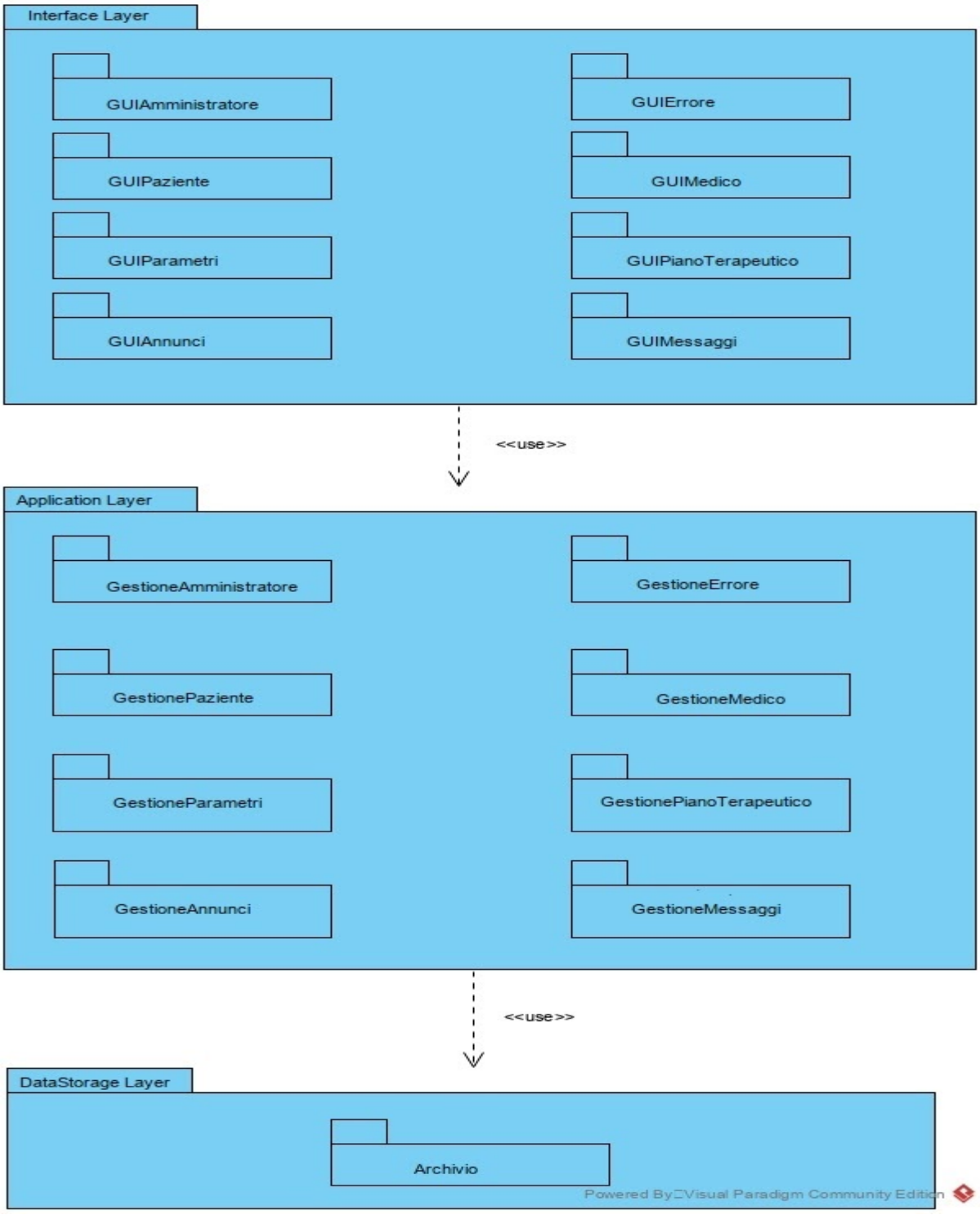
2.2 Data Access Object (DAO)

Questo pattern ha lo scopo di gestire la persistenza dei dati, stratificando e isolando l'accesso al Database tramite query, in questo modo viene fornita la possibilità di operare sui dati senza rivelare l'architettura del Database.


DataStorage
Interroga
Model
+ addBean(Bean)
+ removeBean()
+ getBeanByAttribute(Attribute)
+ getAllBeans()
Usa
Control
+ Model.addBean(Bean)
Crea
Bean
+ getAttribute()
+ setAttribute
Ottiene/Modifica

3. Packages

3.1 PK1 - Package Generale



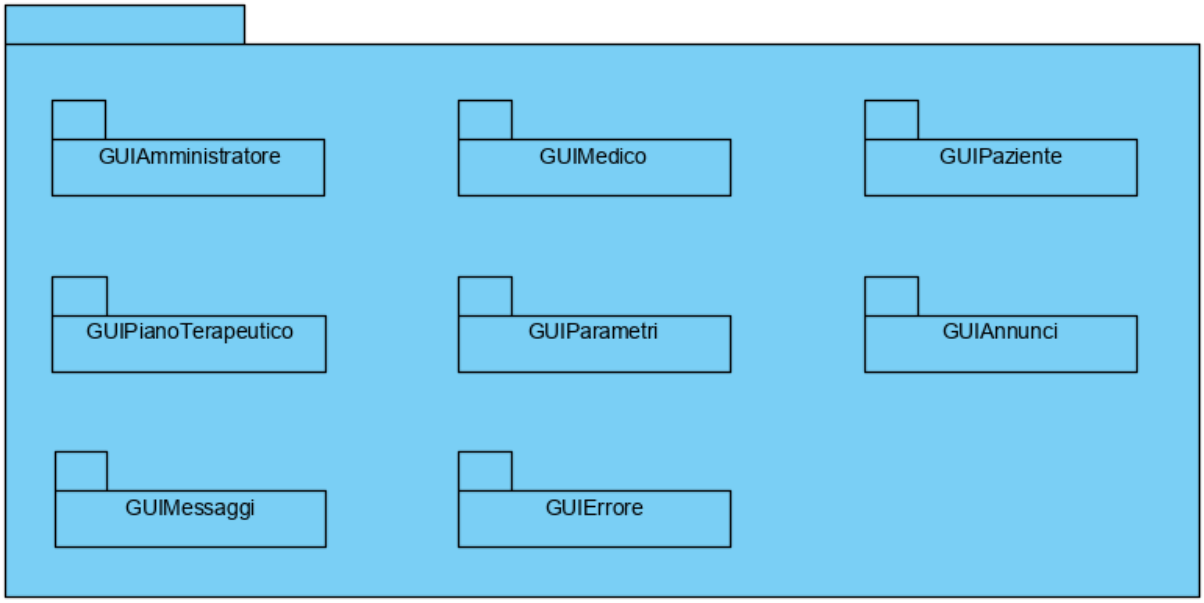
Il diagramma descrive la natura three-layer dell'applicazione mostrandone i tre package principali:

InterfaceLayer: GuiAmministratore, GuiMedico, GuiPaziente, GuiErrore , GuiPianoTerapeutico, GuiParametri, GuiAnnunnci , GuiMessaggi indicano i sottosistemi che contengono tutti gli oggetti boundary;

ApplicationLogicLayer: contiene i sottosistemi individuati :Gestione Amministratore, GestioneMedico, GestionePaziente, GestionePianoTerapeutico, GestioneMessaggi, GestioneAnnunci, GestioneParametri e Gestione Errori;

DataStorageLayer: Archivio, sottosistema che ha il compito di effettuare operazioni che riguardano il database.

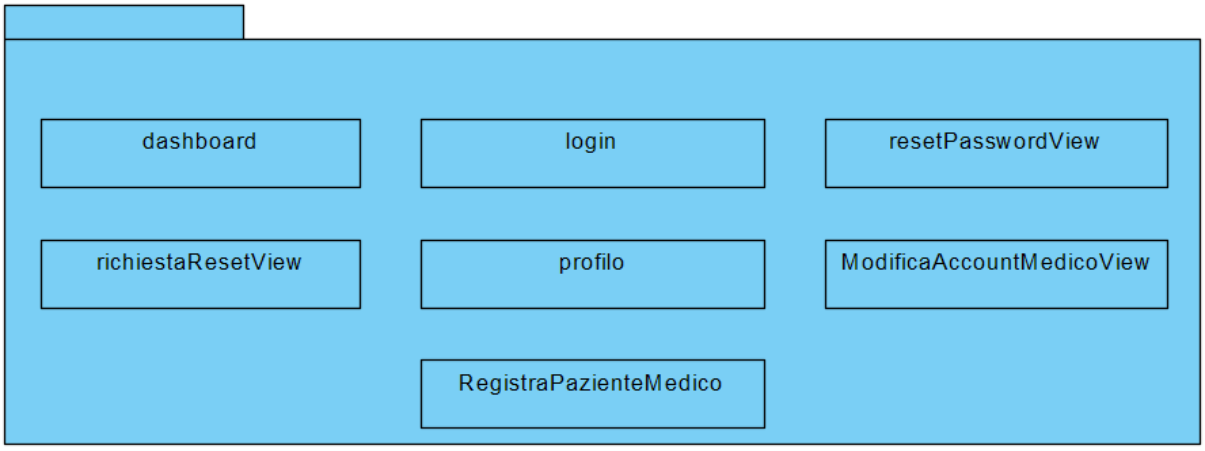
3.2 PK2 - Package Interface Generale



Il diagramma descrive le interfacce delle varie sezioni del sistema:

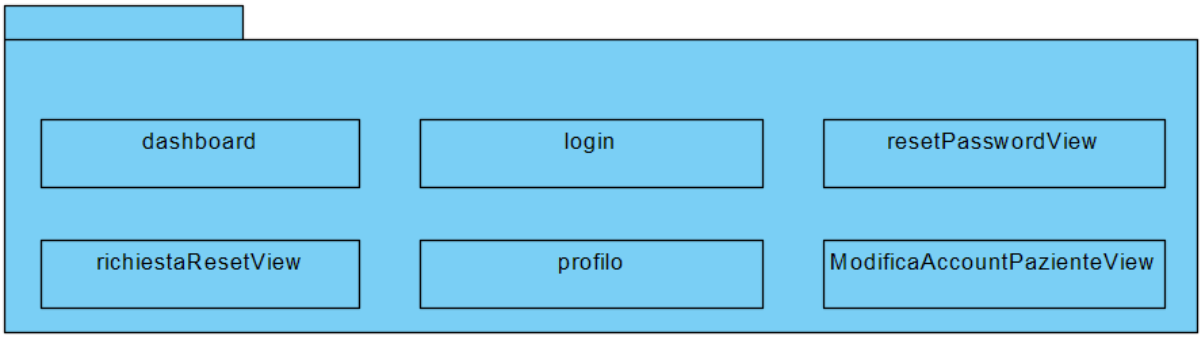
GUIAmministratore, GUIMedico, GUIPaziente, GUIPianoTerapeutico, GUIParametri, GUIAnnunci, GUIMessaggi.

3.2.1 PK2.1 - Package Interface GUIMedico



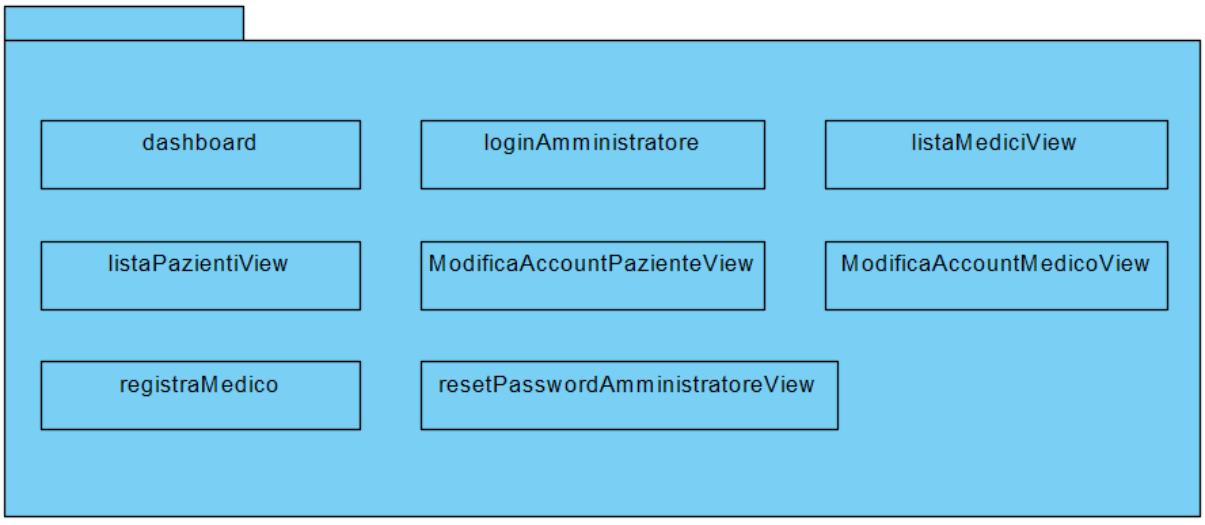
Il diagramma descrive le interfacce del sottosistema "GUIMedico".

3.2.2 PK2.2 - Package Interface GUIPaziente



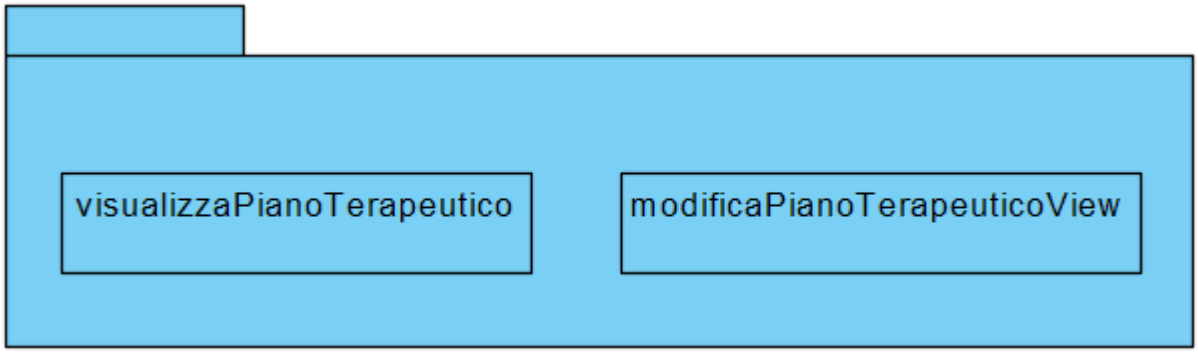
Il diagramma descrive le interfacce del sottosistema "GUIPaziente".

3.2.3 PK2.3 - Package Interface GUIAmministratore



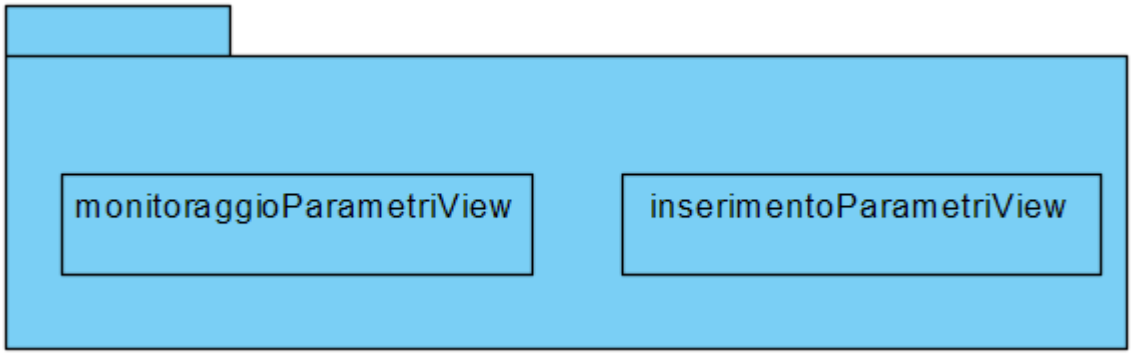
Il diagramma descrive le interfacce del sottosistema "GUIAmministratore".

3.2.4 PK2.4 - Package Interface GUIPianoTerapeutico



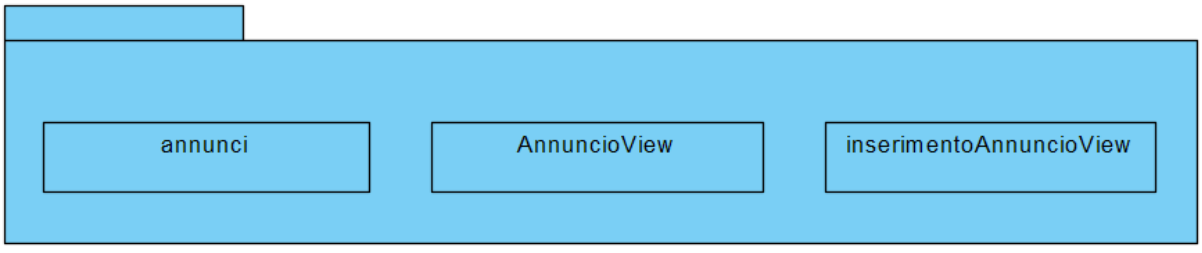
Il diagramma descrive le interfacce del sottosistema "GUIPianoTerapeutico".

3.2.5 PK2.5 - Package Interface GUIParametri



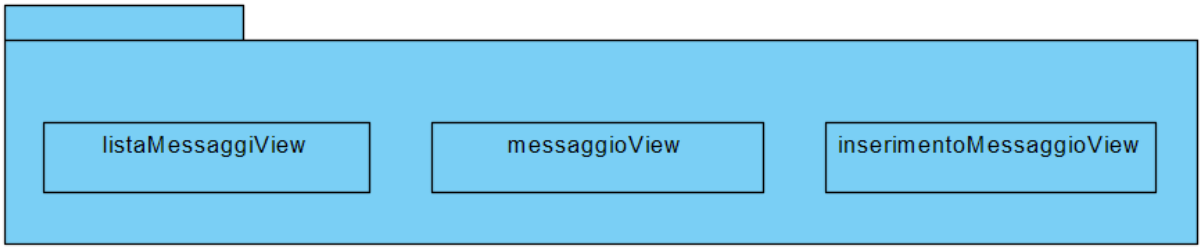
Il diagramma descrive le interfacce del sottosistema "GUIParametri".

3.2.6 PK2.6 - Package Interface GUIAnnunci



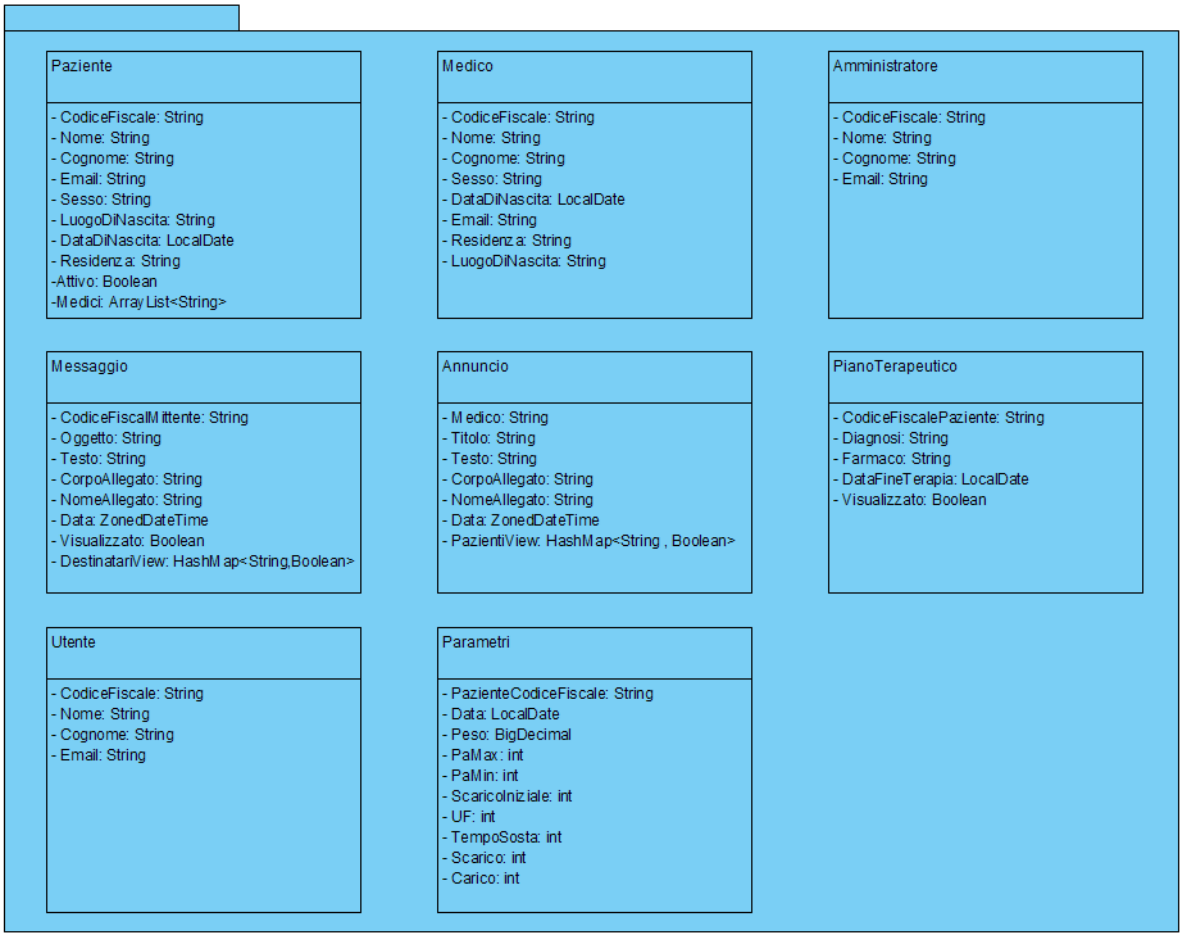
Il diagramma descrive le interfacce del sottosistema "GUIAnnunci".

3.2.7 PK2.7 - Package Interface GUIMessaggi



Il diagramma descrive le interfacce del sottosistema "GUIMessaggi".

3.3 PK3 - Entity



Il package principale contiene i package delle entity del nostro sistema. In esso sono definiti gli attributi di tali entity, corredati di tipo e visibilità.

3.3.1 Medico

Nome	Medico
Descrizione	Questa classe rappresenta il medico registrato al sito.
Signature dei metodi	+ getCodiceFiscale(): String + setCodiceFiscale(codiceFiscale: String): void + getNome(): String + setNome(nome: String): void + getCognome(): String + setCognome(cognome: String): void + getSesso(): String + setSesso(sesso: String): void + getDataDiNascita(): Date + setDataDiNascita(dataDiNascita: Date): void + getDataFormattata: String + getEmail(): String + setEmail(email: String): void + getResidenza(): String + setResidenza(residenza: String): void + getLuogoDiNascita(): String + setLuogoDiNascita(luogoDiNascita: String): void
Precondizione	**context**: Medico :: setCodiceFiscale(codiceFiscale) **pre:** codiceFiscale non deve avere altre corrispondenze nel database.
Post-condizione	**context**: Medico :: setCodiceFiscale(codiceFiscale) **post:** codiceFiscale è presente nel database.
Invariante	

3.3.2 Paziente

Nome	Paziente
Descrizione	Questa classe rappresenta il paziente registrato al sito.
Signature dei metodi	+ getCodiceFiscale(): String + setCodiceFiscale(codiceFiscale: String): void + getNome(): String + setNome(nome: String): void + getCognome(): String + setCognome(cognome: String): void + getSesso(): String + setSesso(sesso: String): void + getDataDiNascita(): Date + setDataDiNascita(dataDiNascita: Date): void + getDataFormattata(): String + getMedici(): ArrayList<String> + setMedici(medici: ArrayList<String>): void + addMedico(medicoCodiceFiscale: String): void + getEmail(): String + setEmail(email: String): void + getResidenza(): String + setResidenza(residenza: String): void + getLuogoDiNascita(): String + setLuogoDiNascita(luogoDiNascita: String): void + isAttivo(): Boolean + setAttivo(attivo: Boolean): void
Precondizione	**context**: Paziente :: setCodiceFiscale(codiceFiscale) **pre:** codiceFiscale non deve avere altre corrispondenze nel database.
Post-condizione	**context**: Paziente :: setCodiceFiscale(codiceFiscale) **post:** codiceFiscale è presente nel database.
Invariante	

3.3.3 Amministratore

Nome	Amministratore
Descrizione	Questa classe rappresenta l'amministratore registrato al sito.
Signature dei metodi	+ getCodiceFiscale(): String + setCodiceFiscale(codiceFiscale: String): void + getNome(): String + setNome(nome: String): void + getCognome(): String + setCognome(cognome: String): void + getEmail(): String + setEmail(email: String): void
Precondizione	
Post-condizione	
Invariante	

3.3.4 Messaggio

Nome	Messaggio
Descrizione	Questa classe rappresenta l'oggetto Messaggio.
Signature dei metodi	+ getCodiceFiscaleMittente(): String + setCodiceFiscaleMittente(codiceFiscaleMittente: String): void + getDestinatariView(): HashMap<String,Boolean> + setDestinatariView(destinatariView: HashMap<String,Boolean>): void + getOggetto(): String + setOggetto(oggetto: String): void + getTesto(): String + setTesto(testo: String): void + getCorpoAllegato(): String + setCorpoAllegato(corpoAllegato: String): void + getNomeAllegato(): String + setNomeAllegato(nomeAllegato: String): void + getData(): ZonedDateTime

	+ getDataFormattata(): String + getOraFormattata(): String + setData(data: ZonedDateTime): void + getIdMessaggio(): String + setIdMessaggio(idMessaggio: String): void + getVisualizzato(): Boolean + setVisualizzato(visualizzato: Boolean): void
Precondizione	**context**: Messaggio :: getDestinatariiView() **pre:** I destinatari sono presenti nel database.
Post-condizione	
Invariante	

3.3.5 Annuncio

Nome	Annuncio
Descrizione	Questa classe rappresenta l'oggetto Annuncio.
Signature dei metodi	+ getMedico(): String + setMedico(mittente: String): void + getPazientiView(): HashMap<String,Boolean> + setPazientiView(pazientiView: HashMap<String,Boolean>): void + getTitolo(): String + setTitolo(titolo: String): void + getTesto(): String + setTesto(testo: String): void + getCorpoAllegato(): String + setCorpoAllegato(corpoAllegato: String): void + getNomeAllegato(): String + setNomeAllegato(nomeAllegato: String): void + getData(): ZonedDateTime + setData(data: ZonedDateTime): void + getDataFormattata(): String + setIdAnnuncio(idAnnuncio: String): void + getIdAnnuncio(): String
Precondizione	**context**: Annuncio :: getPazientiView() **pre:** I pazienti sono presenti nel database.
Post-condizione	

Invariante	

3.3.6 PianoTerapeutico

Nome	PianoTerapeutico
Descrizione	Questa classe rappresenta l'oggetto Piano Terapeutico.
Signature dei metodi	+ getDiagnosi(): String + setDiagnosi(diagnosi: String): void + getCodiceFiscalePaziente(): String + setCodiceFiscalePaziente(codiceFiscalePaziente: String): void + getFarmaco(): String + setFarmaco(farmaco: String): void + getDataFineTerapia(): LocalDate + setDataFineTerapia(dataFineTerapia: LocalDate): void + getDataFormattata(): String + getVisualizzato(): Boolean + setVisualizzato(visualizzato: Boolean): void
Precondizione	
Post-condizione	
Invariante	

3.3.7 SchedaParametri

Nome	SchedaParametri
Descrizione	Questa classe rappresenta l'oggetto Parametri.
Signature dei metodi	+ getData(): LocalDate + setData(data: LocalDate): void + getDataFormattata(): String + getPazienteCodiceFiscale(): String + setPazienteCodiceFiscale(pazienteCodiceFiscale: String): void + getPeso(): BigDecimal + setPeso(peso: BigDecimal): void + getPaMax(): int

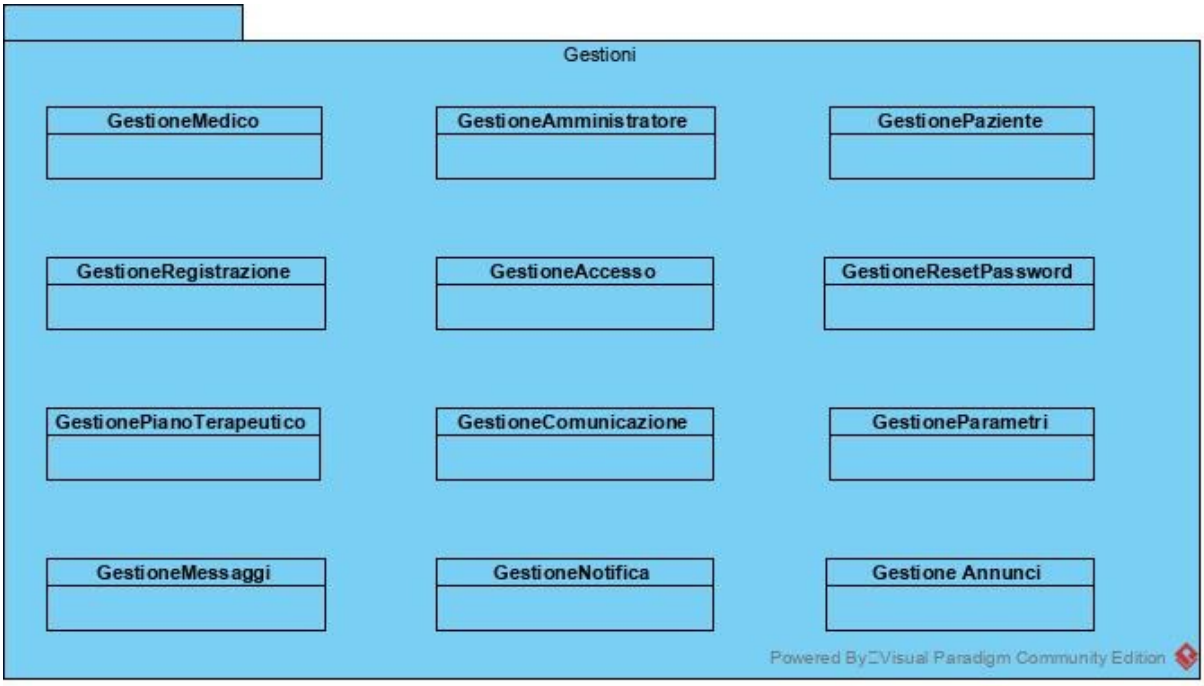
	+ setPaMax(paMax: int): void + getPaMin(): int + setPaMin(paMin: int): void + getScaricoIniziale(): int + setScaricoIniziale(scaricoIniziale: int): void + getUF(): int + setUF(uf: int): void + getTempoSosta(): int + setTempoSosta(tempoSosta: int): void + getScarico(): int + setScarico(scarico: int): void + getCarico(): int + setCarico(carico: int): void
Precondizione	
Post-condizione	
Invariante	

3.3.8 Utente

Nome	Utente
Descrizione	Questa superclasse rappresenta l'utente registrato al sito.
Signature dei metodi	+ getCodiceFiscale(): String + setCodiceFiscale(codiceFiscale: String): void + getNome(): String + setNome(nome: String): void + getCognome(): String + setCognome(cognome: String): void + getEmail(): String + setEmail(email: String): void
Precondizione	
Post-condizione	
Invariante	

3.4 PK4 - Gestioni

Il package "gestioni" si focalizza sulle classi che si occuperanno di implementare la logica del sistema (oggetti control presenti nel RAD).



3.4.1 Gestione Medico

Nome	GestioneMedico
Descrizione	Questa classe modella il gestore del sottosistema "Gestione medico" e ne implementa i servizi.
Signature dei metodi	-modificaAccount(request: HttpServletRequest, response: HttpServletResponse) : void -eliminaAccount(request: HttpServletRequest, response: HttpServletResponse): void -validazione(codiceFiscale: String, nome: String, cognome: String, sesso: String, email: String, residenza: String, luogoDiNascita: String, dataDiNascita: String, password: String, confermaPassword: String): boolean -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context**: gestioneMedico ::modificaAccount(request :

	HttpServletRequest, response : HttpServletResponse) **pre:** request.operazione == "modifica"; validazione (request.codiceFiscale, request.cognome, request.nome, request.sesso, request.email, request.residenza, request.luogoDiNascita, request.dataDiNascita, request.password, request.confermaPsw) == true; **context**: gestioneMedico ::eliminaAccount(request : HttpServletRequest, response : HttpServletResponse) **pre:** request.operazione == "eliminaAccount"; request.getSession().utente != null;
Post-condizione	**context**: gestioneMedico ::modificaAccount(request : HttpServletRequest, response : HttpServletResponse) **post:** I dati prelevati dalla request sono presenti nel database **context**: gestioneMedico ::eliminaAccount(request : HttpServletRequest, response : HttpServletResponse) **post:** request.getSession().utente == null;
Invariante	

3.4.2 Gestione Paziente

Nome	GestionePaziente
Descrizione	Questa classe modella il gestore del sottosistema "Gestione paziente" e ne implementa i servizi.
Signature dei metodi	- disattivaAccount(request: HttpServletRequest, response: HttpServletResponse) : void

	- modificaDatiPersonali(request: HttpServletRequest, response: HttpServletResponse) : void - caricaMedici(request: HttpServletRequest, response: HttpServletResponse): void - validazione(codiceFiscale: String, nome: String, cognome: String, sesso: String, email: String, residenza: String, luogoDiNascita: String, dataDiNascita: String, password: String, confermaPassword: String): boolean -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestionePaziente::disattivaAccount(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "disattivaAccount" && request.getSession().utente.attivo == true **context:** gestionePaziente::modificaDatiPersonali(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "modificaAccount" && request.getSession().utente.attivo == true && validazione (request.codiceFiscale, request.cognome, request.nome, request.sesso, request.email, request.residenza, request.luogoDiNascita, request.dataDiNascita, request.password, request.confermaPsw) == true;
Post-condizione	**context:** gestionePaziente::disattivaAccount(request: HttpServletRequest, response: HttpServletResponse) **post:** request.getSession().utente.attivo == false **context**: gestionePaziente ::modificaDatiPersonali(request:

	HttpServletRequest, response: HttpServletResponse) **post:** I dati prelevati dalla request sono presenti nel database
Invariante	

3.4.3 Gestione Amministratore

Nome	GestioneAmministratore
Descrizione	Questa classe modella il gestore del sottosistema "Gestione amministratore" e ne implementa i servizi.
Signature dei metodi	-scaricaDatiPazienteMedico(request: HttpServletRequest, response: HttpServletResponse): void -rimuoviAccount(codiceFiscale : String, tipo: String) : void - modificaDatiPersonali(request: HttpServletRequest, response: HttpServletResponse) : void - validazione(codiceFiscale: String, nome: String, cognome: String, sesso: String, email: String, residenza: String, luogoDiNascita: String, dataDiNascita: String, password: String, confermaPassword: String): boolean -validaPassword(vecchiaPassword: String, nuovaPassword: String, confermaPassword: String) -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context**: gestioneAmministratore::scaricaDatiPazienteMedico(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "caricaMedPaz" **context**: gestioneAmministratore::rimuoviAccount(codiceFiscale : String, tipo : String) **pre**: (tipo == "medico" || tipo ==

	"paziente") && la stringa codiceFiscale appartiene ad un paziente o un medico presente nel database **context**: gestioneAmministratore:: modificaDatiPersonali(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "modifica" && validaPassword(request.vecchiaPassword, request.nuovaPassword, request.confermaPassword) == true;
Post-condizione	**context**: gestioneAmministratore::scaricaDatiPaziente Medico(request: HttpServletRequest, response: HttpServletResponse) **post:** la response sottomessa dal metodo contiene i dati relativi a tutti i pazienti e tutti i medici **context**: gestioneAmministratore::rimuoviAccount(cod iceFiscale : String, tipo : String) **post**: il paziente o medico individuato dalla coppia tipo-codiceFiscale non è più presente nel database **context**: gestioneAmministratore:: modificaDatiPersonali(request: HttpServletRequest, response: HttpServletResponse) **post:** i dati prelevati dalla request sono presenti nel database
Invariante	

3.4.4 Gestione Piano Terapeutico

Nome	GestionePianoTerapeutico
Descrizione	Questa classe implementa i servizi di gestione del piano terapeutico.
Signature dei metodi	- visualizzaPiano(request: HttpServletRequest,

	response: HttpServletResponse) : void - modificaPiano(request: HttpServletRequest, response: HttpServletResponse) : void -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestionePianoTerapeutico::modificaPiano(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "modifica" && request.getSession().getAttribute("utente") è di tipo Medico && valida(request.data, request.codiceFiscale) == true **context:** gestionePianoTerapeutico::visualizzaPiano(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "visualizza" && request.CFPaziente contiene un codice fiscale appartenente a un paziente presente nel database
Post-condizione	**context:** gestionePianoTerapeutico::modificaPiano(request: HttpServletRequest, response: HttpServletResponse) **post:** i dati prelevati dalla request sono stati correttamente inseriti nel database **context:** gestionePianoTerapeutico::visualizzaPiano(request: HttpServletRequest, response: HttpServletResponse) **post:** request.pianoTerapeutico contiene un Piano Terapeutico presente nel database e legato al paziente avente CF uguale a request.CFPaziente
Invariante	

3.4.5 Gestione Parametri

Nome	GestioneParametri
Descrizione	Questa classe implementa i servizi di gestione dei parametri personali dei pazienti. Inoltre aggrega le schede parametri archiviate in report scaricabili dal medico.
Signature dei metodi	- inserisciParametri(request: HttpServletRequest, response: HttpServletResponse) : void - monitoraParametri(request: HttpServletRequest) : void - sonoValidi(cf: String, newPeso: BigDecimal , newPaMin: int, newPaMax: int, newScaricoIniziale: int, newUf: int, newTempoSosta: int, newScarico: int, newCarico: int): boolean - creaExcel(request: HttpServletRequest, response: HttpServletResponse):void -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestioneParametri::creaExcel(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione== “download” && request.CFPaziente contiene il codice fiscale di un paziente presente nel database **context:** gestioneParametri::inserisciParametri(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == ”inserisciScheda” && sonoValidi(request.cf. request.peso, request.paMin, request.paMax, request.scaricoIniziale, request.caricoIniziale, request.uf, request.tempoSosta, request.scarico, request.carico) == true; **context:** gestioneParametri::monitoraParametri(request

	: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "visualizzaScheda" && request.CFPaziente appartiene ad un paziente presente nel database
Post-condizione	**context:** gestioneParametri::creaExcel(request: HttpServletRequest, response: HttpServletResponse) **post:** la response contiene un file excel scaricabile contenente una lista degli inserimenti di parametri del paziente identificato da request.CFPaziente **context:** gestioneParametri::inserisciParametri(request: HttpServletRequest, response: HttpServletResponse) **post:** una nuova Scheda Parametri contenente i dati prelevati dalla request è presente nel database **context:** gestioneParametri::monitoraParametri(request : HttpServletRequest, response: HttpServletResponse) **post:** request.schedeParametri contiene una collezione di schede parametri presenti nel database, inserite dal paziente identificato da request.CFPaziente
Invariante	

3.4.6 Gestione Accesso

Nome	GestioneAccesso
Descrizione	Questa classe implementa i servizi di gestione dell'accesso al sito da parte di pazienti, medici e amministratori registrati.
Signature dei metodi	- loginUtente(request: HttpServletRequest, response: HttpServletResponse) : void - loginAmministratore(request:

	HttpServletRequest, response: HttpServletResponse) : void - logout(request: HttpServletRequest) : void - controllaParametri(codiceFiscale: String, password: String): boolean -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestioneAccesso::loginUtente(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "loginUtente" && controllaParametri(request.codiceFiscale, request.password) == true **context:** gestioneAccesso::loginAmministratore(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "loginAmministratore" && controllaParametri(request.codiceFiscale, request.password) == true **context:** gestioneAccesso::logout(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.getSession().accessDone == true
Post-condizione	**context:** gestioneAccesso::loginUtente(request: HttpServletRequest, response: HttpServletResponse) **post:** (request.getSession().isPaziente == true || request.getSession().isMedico == true) && request.getSession().utente != null && request.getSession().accessDone == true **context:** gestioneAccesso::loginAmministratore(request: HttpServletRequest, response: HttpServletResponse) **post:** request.getSession().isAmministratore == true && request.getSession().utente !=

	null && request.getSession().accessDone == true **context:** gestioneAccesso::logout(request: HttpServletRequest, response: HttpServletResponse) **post:** request.getSession()== null
Invariante	

3.4.7 Gestione Registrazione

Nome	GestioneRegistrazione
Descrizione	Questa classe implementa i servizi di gestione della registrazione al sito.
Signature dei metodi	- registraMedico(request: HttpServletRequest, response: HttpServletResponse) : void - registraPaziente(request: HttpServletRequest, response: HttpServletResponse) : void - validazione(codiceFiscale: String, nome: String, cognome: String, sesso: String, email: String, residenza: String, luogoDiNascita: String, dataDiNascita: String, password: String): boolean -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context**: gestioneRegistrazione:: registraMedico(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "registraMedico" && validazione(request.codiceFiscale, request.nome, request.cognome, request.sesso, request.email, request.password, request.residenza, request.luogoDiNascita, request.dataDiNascita) == true **context**: gestioneRegistrazione:: registraPaziente(request: HttpServletRequest, response: HttpServletResponse)

	pre: request.operazione == "registraPaziente" && validazione(request.codiceFiscale, request.nome, request.cognome, request.sesso, request.email, request.password, request.residenza, request.luogoDiNascita, request.dataDiNascita) == true
Post-condizione	**context**: gestioneRegistrazione:: registraMedico(request: HttpServletRequest, response: HttpServletResponse) **post:** il database contiene un nuovo medico registrato con i dati prelevati dalla request **context**: gestioneRegistrazione:: registraPaziente(request: HttpServletRequest, response: HttpServletResponse) **post:** il database contiene un nuovo paziente registrato con i dati prelevati dalla request
Invariante	

3.4.8 Gestione Messaggi

Nome	GestioneMessaggi
Descrizione	Questa classe implementa i servizi di gestione dei messaggi scambiati tra pazienti e medici.
Signature dei metodi	- visualizzaMessaggio(request: HttpServletRequest, response: HttpServletResponse) : void - visualizzaListaMessaggi(request: HttpServletRequest, response: HttpServletResponse) : void -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestioneMessaggi::visualizzaMessaggio(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "visualizzaMessaggio" && request.idMessaggio è l'ID di un messaggio

	presente nel database **context:** gestioneMessaggi::visualizzaListaMessaggi(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "visualizzaElencoMessaggio"
Post-condizione	**context:** gestioneMessaggi::visualizzaMessaggio(request: HttpServletRequest, response: HttpServletResponse) **post:** request.messaggio contiene le informazioni relative al messaggio referenziato nel database da request.idMessaggio **context:** gestioneMessaggi::visualizzaListaMessaggi(request: HttpServletRequest, response: HttpServletResponse) **post:** request.messaggi contiene la lista dei messaggi ricevuti dall'utente in sessione
Invariante	

3.4.9 Gestione Annunci

Nome	GestioneAnnunci
Descrizione	Questa classe implementa i servizi di gestione degli annunci inviati dai medici.
Signature dei metodi	- generaDownload(request: HttpServletRequest, response: HttpServletResponse): void - rimuoviAnnuncio(request: HttpServletRequest): void - visualizzaAnnunciPersonali(request: HttpServletRequest, response: HttpServletResponse) : void -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void

Precondizione	**context:** gestioneAnnunci::generaDownload(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "generaDownload" && request.id contiene l'ID di un annuncio presente nel database **context:** gestioneAnnunci::rimuoviAnnuncio(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "rimuoviAnnuncio" && request.id contiene l'ID di un annuncio presente nel database **context:** gestioneAnnunci::visualizzaAnnunciPersonali(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "visualizzaPersonali"
Post-condizione	**context:** gestioneAnnunci::generaDownload(request: HttpServletRequest, response: HttpServletResponse) **post:** response contiene il file allegato all'annuncio referenziato nel database da request.id **context:** gestioneAnnunci::rimuoviAnnuncio(request: HttpServletRequest, response: HttpServletResponse) **post:** il database non contiene più l'annuncio previamente referenziato dalla stringa request.id **context:** gestioneAnnunci::visualizzaAnnunciPersonali(request: HttpServletRequest, response: HttpServletResponse) **post:** request.annunci contiene la lista degli annunci visibili dall'utente loggato
Invariante	

3.4.10 Gestione Comunicazione

Nome	GestioneComunicazione
Descrizione	Questa classe implementa i servizi di gestione delle comunicazioni (messaggi e annunci) inviati da/a medici e pazienti.
Signature dei metodi	-caricaDestinatari(request: HttpServletRequest, response: HttpServletResponse): void -inviaComunicazione(request: HttpServletRequest, response: HttpServletResponse, operazione: String): void -caricaAllegato(request: HttpServletRequest,session: HttpSession, tipo:String): void -rimuoviAllegato(tipo:String, session: HttpSession): void -rimuoviIncompleta(tipo:String, session: HttpSession): void -controllaParametri(codiceFiscale: String, oggetto: String, testo: String): boolean -controllaFile(nomeFile: String, dimensioneFile: long): boolean -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestioneComunicazione::caricaDestinatari(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "caricaDestinatariMessaggio" **context:** gestioneComunicazione::inviaComunicazione(request: HttpServletRequest, response: HttpServletResponse, operazione: String) **pre:** request.operazione == "inviaMessaggio" && controllaParametri(request.CFMittente, request.oggetto, request.testo) == true

	context: gestioneComunicazione::caricaAllegato(request: HttpServletRequest, tipo:String, session:HttpSession) **pre:** request.operazione == "caricaAllegato" && request.file contiene l'allegato && tipo != null **context:** gestioneComunicazione::rimuoviAllegato(tipo:String, session:HttpSession) **pre:** request.operazione == "rimuoviAllegato" && tipo != null && session.id contiene l'id della comunicazione il cui allegato va rimosso **context:** gestioneComunicazione::rimuoviIncompleta (tipo:String, session:HttpSession) && tipo!=null **pre:** request.operazione == "rimuoviMessaggioIncompleto"
Post-condizione	**context:** gestioneComunicazione::caricaDestinatari(request: HttpServletRequest, response: HttpServletResponse) **post:** request.mediciCuranti contiene l'elenco di medici destinatari || request.pazientiSeguiti contiene l'elenco di pazienti destinatari **context:** gestioneComunicazione::inviaComunicazione(request: HttpServletRequest, response: HttpServletResponse, operazione: String) **post:** il database contiene una nuova comunicazione (di tipo Messaggio o Annuncio) con i dati prelevati dalla request **context:** gestioneComunicazione::caricaAllegato(request: HttpServletRequest, tipo:String, session:HttpSession) **post:** session.id contiene l'id della nuova comunicazione(di tipo Messaggio o Annuncio) && session.nomeFile contiene il

	nome del file && session.allegato contiene l'allegato **context:** gestioneComunicazione::rimuoviAllegato(tipo :String, session:HttpSession) **post:** request.getSession().allegato == null && request.getSession().nomeFile == null && request.getSession().id == null **context:** gestioneComunicazione::rimuoviIncompleta (tipo:String, session:HttpSession) && tipo!=null **post:** request.getSession().allegato == null && request.getSession().nomeFile == null && request.getSession().id == null
Invariante	

3.4.11 Gestione Notifiche

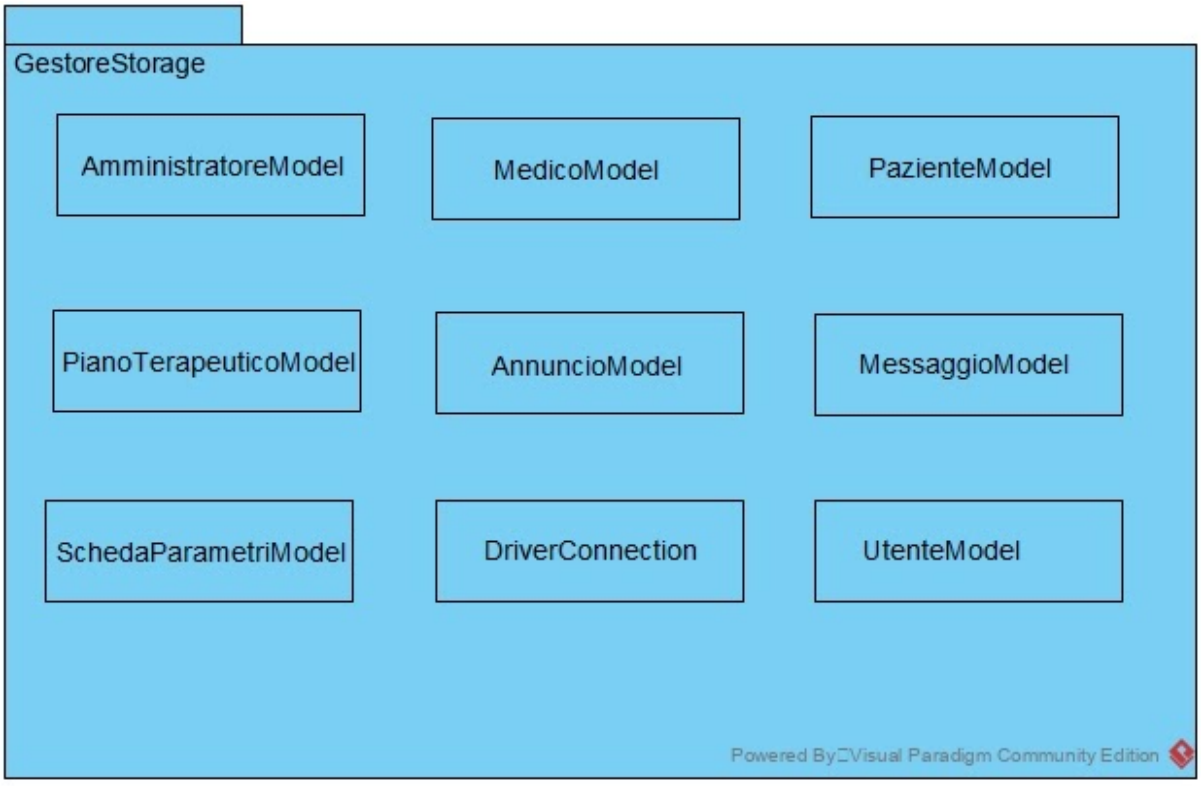
Nome	GestioneNotifiche
Descrizione	Questa classe implementa i servizi di gestione delle notifiche generate dall'invio di annunci e messaggi, e dall'aggiornamento del piano terapeutico di un paziente.
Signature dei metodi	-doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** **pre:**
Post-condizione	**context:** **post:**
Invariante	

3.4.12 Gestione Reset Password

Nome	GestioneResetPassword
Descrizione	Questa classe implementa i servizi di gestione per il reset della password richiesto da medici e pazienti
Signature dei metodi	-identificaRichiedente(request: HttpServletRequest, response: HttpServletResponse) -richiediReset(request: HttpServletRequest, response: HttpServletResponse): void -effettuaReset(request: HttpServletRequest, response: HttpServletResponse): void -validaMail(email: String): boolean -validaReset(email: String, codiceFiscale: String , password: String, confermaPsw: String) -doGet(request: HttpServletRequest, response: HttpServletResponse): void -doPost(request: HttpServletRequest, response: HttpServletResponse): void
Precondizione	**context:** gestioneResetPassword::identificaRichiedente(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "identificaRichiedente" && request.codiceFiscale contiene il codice fiscale dell'utente che richiede il reset password. **context:** gestioneResetPassword::effettuaReset(request: HttpServletRequest, response: HttpServletResponse) **pre:** request.operazione == "reset" && request.email contiene l'email dell'utente che ha effettuato la richiesta && request.codiceFiscale contiene il codice fiscale dell'utente che ha effettuato la richiesta && request.password contiene la password dell'utente che ha effettuato la richiesta && request.confermaPassword contiene la stessa

	stringa contenuta in password
Post-condizione	**context**: gestioneResetPassword::effettuaReset(request: HttpServletRequest, response: HttpServletResponse) **post:** viene effettuata la modifica della password nel database
Invariante	

3.5 PK5 - Storage



3.5.1 DriverConnection

Nome	DriverConnection
Descrizione	Questa classe rappresenta la gestione del pool di connessione con database.
Signature dei metodi	- getConnection(): MongoDatabase
Precondizione	
Post-condizione	

Invariante	

3.5.2 AmministratoreModel

Nome	AmministratoreModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti l'amministratore.
Signature dei metodi	+getAmministratoreByCFPassword(codiceFiscale : String, password : String): Amministratore + getPassword(codiceFiscale : String): String + updateAmministratore(daAggiornare : Amministratore): void + getAmministratoreByCF(codiceFiscale : String): Amministratore
Precondizione	**context:** AmministratoreModel :: getAmministratoreByCFPassword(codiceFiscale, password) **pre:** codiceFiscale != null && password != null **context:** AmministratoreModel :: updateAmministratore(daAggiornare) **pre:** daAggiornare!= null **context:** AmministratoreModel :: getPassword(codiceFiscale) **pre:** codiceFiscale!= null **context:** AmministratoreModel :: getAmministratoreByCF(codiceFiscale) **pre:** codiceFiscale!= null
Post-condizione	
Invariante	

3.5.3 MedicoModel

Nome	MedicoModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti il medico.
Signature dei metodi	+ getMedicoByCFPassword(codiceFiscale : String, password: String): Medico + addMedico(daAggiungere : Medico, password : String): void + getMedicoByCF(codiceFiscale: String): Medico + updateMedico(daAggiornare : Medico): void + removeMedico(daRimuovere: Medico): void + getAllMedici(): ArrayList<Medico> + getMedicoByCF(codiceFiscale : String): Medico +getMediciByPazienteSeguito(mediciPaziente: Paziente): ArrayList<Medico> + getMedicoByEmail(String email): Medico + updatePasswordMedico(codiceFiscale : String, nuovaPassword : String):void + checkEmail(email : String): boolean
Precondizione	**context:** MedicoModel :: getMedicoByCFPassword(codiceFiscale) **pre:** codiceFiscale!= null **context:** MedicoModel :: addMedico(daAggiungere) **pre:** daAggiungere!= null **context:** MedicoModel :: updateMedico(daAggiornare) **pre:** daAggiornare!= null **context:** MedicoModel :: removeMedico(daRimuovere) **pre:** daRimuovere!= null **context:** MedicoModel :: getMedicoByCF(codiceFiscale) **pre:** codiceFiscale!= null **context:** MedicoModel :: getMediciByPazienteSeguito(mediciPaziente) **pre:** mediciPaziente!= null **pre:** codiceFiscale!= null

	context: MedicoModel :: getMedicoByEmail(email) **pre:** email!= null **context:** MedicoModel :: updatePasswordMedico(codiceFiscale, nuovaPassword) **pre:** codiceFiscale!= null && nuovaPassword!= null **context:** MedicoModel :: checkEmail(email) **pre:** email!= null
Post-condizione	
Invariante	

3.5.4 PazienteModel

Nome	PazienteModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti il paziente.
Signature dei metodi	+getPazienteByCFPassword(codiceFiscale : String, password : String): Paziente + addPaziente(daAggiungere : Paziente, password : String): void + updatePaziente(daAggiornare : Paziente): void + removePaziente(daRimuovere : Paziente): void + getAllPazienti(): ArrayList<Paziente> + getPazienteByCF(codiceFiscale : String): Paziente + getPazientiSeguiti(codiceFiscaleMedico : String): ArrayList<Paziente> + getIdPazienteByCF(codiceFiscale : String): String + changePassword(daAggiornare : String. password : String): void
Precondizione	**context:** PazienteModel :: getPazienteByCFPassword(codiceFiscale, password) **pre:** codiceFiscale!= null && password != null **context:** PazienteModel :: addPaziente(daAggiungere) **pre:** daAggiungere!= null **context:** PazienteModel :: updatePaziente(daAggiornare) **pre:** daAggiornare!= null **context:** PazienteModel :: removePaziente(daRimuovere) **pre:** daRimuovere!= null **context:** PazienteModel :: getPazienteByCF(codiceFiscale) **pre:** codiceFiscale!= null **context:** PazienteModel :: getPazientiSeguiti(codiceFiscaleMedico) **pre:** codiceFiscaleMedico!= null

	context: PazienteModel :: getIdPazienteByCF(codiceFiscale) **pre**: codiceFiscale!= null **context**: PazienteModel :: changePassword(password) **pre**: password != null
Post-condizione	
Invariante	

3.5.5 PianoTerapeuticoModel

Nome	PianoterapeuticoModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti il piano terapeutico.
Signature dei metodi	+ addPianoTerapeutico(daAggiungere : PianoTerapeutico): void + updatePianoTerapeutico(daAggiornare: PianoTerapeutico): void +getPianoTerapeuticoByPaziente(codiceFiscalePaziente : String): PianoTerapeutico +isPianoTerapeuticoVisualizzato(codiceFiscalePaziente : String): boolean +setVisualizzatoPianoTerapeutico(codiceFiscalePaziente : String, visualizzato : Boolean): void +removePianoTerapeutico(codiceFiscalePaziente : String): void
Precondizione	**context:** PianoTerapeuticoModel :: addPianoTerapeutico(toAdd) **pre:** daAggiungeredaAggiungere!= null **context:** PianoTerapeuticoModel :: updatePianoTerapeutico(daAggiornare) **pre:** daAggiornare!= null **context:** PianoTerapeuticoModel :: getPianoTerapeuticoByPaziente(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null **context:** PianoTerapeuticoModel :: isPianoTerapeuticoVisualizzato(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null **context:** PianoTerapeuticoModel :: setVisualizzatoPianoTerapeutico(codiceFiscalePaziente, visualizzato) **pre:** codiceFiscalePaziente!= null && visualizzato!= null **context:** PianoTerapeuticoModel :: removePianoTerapeutico(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null

Post-condizione	
Invariante	

3.5.6 AnnuncioModel

Nome	AnnuncioModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti l'annuncio.
Signature dei metodi	+ addAnnuncio(daAggiungere: Annuncio): String + getAnnuncioById(idAnnuncio : String): Annuncio + updateAnnuncio(daAggiornare: Annuncio): void + deleteAnnuncioById(idAnnuncio : String): void +getAnnunciByCFPaziente(codiceFiscalePaziente : String): ArrayList<Annuncio> +getAnnunciByCFMedico(codiceFiscaleMedico : String): ArrayList<Annuncio> +countAnnunciNonLetti(codiceFiscalePaziente: String): int + setVisualizzatoAnnuncio(idAnnuncio : String, CFPaziente : String, visualizzato : Boolean): void
Precondizione	**context:** AnnuncioModel:: addAnnuncio(daAggiungere) **pre:** daAggiungere!= null **context**: AnnuncioModel:: getAnnuncioById(idAnnuncio) **pre:** idAnnuncio!= null **context**: AnnuncioModel:: updateAnnuncio(daAggiornare) **pre:** daAggiornare!= null **Context:** AnnuncioModel:: deleteAnnuncioById(idAnnuncio) **pre:** idAnnuncio!= null **Context:** AnnuncioModel:: getAnnunciByCFPaziente(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null **Context:** AnnuncioModel:: getAnnunciByCFMedico(codiceFiscaleMedico) **pre:** codiceFiscaleMedico!= null **pre:** codiceFiscalePaziente!= null

	Context: AnnuncioModel:: countAnnunciNonLetti(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null **Context:** AnnuncioModel:: setVisualizzatoAnnuncio(idAnnuncio, CFPaziente, visualizzato) **pre:** idAnnuncio!= null && CFPaziente!= null && visualizzato!= nul
Post-condizione	
Invariante	

3.5.7 MessaggioModel

Nome	MessaggioModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti il messaggio.
Signature dei metodi	+ addMessaggio(daAggiungere : Messaggio): String + deleteMessaggioById(idMessaggio : String):void +getMessaggiByDestinatario(CFDestinatario : String): ArrayList<Messaggio> + getMessaggioById(idMessaggio : String): Messaggio + updateMessaggio(daAggiornare : Messaggio): void + setVisualizzatoMessaggio(idMessaggio : String, CFDestinatario : String, visualizzato : Boolean): void + countMessaggiNonLetti(CFDestinatario : String): int
Precondizione	**context:** MessaggioModel:: addMessaggio(daAggiungere) **pre:** daAggiungere!= null **Context:** MessaggioModel:: deleteMessaggioById(idMessaggio) **pre:** idMessaggio!= null **Context:** MessaggioModel:: getMessaggiByDestinatario(CFDestinatario) **pre:** CFDestinatario!= null **Context:** MessaggioModel:: getMessaggioById(idMessaggio) **pre:** idMessaggio!= null **Context:** MessaggioModel:: updateMessaggio(daAggiornare) **pre:** daAggiornare!= null **Context:** MessaggioModel:: setVisualizzatoMessaggio(idMessaggio, CFDestinatario, visualizzato) **pre:** idMessaggio!= null && CFDestinatario!= null && visualizzato!= null **Context:** MessaggioModel:: countMessaggiNonLetti(CFDestinatario) **pre:** CFDestinatario!= null

Post-condizione	
Invariante	

3.5.8 SchedaParametriModel

Nome	SchedaParametriModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti la scheda dei parametri.
Signature dei metodi	+getSchedaParametriByCF(codiceFiscalePaziente : String): ArrayList<SchedaParametri> + addSchedaParametri(daAggiungere: SchedaParametri): void + getReportByPaziente(codiceFiscalePaziente: String, dataInizio: LocalDate, dataFine: LocalDate): ArrayList<SchedaParamertri>
Precondizione	**context:** SchedaParametriModel:: getSchedaParametriByCF(codiceFiscalePaziente) **pre:** codiceFiscalePaziente!= null **context:** SchedaParametriModel:: addSchedaParametri(daAggiungere) **pre:** daAggiungere!= null **Context:** SchedaParametriModel:: getReportByPaziente(codiceFiscalePaziente, dataInizio, dataFine) **pre:** codiceFiscalePaziente!= null && dataInizio!= null && dataFine!= null && dataInizio < dataFine
Post-condizione	
Invariante	

3.5.9 UtenteModel

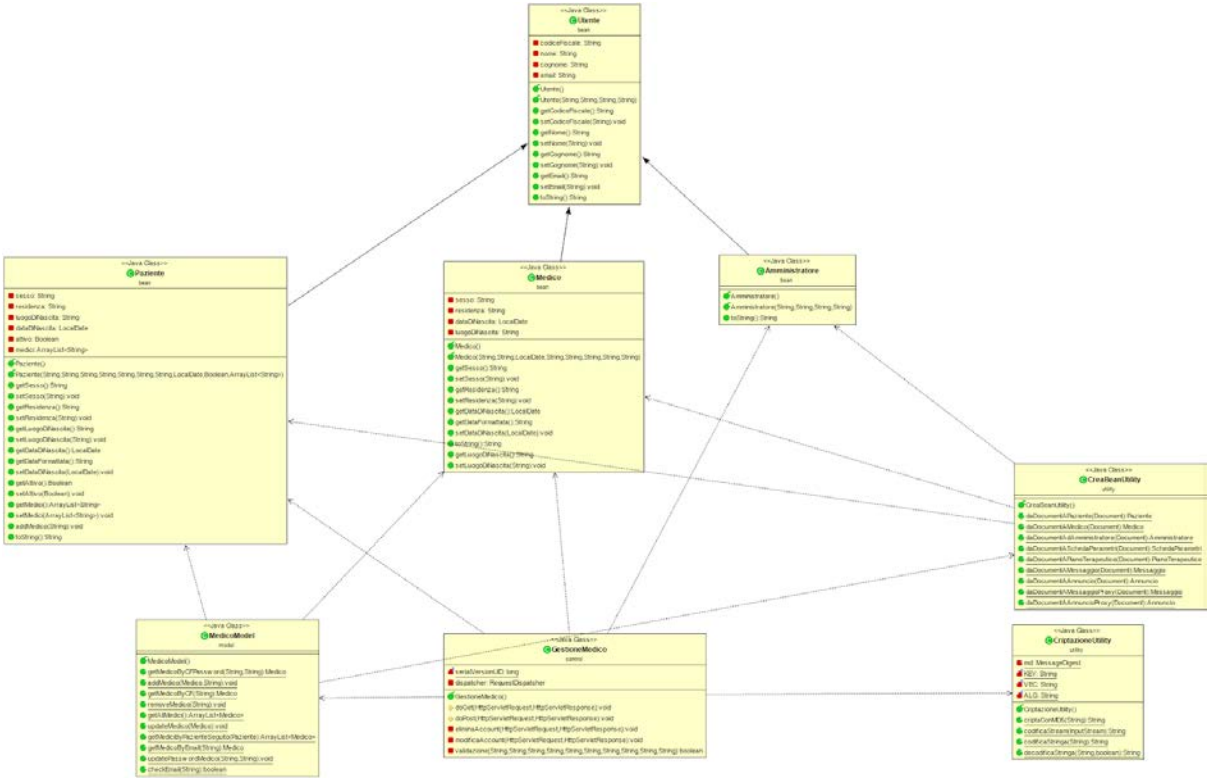
Nome	UtenteModel
Descrizione	Questa classe è un manager che si occupa di interagire con il database. Gestisce le query riguardanti la generalizzazione Utente.
Signature dei metodi	+ getUtenteByCF(codiceFiscale : String): Utente
Precondizione	**context:** UtenteModel:: getUtenteByCF(codiceFiscale) **pre:** codiceFiscale!= null

Post-condizione	
Invariante	

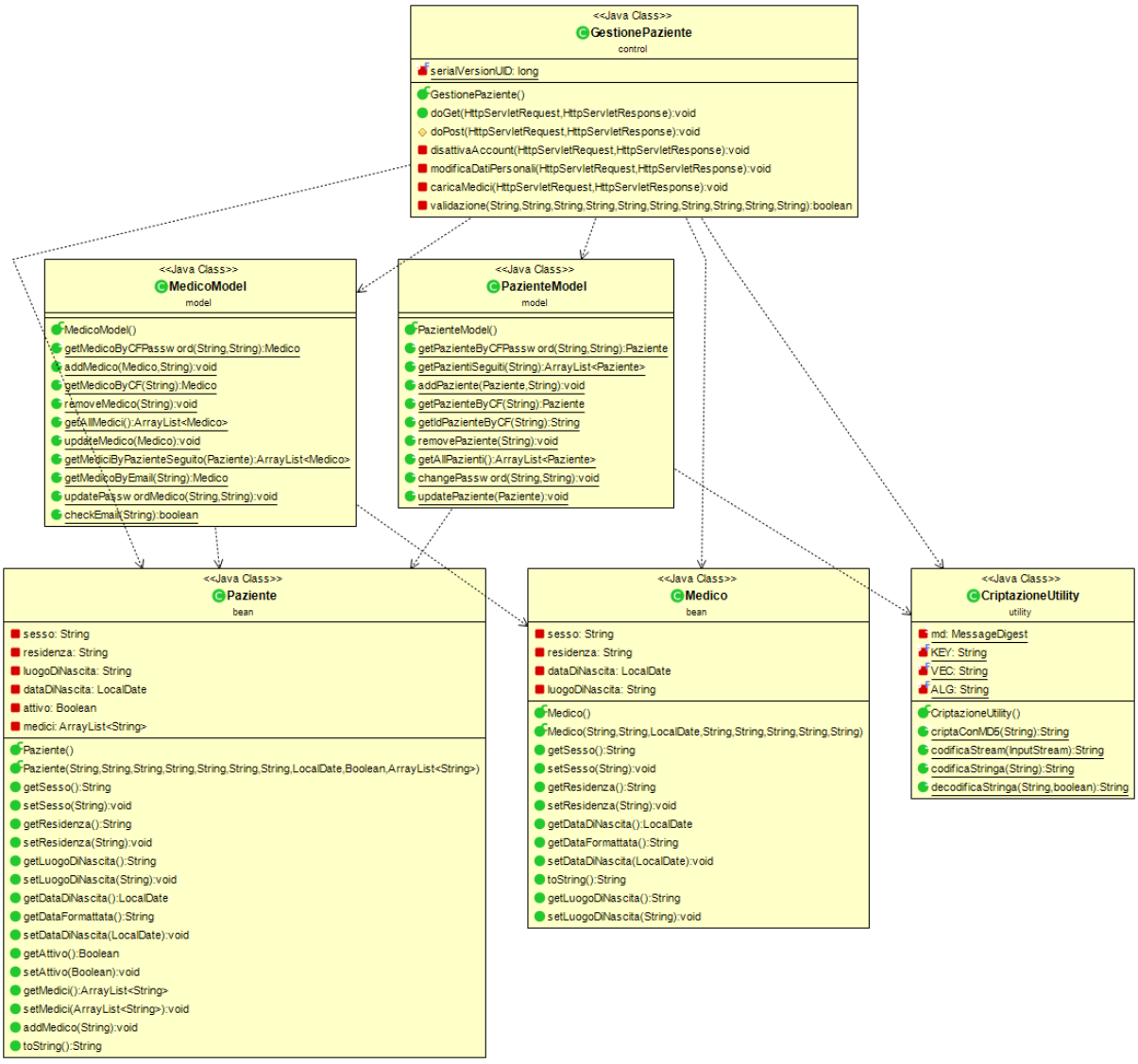
4. Interfacce delle classi

I seguenti diagrammi dei packages sono stati realizzati utilizzando ObjectAID, un plugin del framework di Eclipse, dopo aver completato lo sviluppo.

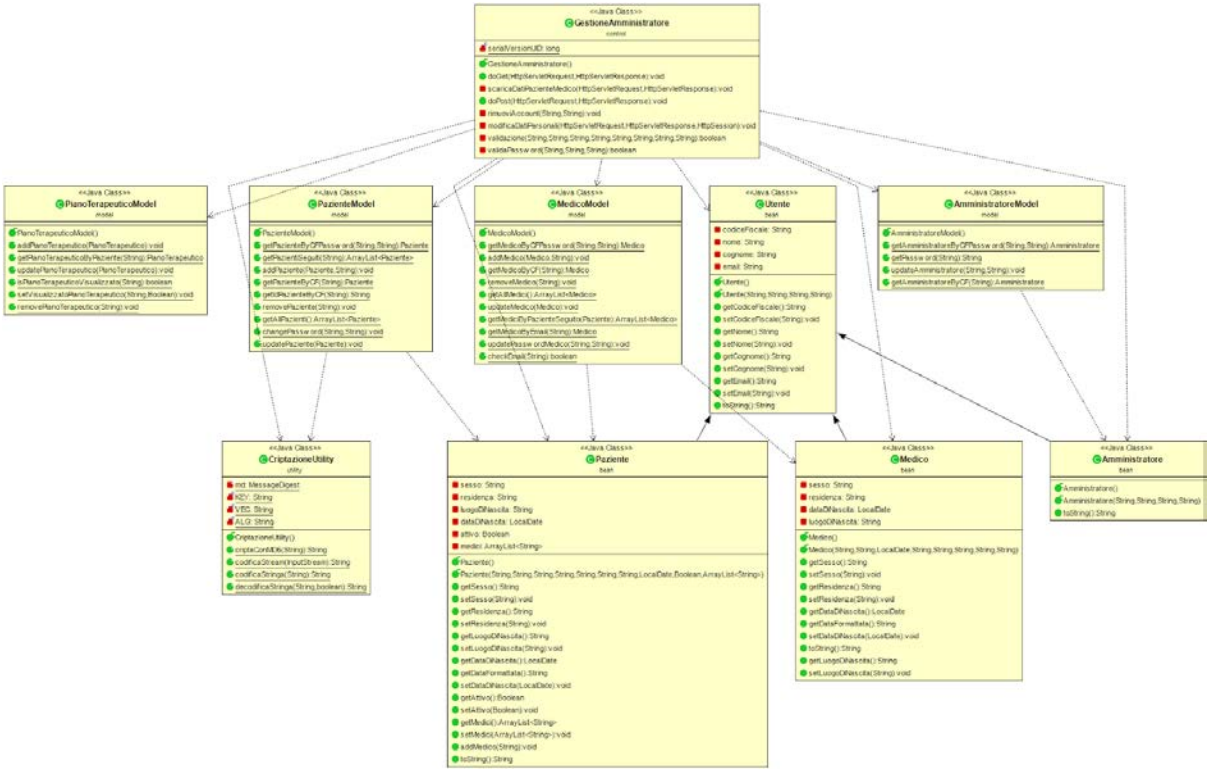
4.1 CD - Gestione Medico



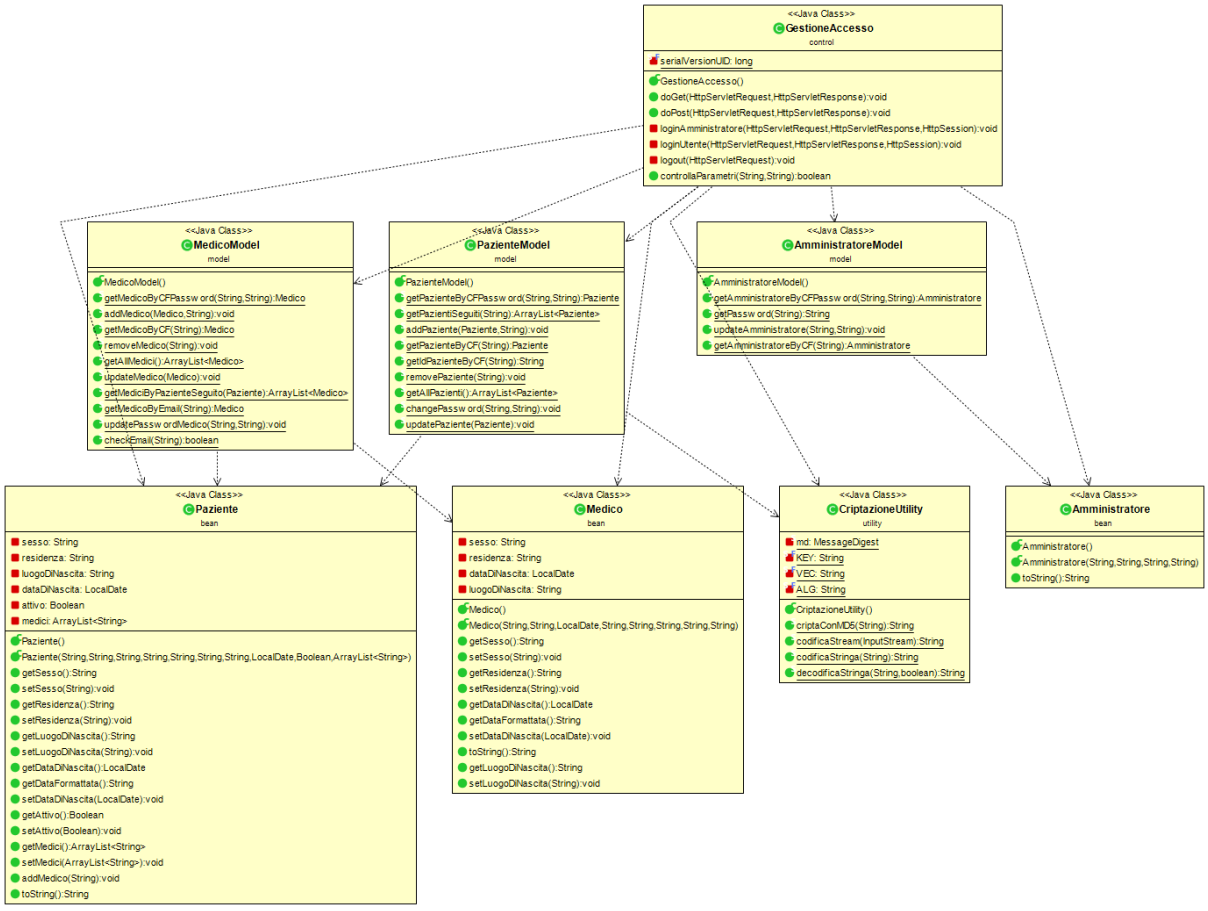
4.2 CD - Gestione Paziente



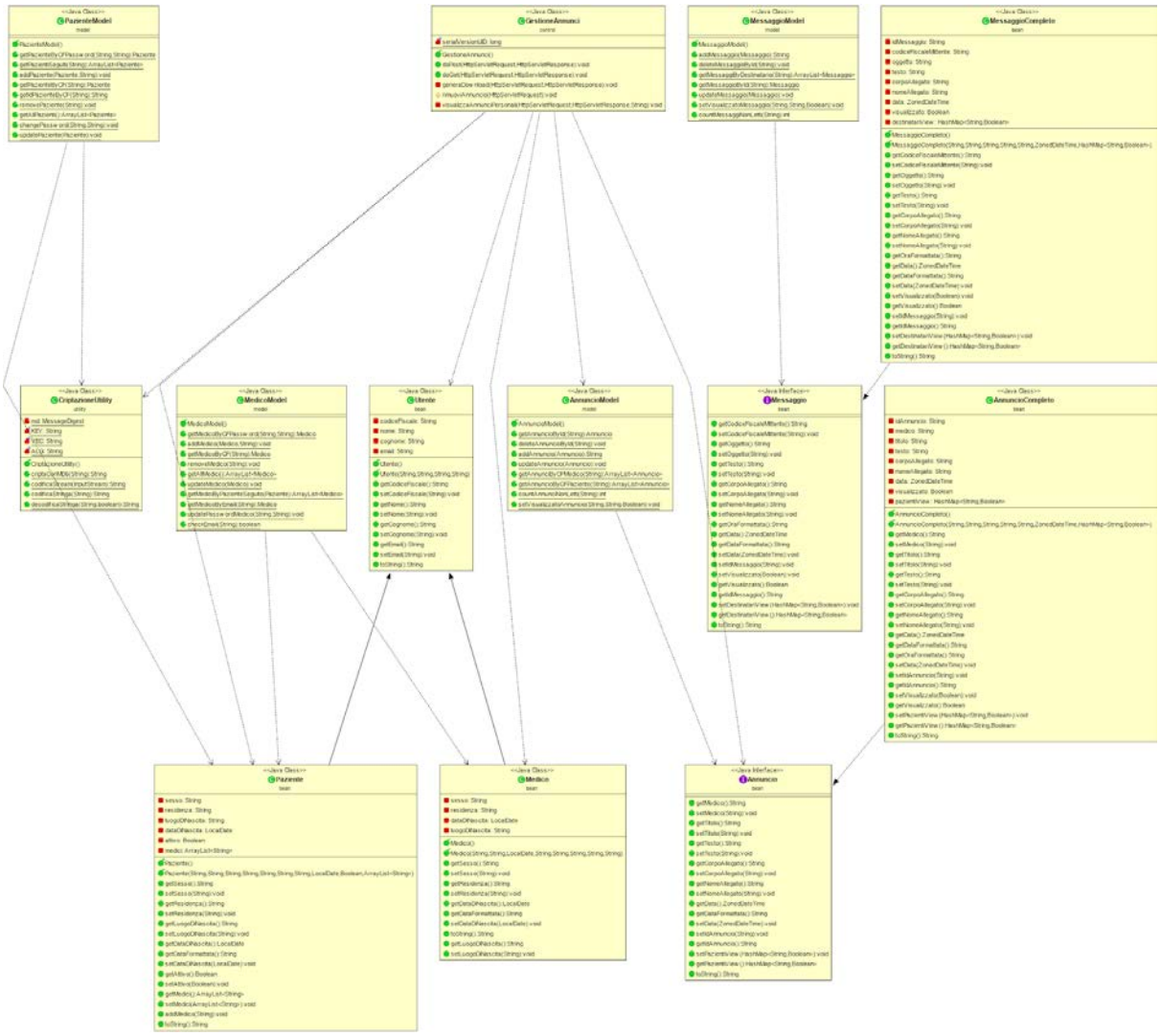
4.3 CD - Gestione Amministratore



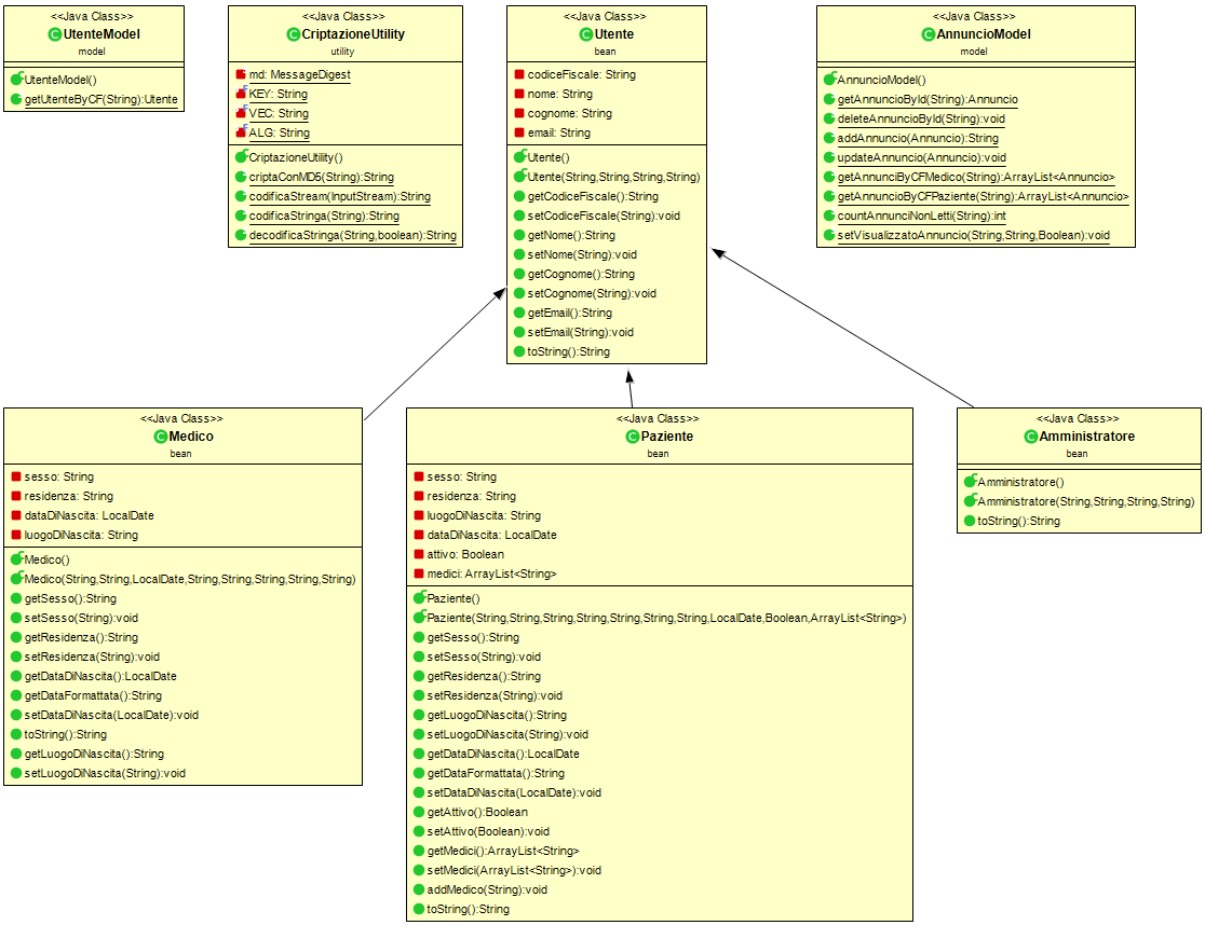
4.4 CD - Gestione Accesso



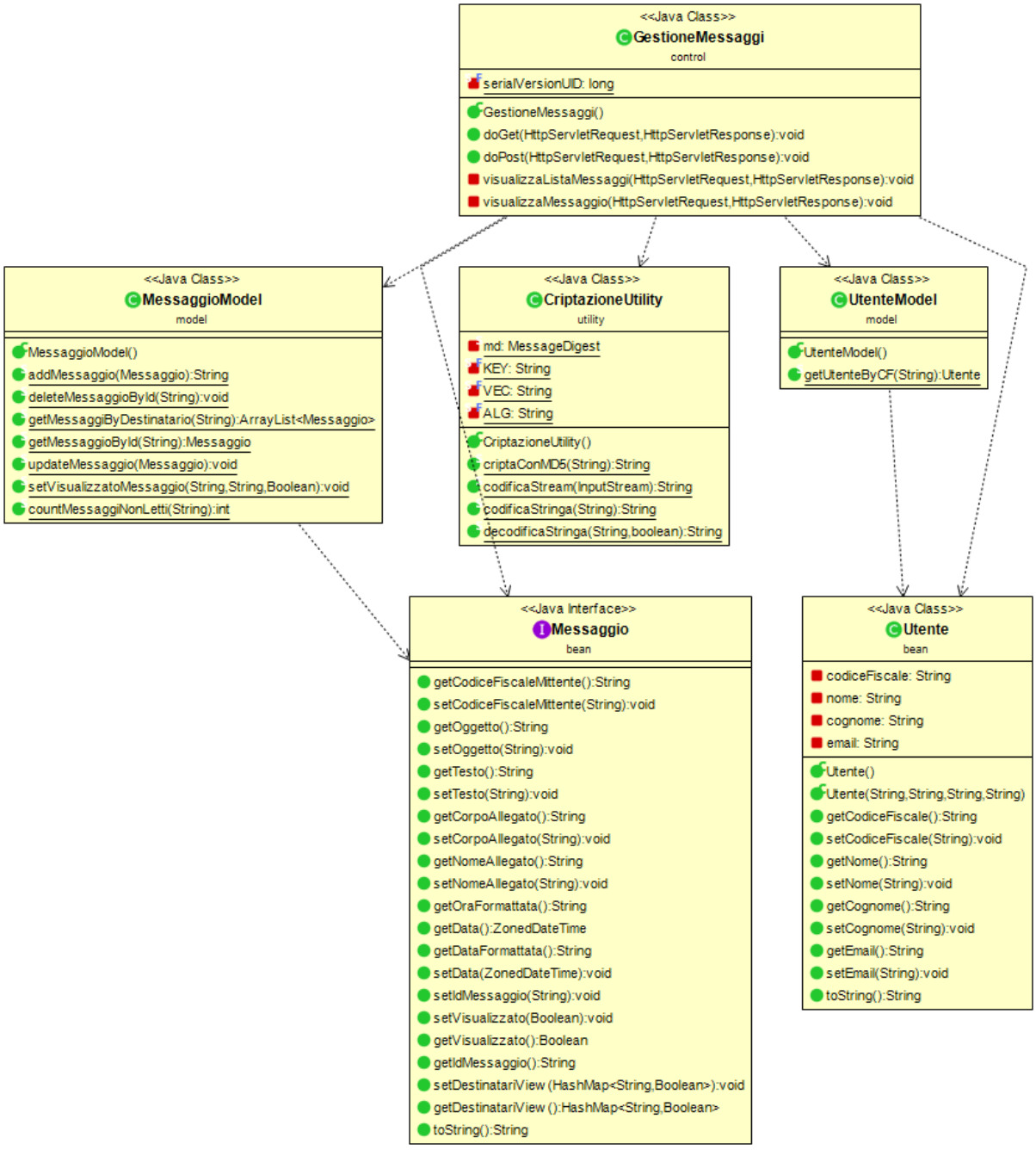
4.5 CD - Gestione Comunicazione



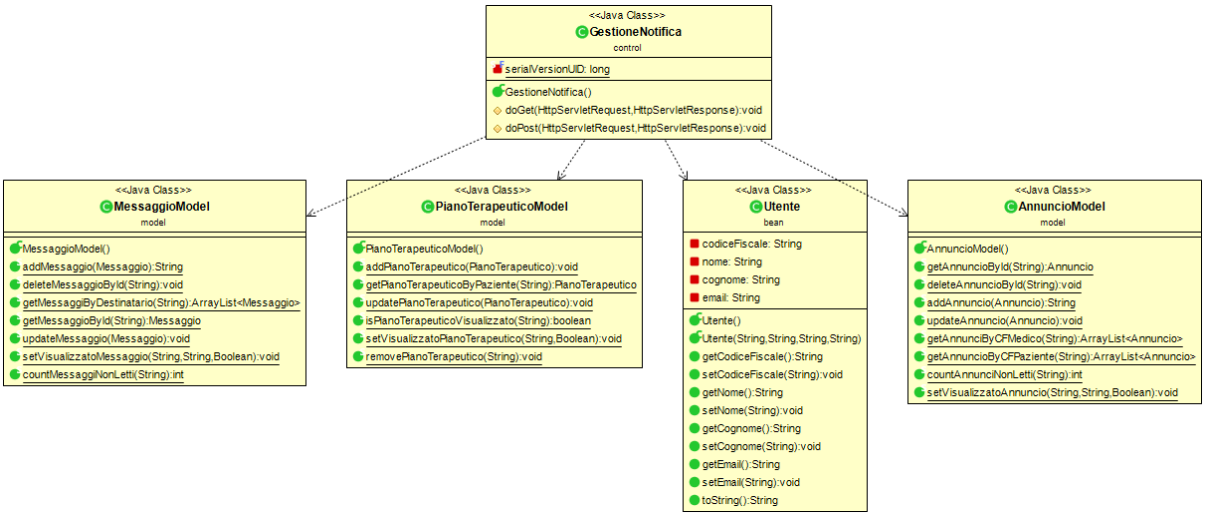
4.6 CD - Gestione Annuncio



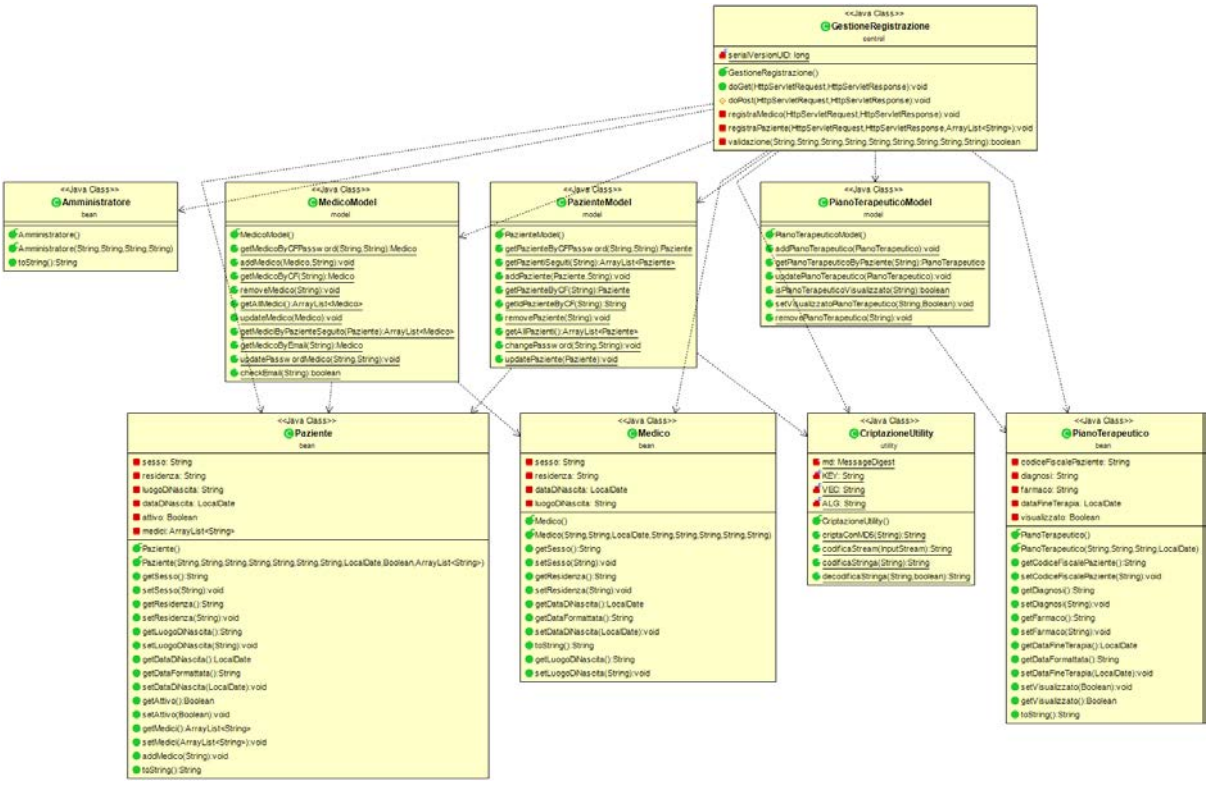
4.7 CD - Gestione Messaggio



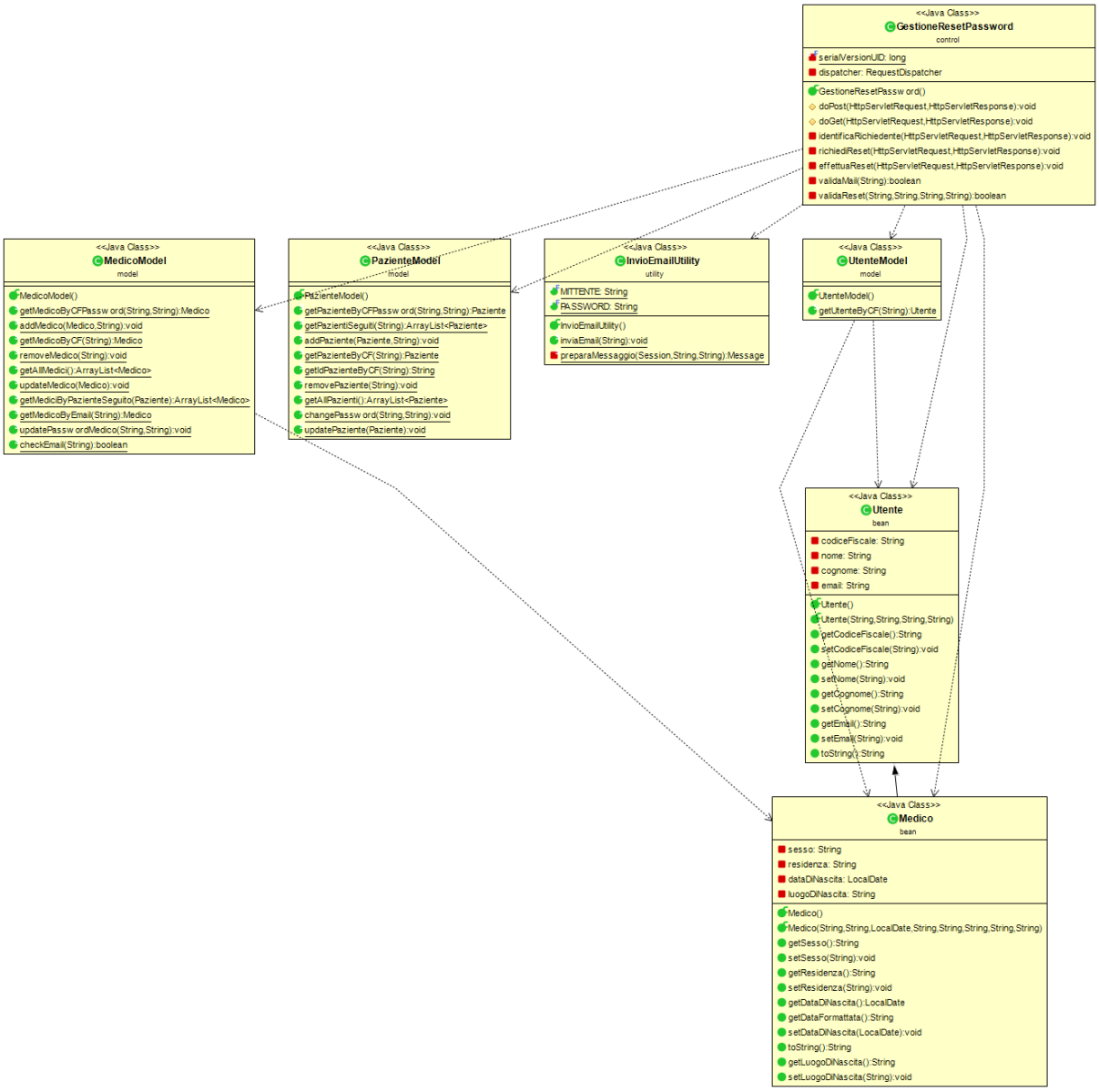
4.8 CD - Gestione Notifica



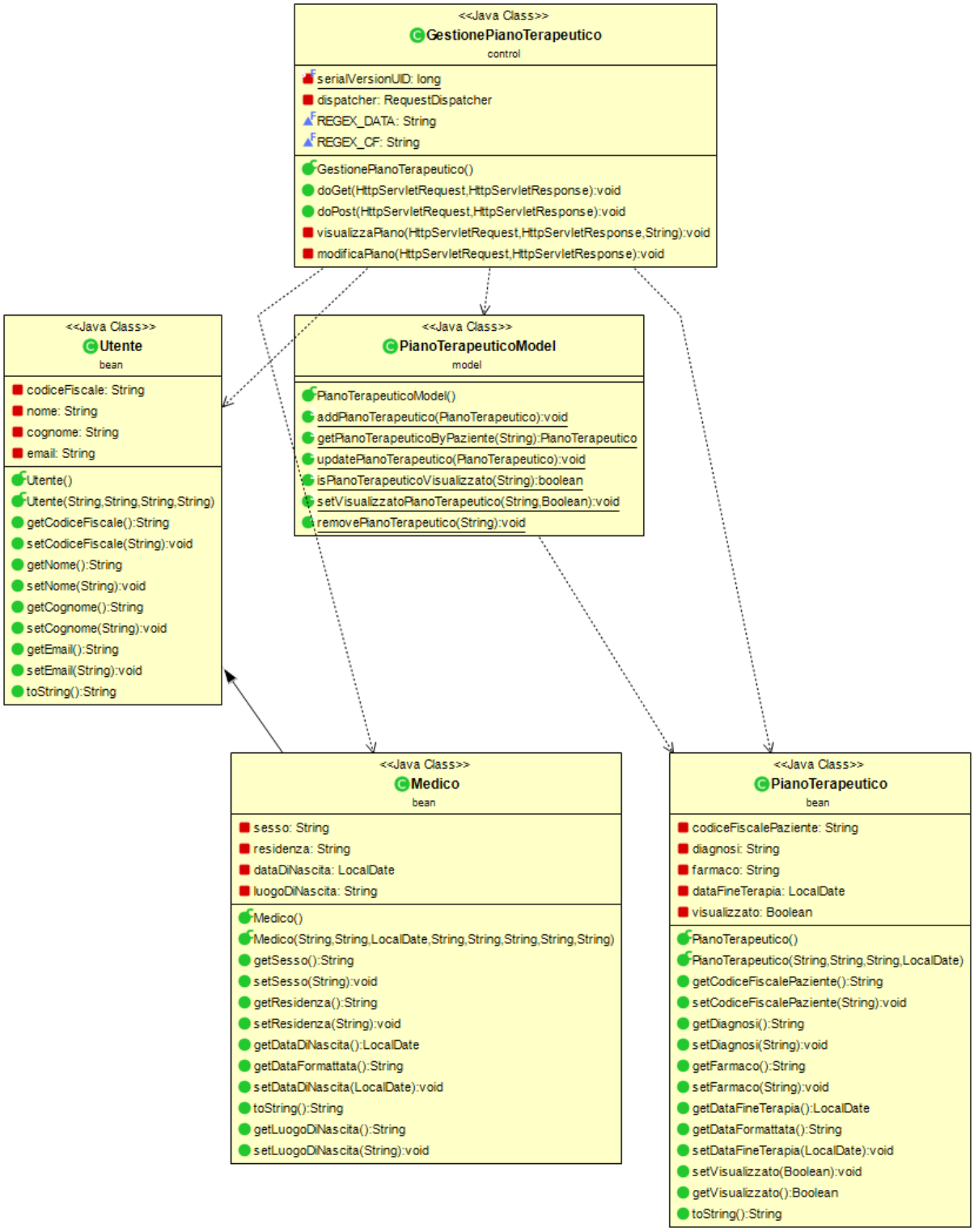
4.9 CD - Gestione Registrazione



4.10 CD - Gestione Reset Password



4.11 CD - Gestione Piano Terapeutico



4.12 CD - Gestione Parametri

5. Glossario

- **Javascript:** Linguaggio di scripting orientato agli oggetti e agli eventi, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web.
- **Eclipse:** Ambiente di sviluppo integrato multi-linguaggio e multi-piattaforma.
- **Javadoc:** È un applicativo incluso nel Java Development Kit utilizzato per la generazione automatica della documentazione del codice sorgente scritto in linguaggio Java.
- **Off-the-shelf:** Prodotti o servizi forniti da terze parte di cui viene fatto uso nello sviluppo del progetto.