# Object Design Document

## MediCare

| | |
|---|---|
| Riferimento | C14_ODD.pdf |
| Versione | 2.0 |
| Data | 20/12/2023 |
| Destinatario | F. Ferrucci, F. Palomba |
| Presentato da | Primo Vinicio Calabrese<br>Giovanni Casaburi<br>Matteo Avella<br>Gianluca Palumbo<br>Salvatore Basilicata<br>Domenico Alessandro Urciuoli |
| Approvato da | Luca Contrasto, Matteo Cicalese |

Team Members

| Ruolo | Nome | Acronimo | Contatto |
|---|---|---|---|
| Project Manager | Matteo Cicalese | MC | m.cicalese18@studenti.unisa.it |
| Project Manager | Luca Contrasto | LC | l.contrasto@studenti.unisa.it |
| Team Member | Primo Vinicio Calabrese | PVC | p.calabrese17@studenti.unisa.it |
| Team Member | Giovanni Casaburi | GC | g.casaburi16@studenti.unisa.it |
| Team Member | Matteo Avella | MA | m.avella17@studenti.unisa.it |
| Team Member | Gianluca Palumbo | GP | g.palumbo40@studenti.unisa.it |
| Team Member | Salvatore Basilicata | SB | s.basilicata@studenti.unisa.it |
| Team Member | Domenico Alessandro Urciuoli | DAU | d.urciuoli2@studenti.unisa.it |

# Revision History

| Data | Versione | Descrizione | Autori |
|---|---|---|---|
| 20/12/2023 | 0.1 | Prima stesura | Gianluca Palumbo |
| 21/12/2023 | 0.2 | Introduzione | Salvatore Basilicata, Primo V.Calabrese |
| 18/01/2023 | 0.3 | revisione | Giovanni Casaburi, Primo V.Calabrese |
| 19/01/2023 | 0.4 | Class diagram | Primo V.Calabrese |
| 19/01/2023 | 0.5 | Class Interfaces | Giovanni Casaburi |
| 22/01/2024 | 2.0 | Revisione ODD per la consegna | Giovanni Casaburi, Gianluca Palumbo, Matteo Avella, Salvatore Basilicata, Primo Vinicio Calabrese, Domenico Alessandro Urciuoli |
| | | | |
| | | | |
| | | | |

## Sommario



# 1. Introduzione

Il software da noi proposto, MediCare, si pone come obiettivo quello di fornire una piattaforma digitale che facilita la gestione e la fornitura di servizi di assistenza sanitaria, migliorando l'efficienza, la qualità e l'accessibilità delle cure mediche.

In questa prima sezione del documento saranno elencate: le linee guida per la fase di implementazione, le definizioni, gli acronimi e i documenti di riferimento.

## 1.1 Linee Guida per la Scrittura del Codice

Le linee guida includono una lista di standard che gli sviluppatori devono seguire per la progettazione delle interfacce. Di seguito viene presentata una lista di link alle documentazioni che verranno usate come riferimento:

- **HTML&CSS**
- **Python**

## 1.2 Definizioni, acronimi e abbreviazioni

Si elencano tutte le definizioni, acronimi e abbreviazioni presenti all'interno del documento:

- **ODD**: Object Design Document
- **HTML:** HyperText Markup Language
- **CSS:** Cascading Style Sheets
- **JS:** JavaScript
- **DB:** DataBase
- **DAO:** Data Access Object

## 1.3 Riferimenti e Link Utili

I riferimenti sono:

- Libro, "Object Oriented Software Engineering using UML, Patterns and Java" Edizione: 3rd Edition Anno: 2014
  Autori: Bernd Bruegge, Allen H. Dutoit
- Requirement Analisys Document
- Statement of work
- System Design Document
- Test Plan
- Test Case Specification
- Matrice di tracciabilità
- Spiegazione cosa sia un "cup"
- Dataset

# 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 ricalca la struttura di directory standard definita da Flask.

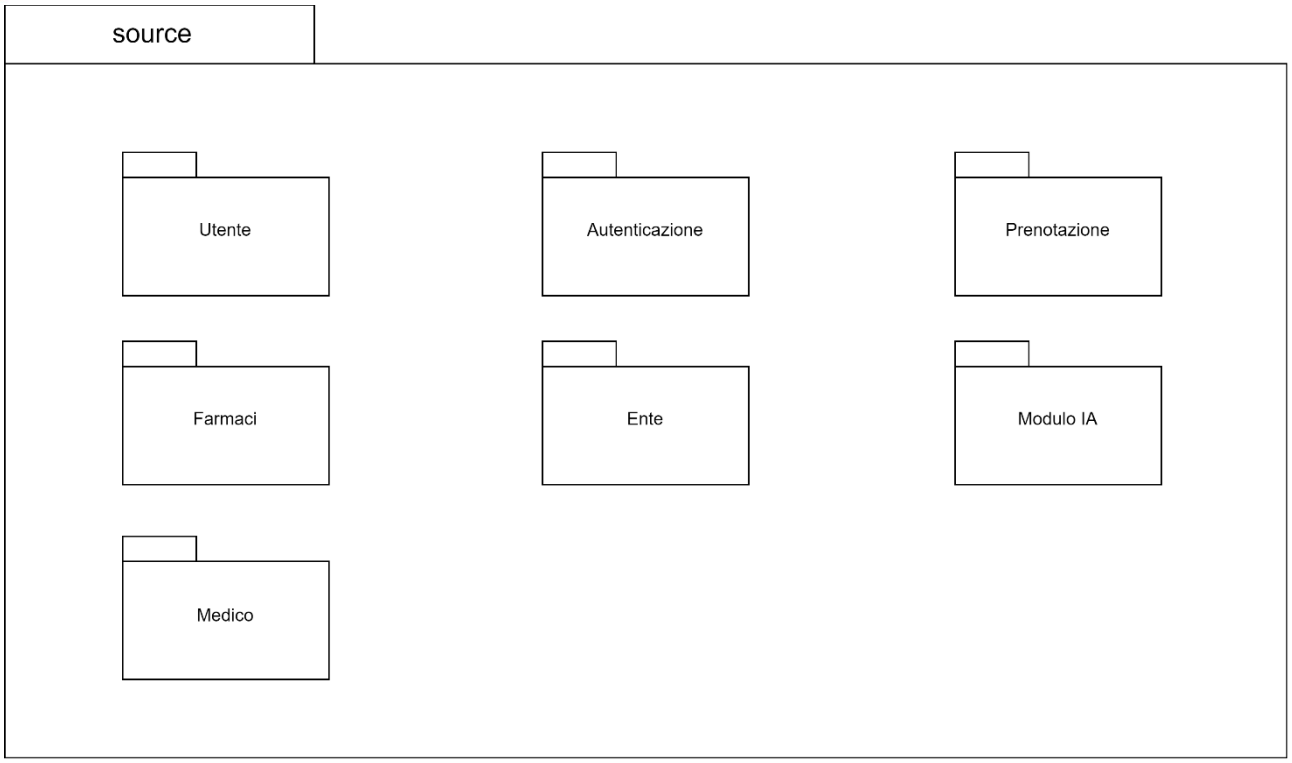
- **flaskDir**/, una cartella che contiene tutti i packages e files del codice sorgente.
  - **MediCare**/ una cartella contenente tutte le Entity del sistema.
  - **source/** una cartella contenente tutti i package dei Controller e Services.
  - **templates**/ una cartella contenente tutte le pagine html per l'interfaccia grafica.
  - **static**/ una cartella contenente tutti i file CSS per l'interfaccia grafica.
    - **images/** una cartella contenente tutte le immagini o video per l'interfaccia grafica.
    - **grafica/** una cartella contenente tutti i file CSS dell'interfaccia grafica.
  - **testing/**, una cartella contenente i package dei moduli di testing.
    - **BackEnd/** una cartella contenente tutti i testing d'unità.
    - **FrontEnd/** una cartella contenente tutti i testing di sistema.

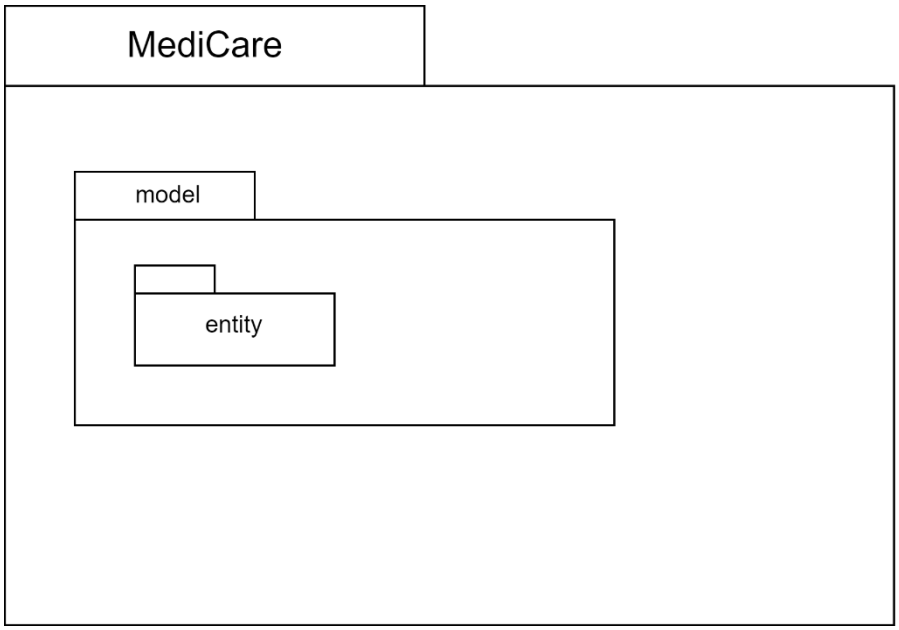
## Suddivisione Model e Service di Medicare

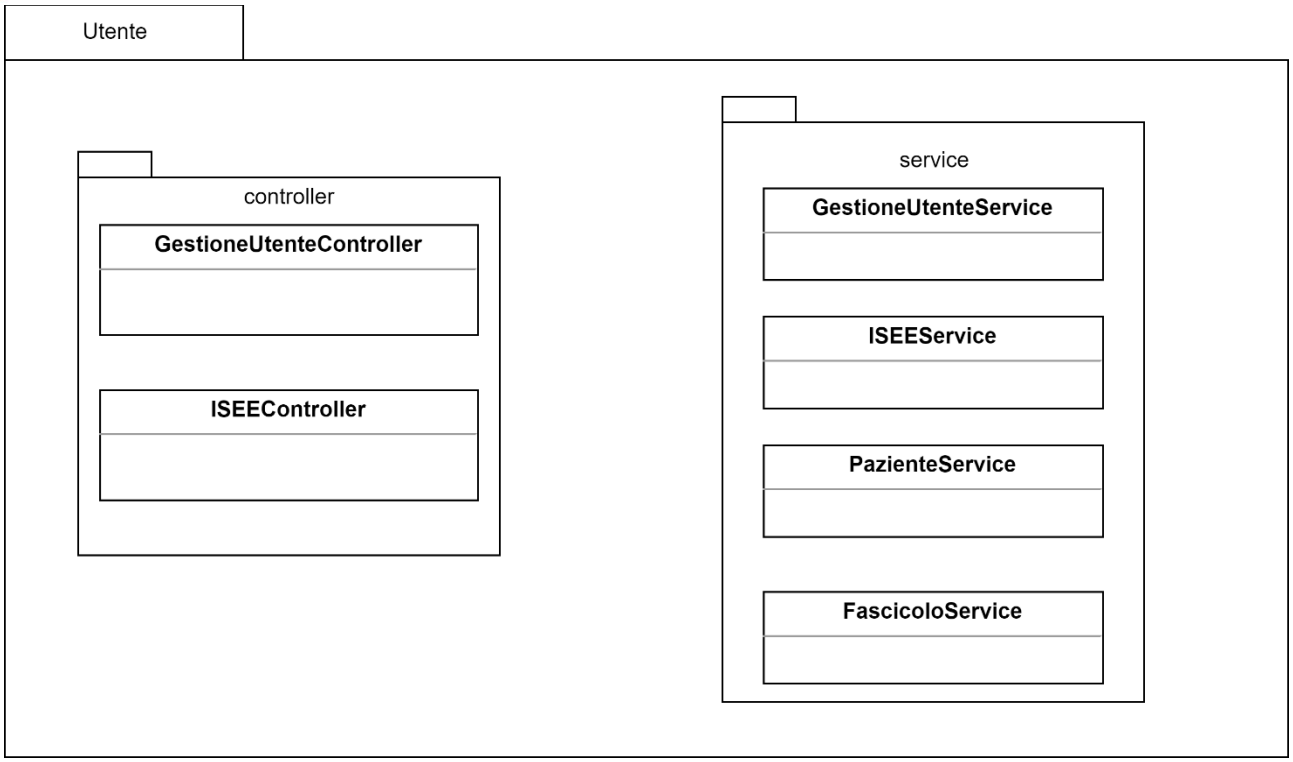
Nella presente sezione si illustra il modo in cui MediCare suddivide il Model dai sottosistemi che offrono i servizi. La suddivisione è stata organizzata seguendo questi criteri:

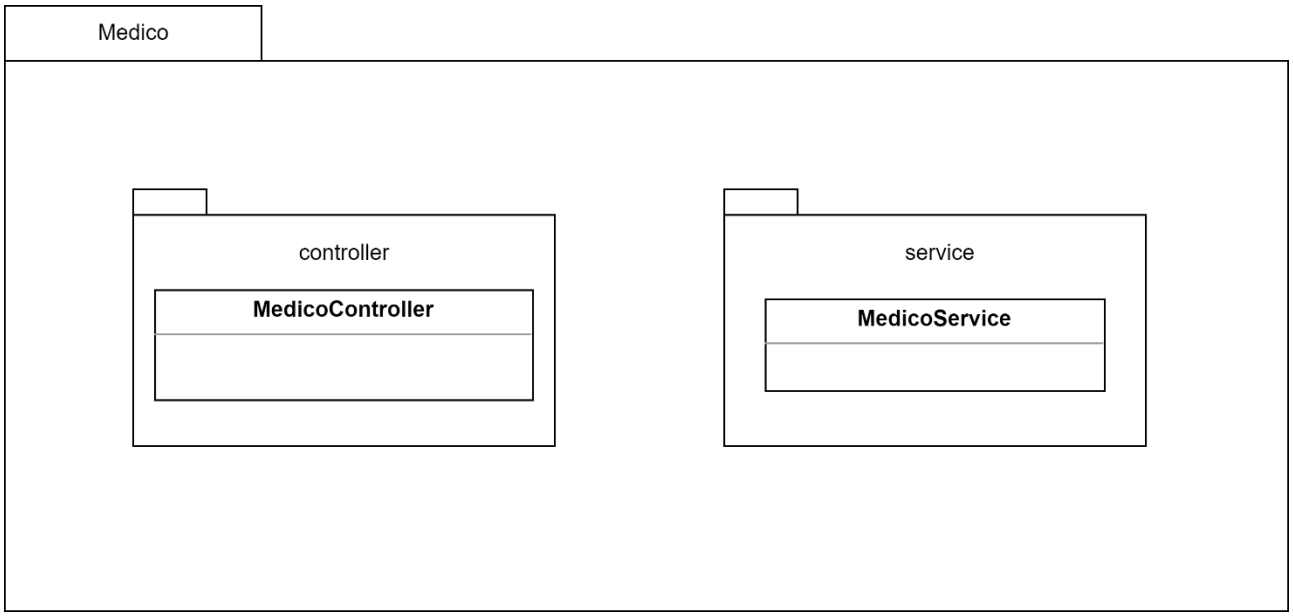
1. Creare una cartella source in cui saranno conservati i package di ogni sottosistema, in cui è presente la corrispettiva classe Controller e Service, con eventuali classi aggiuntive di utilità.
2. Creare un package separato per le classi del Data Access Layer chiamato *model*. Esso contiene le classi Entity, che grazie a Flask, fungono anche da DAO per accedere al database. Si è deciso di effettuare questa suddivisione per facilitare la manutenibilità ed estendibilità di queste classi, siccome rappresentano un punto critico ed essenziale del sistema.

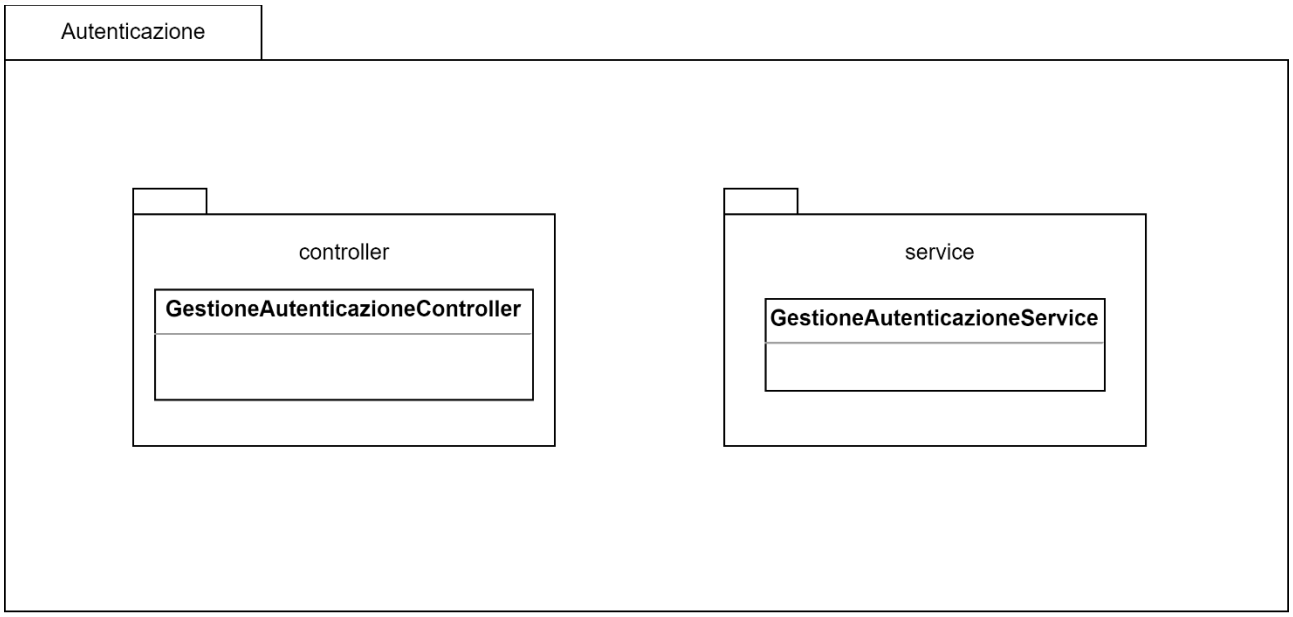
Tale suddivisione dei package ha portato alla creazione di una diretta dipendenza tra tutti i package con il package model. Di seguito è illustrata la struttura dei due package mediante un disegno:

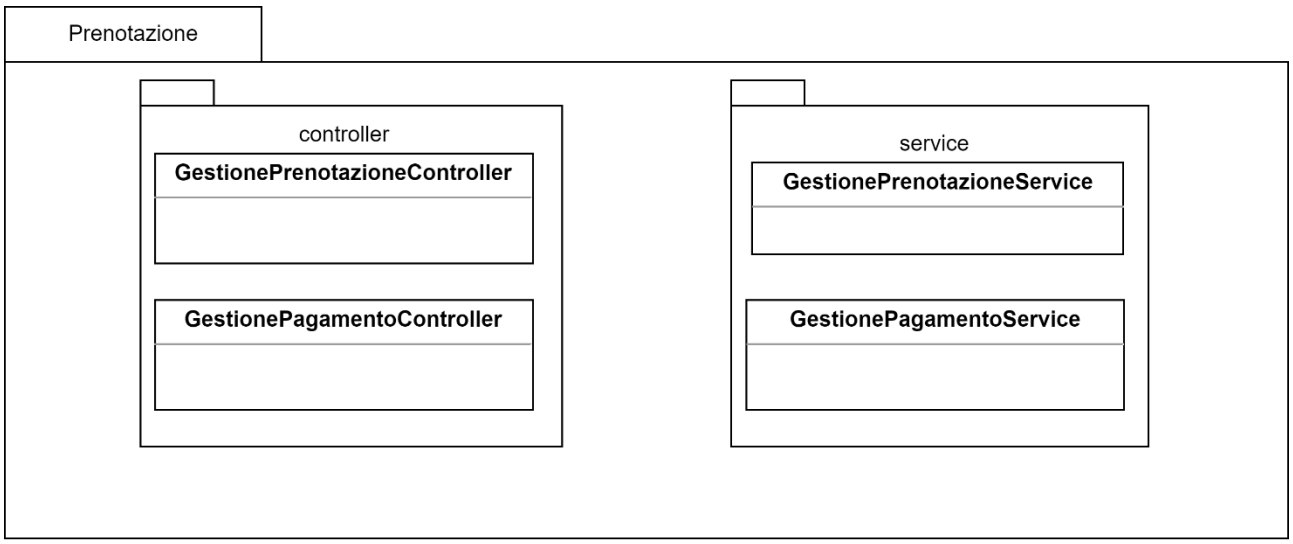

source
Utente
Autenticazione
Prenotazione
Farmaci
Ente
Modulo IA
Medico

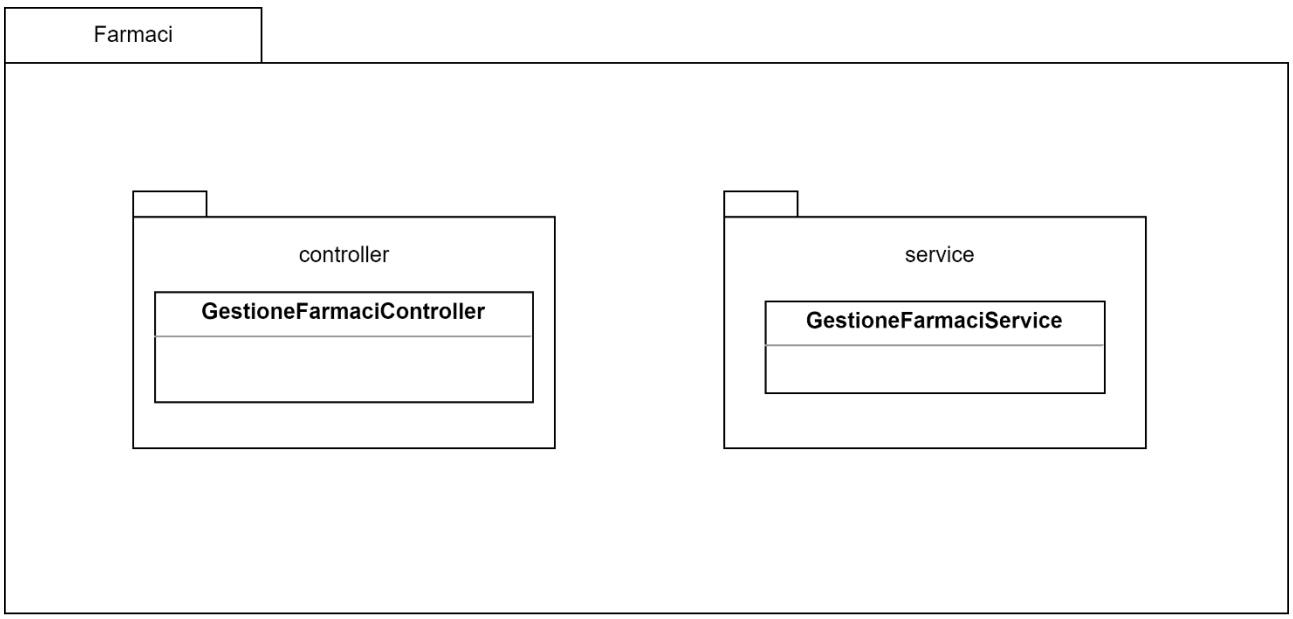

MediCare
model
entity

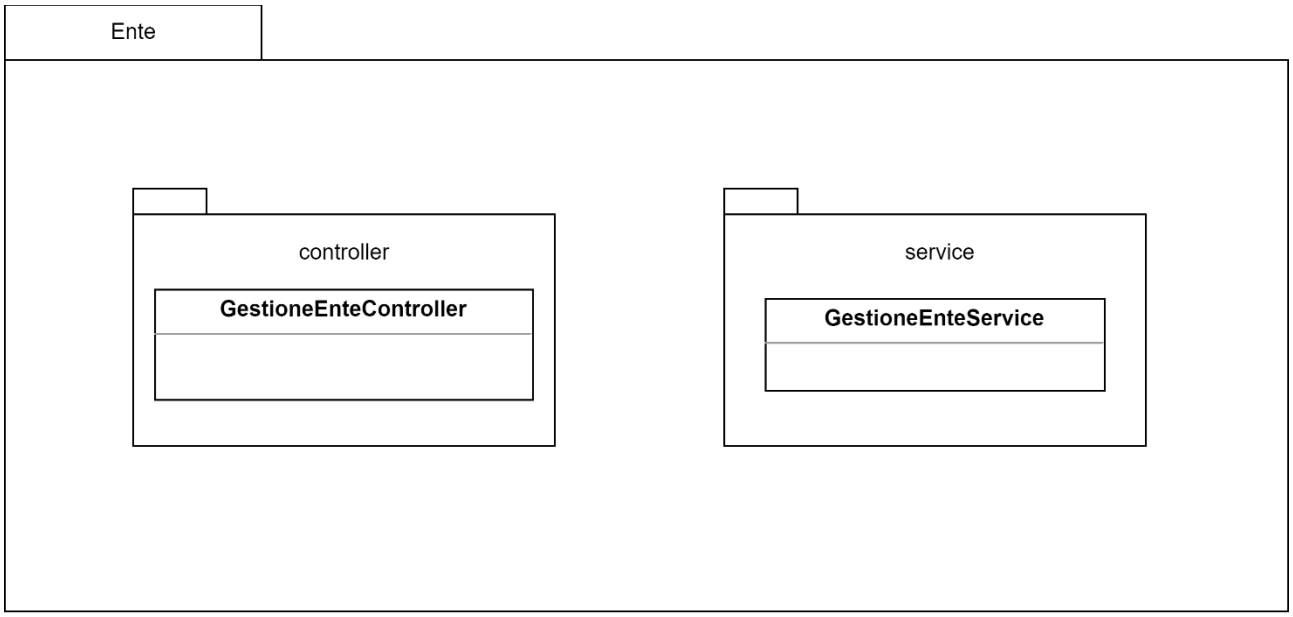

Utente
controller
GestioneUtenteController
ISEEController
service
GestioneUtenteService
ISEEService
PazienteService
FascicoloService

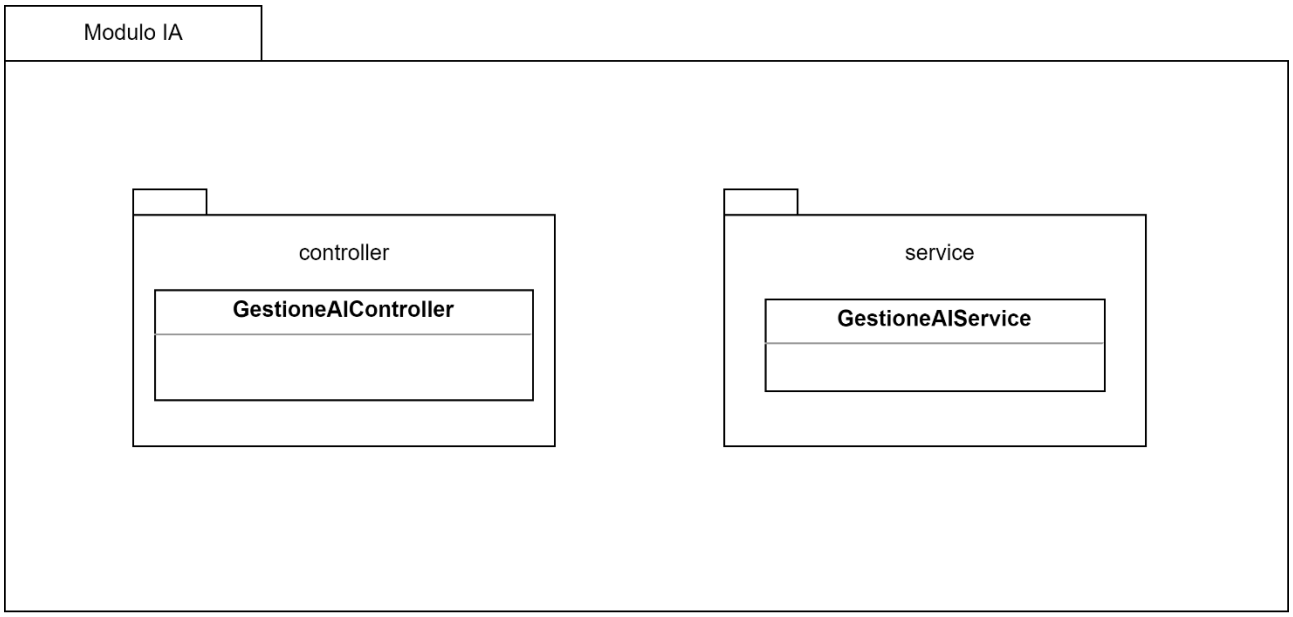

Medico
controller
MedicoController
service
MedicoService


Autenticazione
controller
GestioneAutenticazioneController
service
GestioneAutenticazioneService


Prenotazione
controller
GestionePrenotazioneController
GestionePagamentoController
service
GestionePrenotazioneService
GestionePagamentoService


Farmaci
controller
GestioneFarmaciController
service
GestioneFarmaciService


Ente
controller
GestioneEnteController
service
GestioneEnteService


Modulo IA
controller
GestioneAIController
service
GestioneAIService

# 3. Class Interfaces

### Package Utente

| **Nome classe** | ISEEService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative all'inserimento e modifica dell'ISEE di un utente. |
| **Metodi** | +changeISEE(float ISEE): Boolean |
| **Invariante di classe** | / |

| Nome Metodo | +changeISEE(float prezzo) |
|---|---|
| Descrizione | Questo metodo consente al medico privato di eliminare il suo account |
| Pre-condizione | / |
| Post-condizione | **Context**:ISEEService::changeISEE(float ISEE)<br>**post**: result==None \|\| result=False |

| **Nome classe** | PazienteService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità per gestire il paziente sulla piattaforma. |
| **Metodi** | + retrievePaziente(String email, String password): Paziente<br>+ getListaPrenotazioni(Paziente user): List<Prenotazioni)<br>+ getListaVaccini(Paziente user): List<DocumentiSanitario><br>+ eliminaPaziente(String CF): Boolean<br>+ getmoduloAIresult( Paziente p): DocumentoSanitario |
| **Invariante di classe** | / |

| Nome Metodo | +retrievePaziente(String email, String password) |
|---|---|
| Descrizione | Questo metodo consente di verificare se un paziente è presente nel database. |
| Pre-condizione | / |
| Post-condizione | **Context**:PazienteService::retrievePaziente(String email ,String password)<br>**post**: paziente==None \|\| paziente!=None |

| Nome Metodo | +getListaPrenotazioni(Paziente user) |
|---|---|
| Descrizione | Questo metodo consente di prendere tutte le prenotazioni di un determinato paziente. |
| Pre-condizione | / |
| Post-condizione | **Context**:PazienteService::getListaPrenotazioni(Paziente user)<br>**Post:** result==None \|\| result!=None |

| Nome Metodo | +getListaVaccini(Paziente user) |
|---|---|
| Descrizione | Questo metodo consente di prendere tutti i vaccini di un determinato paziente. |
| Pre-condizione | / |
| Post-condizione | **Context**:PazienteService::getListaVaccini(Paziente user)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +eliminaPaziente(String CF) |
|---|---|
| Descrizione | Questo metodo consente di eliminare un paziente dalla piattaforma. |
| Pre-condizione | / |
| Post-condizione | **Context**:PazienteService::eliminaPaziente(String CF)<br>**post**: result==True \|\| result==False |
| Nome Metodo | +getmoduloAIresult(Paziente p) |
| Descrizione | Questo metodo consente di recuperare il documento sanitario relativo alla diagnosi cardiaca effettuata dal modulo AI. |
| Pre-condizione | / |
| Post-condizione | **Context**:PazienteService::getmoduloAIresult(Paziente p)<br>**post**: doc==None \|\| doc!= None |

| **Nome classe** | FascicoloService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative all'FSE di un determinato paziente. |
| **Metodi** | + getDocumentiSanitari(String cf): List<DocumentoSanitario><br>+ addDocumento(String tipo, String descrizione, String richiamo, String codicefiscale):void |
| **Invariante di classe** | / |

| Nome Metodo | +getDocumentiSanitari(String cf): List<DocumentoSanitario> |
|---|---|
| Descrizione | Questo metodo consente di prendere tutti i documenti sanitari del fascicolo di un determinato utente. |
| Pre-condizione | / |
| Post-condizione | **Context**:FascicoloService::getDocumentiSanitari(String cf)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +addDocumento(String tipo, String descrizione, String richiamo, String codicefiscale) |
|---|---|
| Descrizione | Questo metodo consente al medico di aggiungere un documento sanitario nel fascicolo di un paziente che ha visitato. |
| Pre-condizione | / |
| Post-condizione | **Context**:FascicoloService::addDocumento(String tipo, String descrizione, String richiamo, String codicefiscale)<br>**post**: / |

## Package Autenticazione

| **Nome classe** | AutenticazioneService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative all'autenitcazione e registrazione di un utente sulla piattaforma. |
| **Metodi** | + login_page(String email, String password, String tipo): Boolean<br>+ auth_2fa(String codice): Boolean<br>+loginEnte_page(String email, String password): Boolean<br>+ RegistrazioneMedico_page(String email, String password, String nome, String cognome, int iscrizione, String specializzazione, String città): Boolean<br>+ RegistrazionePaziente_page(String email, String password, String nome, String cognome, String CF, String cellulare, String luogoNascita, String domicilio, String dataNascita, String sesso): Boolean<br>+RegistrazioneEnte(String email, String password, String nome, String città): Boolean<br>+logout():Boolean |
| **Invariante di classe** | / |

| Nome Metodo | login_page(String email, String password, String tipo) |
|---|---|
| Descrizione | Questo metodo permette ad un utente diverso dall'ente di autenticarsi |
| Pre-condizione | / |
| Post-condizione | **context**:AutenticazioneService:: login_page(String email, String password, String tipo)<br>**post**:current_user.is_authenticated==True \|\|<br>current_user.is_authenticated==False |

| Nome Metodo | auth_2fa(String codice) |
|---|---|
| Descrizione | Questo metodo verifica che il codice immesso dal paziente è corretto |
| Pre-condizione | / |
| Post-condizione | **context**:AutenticazioneService:: auth_2fa (String codice)<br>**post**:result==True \|\| result==False |

| Nome Metodo | loginEnte(String email, String password, String nome, String città) |
|---|---|
| Descrizione | Questo metodo permette all'ente sanitario di autenticarsi |
| Pre-condizione | / |
| Post-condizione | **context**:AutenticazioneService::loginEnte(String email, String password, String nome, String città)<br>**post**:current_user.is_authenticated==True \|\|<br>current_user.is_authenticated==False |

| Nome Metodo | RegistrazioneMedico_page(String email, String password, String nome, String cognome, int iscrizione, String specializzazione, String città) |
|---|---|
| Descrizione | Questo metodo permette al medico privato di registrarsi sulla piattaforma |
| Pre-condizione | / |

| | |
|---|---|
| Post-condizione | **context**:AutenticazioneService::RegistrazioneMedico_page(String email, String password, String nome, String cognome, int iscrizione, String specializzazione, String città)<br>**post**: result==true \|\| MedicoService.retrieveMedico(email,password)!=None and result==False \|\| result==False |

| Nome Metodo | RegistrazioneEnte(String email, String password, String nome, String città) |
|---|---|
| Descrizione | Questo metodo permette all'Ente Sanitario di registrarsi sulla piattaforma |
| Pre-condizione | / |
| Post-condizione | **context**:AutenticazioneService::RegistrazioneEnte(String email, String password, String nome, String città)<br>**post**:result==True \|\| GestioneEnteService.retrieveEnte(email,password)!=None and result==False \|\| result==False |

| Nome Metodo | RegistrazionePaziente_page(String email, String password, String nome, String cognome, String CF, String cellulare, String luogoNascita, String domicilio, String dataNascita, String sesso) |
|---|---|
| Descrizione | Questo metodo permette al paziente di registrarsi sulla piattaforma |
| Pre-condizione | / |
| Post-condizione | **context**:AutenticazioneService:: RegistrazionePaziente_page(String email, String password, String nome, String cognome, String CF, String cellulare, String luogoNascita, String domicilio, String dataNascita, String sesso)<br><br>**post**:result==True \|\| PazienteService.retrievePaziente(email,password)!=None and result==False \|\| result==False |

| Nome Metodo | logout() |
|---|---|
| Descrizione | Questo metodo permette ad utente qualsiasi di effettuare il logout dalla piattaforma |
| Pre-condizione | Current_user.is_authenticated==True |
| Post-condizione | **context**:AutenticazioneService:: logout()<br>**post**:logout_user()==True |

## Package Medico

| **Nome classe** | MedicoService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla gestione del medico |
| **Metodi** | + getPazienti(String dottore): List<Paziente><br>+ getMedico(String email): Medico<br>+ retrieveMedico(String email, String password): Medico<br>+ getListaMedici(): List<Medico><br>+ getListaCentri():<ListMedico><br>+filtraMedici(String specializzazione, String città): List<Medico><br>+ addMedicotoLista(Medico medico):void<br>+ rimuoviMedico(String email): Boolean |

| **Invariante di classe** | / |
|---|---|

| Nome Metodo | +getPazienti(String dottore) |
|---|---|
| Descrizione | Questo metodo permette al medico di vedere tutti i suoi pazienti per cui ha effettuato una visita. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: getPazienti(String dottore)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +getMedico(String email):Medico |
|---|---|
| Descrizione | Questo metodo permette di cercare un determinato medico sulla piattaforma per email per la registrazione. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: getMedico(String email)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +retrieveMedico(String email, String password) |
|---|---|
| Descrizione | Questo metodo permette di trovare un determinato medico sulla piattaforma per email e password per il login. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: retrieveMedicoi(String email, String password)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +getListaMedici() |
|---|---|
| Descrizione | Questo metodo permette al paziente di visualizzare tutti i medici della piattaforma. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: getListaMedici()<br>**post**: result!=None |

| Nome Metodo | +getListaCentri() |
|---|---|
| Descrizione | Questo metodo permette al paziente di visualizzare tutti i centri vaccinali registrati sulla piattaforma |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: getListaCentri()<br>**post**: result!=None |

| Nome Metodo | +filtraMedici(String specializzazione, String citta) |
|---|---|
| Descrizione | Questo metodo permette al paziente di filtrare la lista dei medici in base alla specializzazione ed alla città. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: filtraMedici(String specializzazione, String citta) |

| | |
|---|---|
| | **post**: result==None \|\| result!=None |

| Nome Metodo | +addMedicotoLista(Medico medico) |
|---|---|
| Descrizione | Questo metodo permette di aggiungere un medico appena registrato sulla piattaforma alla lista dei medici. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: addMedicotoLista(Medico medico)<br>**post**: / |

| Nome Metodo | +rimuoviMedico(String email) |
|---|---|
| Descrizione | Questo metodo permette al medico di eliminare il proprio account sulla piattaforma in base alla email. |
| Pre-condizione | / |
| Post-condizione | **context**:MedicoService:: rimuoviMedico(String email)<br>**post**: result==True \|\| result=False |

## Package Prenotazione

| **Nome classe** | PrenotazioneService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla prenotazione |
| **Metodi** | +getListaMedici(String specializzazione, String citta):List<Medico><br>+ getListaVaccini(Paziente user):List<DocumentoSanitario><br>+ getListaPrenotazione(Paziente user):List<Prenotazione><br>+getListaPrenotazioneMedico(Medico medico):List<Prenotazione><br>+ confirmisFree(String idmedico, String data, int ora):Boolen<br>+ savePrenotazione(String idmedico, String data, int ora, String tipo, String CF, float prezzo, String carta):Boolean<br>+ saveVaccino e(String idmedico, String data, int ora, String tipo, String CF, float prezzo): Boolean<br>+ modificaPrenotazione(int id, String data, int ora): Boolean<br>+ getGiorniCorrenti():Lista<int><br>+ confirmVaccino(idmedico, data, ora):Boolean |
| **Invariante di classe** | / |

| Nome Metodo | +getListaMedici(String specializzazione, String citta) |
|---|---|
| Descrizione | Questo metodo permette di vedere tutti i medici presenti sulla piattaforma per specializzazione e città. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: getListaMedici(String specializzazione, String citta)<br>**post**: MedicoService.filtraMedici(specializzazione,città)==None \|\| MedicoService.filtraMedici(specializzazione,città)!=None |

| Nome Metodo | + getListaVaccini(Paziente user):List<DocumentoSanitario> |
|---|---|
| Descrizione | Questo metodo permette al Paziente di visualizzare i propri vaccini |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: getListaVaccini(Paziente user)<br>**post**: PazienteService.getListaVaccini(user)==None \|\|<br>PazienteService.getListaVaccini(user)!=None |

| Nome Metodo | + getListaPrenotazione(Paziente user) |
|---|---|
| Descrizione | Questo metodo permette al Paziente di visualizzare tutte le sue prenotazioni |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: getListaPrenotazione(Paziente user)<br>**post**: PazienteService.getListaPrenotazioni(user)==None \|\|<br>PazienteService.getListaPrenotazioni!=None |

| Nome Metodo | +getListaPrenotazioniMedico(Medico medico) |
|---|---|
| Descrizione | Questo metodo permette al medico di vedere tutte le prenotazioni che ha ricevuto. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService::getListaPrenotazioni(Medico medico)<br>**post**: result==None \|\| result!=None |

| Nome Metodo | +confirmIsFree(String idmedico, String data, int ora) |
|---|---|
| Descrizione | Questo metodo permette di verificare se quello slot di tempo è disponibile per prenotare la visita o analisi. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService::confirmIsFree(String idmedico, String data, int ora)<br>**post**: prenotazione!=None and result==False \|\| prenotazione==None and result ==True |

| Nome Metodo | + savePrenotazione(String idmedico, String data, int ora, String tipo, String CF, float prezzo, String carta) |
|---|---|
| Descrizione | Questo metodo permette di salvare la prenotazione sul database. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: savePrenotazione(String idmedico, String data, int ora, String tipo, String CF, float prezzo, String carta)<br>**Post:** result==True \|\| result==False |

| Nome Metodo | + saveVaccino(String idmedico, String data, int ora, String tipo, String CF, float prezzo) |
|---|---|
| Descrizione | Questo metodo permette di salvare la prenotazione di un vaccino sul database |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: saveVaccino(String idmedico, String data, int ora, String tipo, String CF, float prezzo) |

| | |
|---|---|
| | **Post:** result==True \|\| result==False |

| Nome Metodo | + modificaPrenotazione(int id, String data, int ora) |
|---|---|
| Descrizione | Questo metodo permette al paziente di modificare una sua prenotazione esistente. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: modificaPrenotazione(int id, String data, int ora)<br>**Post:** result=True \|\| result==False |

| Nome Metodo | + getGiorniCorrenti() |
|---|---|
| Descrizione | Questo metodo permette di calcolare i giorni rimanenti del mese e quelli da considerare di quello successivo. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: getGiorniCorrenti()<br>**Post:** giorni!=None and len(giorni)==2 |

| Nome Metodo | + confirmVaccino(String idmedico, String data, int ora) |
|---|---|
| Descrizione | Questo metodo permette di verificare che quello slot di tempo è disponibile per prenotare il vaccino. |
| Pre-condizione | / |
| Post-condizione | **context**:PrenotazioneService:: confirmVaccino(String idmedico, String data, int ora)<br>**Post:** result==True \|\| result==False |

| **Nome classe** | PagamentoService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla gestione dei metodi di pagamento dell'utente |
| **Metodi** | + getMetodi(String cf):List<MetodiPagamento><br>+ eliminaMetodo(String PAN, String CF):void<br>+ addCarta(Int CVV, String PAN, String scadenza, String CF):Boolean |
| **Invariante di classe** | / |

| Nome Metodo | + getMetodi(String cf) |
|---|---|
| Descrizione | Questo metodo permette all'utente di vedere tutti i metodi di pagamento che ha inserito. |
| Pre-condizione | / |
| Post-condizione | **Context**:PagamentoService:: getMetodi(String cf)<br><br>**Post**:metodi==None \|\| metodi==None |

| Nome Metodo | +eliminaMetodo(String PAN, String CF) |
|---|---|
| Descrizione | Questo metodo permette all'utente di eliminare un metodo di pagamento che ha inserito. |

| | |
|---|---|
| Pre-condizione | / |
| Post-condizione | **Context**:PagamentoService:: getMetodi(String PAN, String CF)<br>**Post:** db.session.delete(metodo)==None |
| Nome Metodo | + addCarta(Int CVV, String PAN, String scadenza, String CF) |
| Descrizione | Questo metodo permette all'utente d'inserire un metodo di pagamento nuovo. |
| Pre-condizione | / |
| Post-condizione | **Context**:GestionePagamentoService:: addCarta(Int CVV, String PAN, String scadenza, String CF)<br>**Post**: result==True \|\| cartaEsistente!=None and result==None \|\| result==False |

## Package Farmaci

| **Nome classe** | FarmaciService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla visualizzazione dei farmaci |
| **Metodi** | + getFarmaci (): List<Farmaco><br>+ ricerca (String suggest): List<Farmaco><br>+ filtraCatalogo (String categoria, String prezzo): List<Farmaco><br>+ getdettagliFarmaco(int id): Farmaco<br>+ ricercaNome(String nome): List<Farmaco> |
| **Invariante di classe** | / |

| Nome Metodo | + getFarmaci(): List<Farmaco> |
|---|---|
| Descrizione | Questo metodo permette di visualizzare tutti i farmaci dell'archivio |
| Pre-condizione | / |
| Post-condizione | **Context**:FarmaciService::visualizzaFarmaci()<br>**Post:** Farmaci.query.all()!=None |

| Nome Metodo | + ricerca (String suggest) |
|---|---|
| Descrizione | Questo metodo permette di ricevere una lista di suggerimenti in base a ciò che scrive l'utente nella barra di ricerca |
| Pre-condizione | / |
| Post-condizione | **context::**FarmaciService::ricerca(String suggest)<br>**post:** suggest!=None \|\| suggest==None |

| Nome Metodo | + filtraCatalogo (String categoria, String prezzo) |
|---|---|
| Descrizione | Questo metodo permette di filtrare i farmaci per categoria e per prezzo |
| Pre-condizione | / |
| Post-condizione | **context**::FarmaciService::filtraCatalogo(String categoria, String prezzo)<br>**post**: listaFiltrata==None \|\| listaFiltrata!=None |

| Nome Metodo | + getdettagliFarmaco(int id) |
|---|---|

| | |
|---|---|
| Descrizione | Questo metodo permette di visualizzare i dettagli del farmaco trovato per id |
| Pre-condizione | / |
| Post-condizione | **Context**:FarmaciService::getdettagliFarmaco(int id)<br>**Post**: Farmaco.query.filter_by(ID=id)!=None |

| Nome Metodo | + ricercaNome(String nome) |
|---|---|
| Descrizione | Questo metodo permette di cercare dei farmaci in base al nome |
| Pre-condizione | / |
| Post-condizione | **Context**:FarmaciService::ricercaNome(String nome)<br>**Post**: listFarmaci=None \|\| listFarmaci==None |

## Package Ente

| **Nome classe** | EnteService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla gestione dell'Ente di account medici pubblici |
| **Metodi** | + creaReparto(String nome ,String Email, String Password,String specializzazione, String città, String ente ): Boolean<br>+ deleteReparto(String Email): void<br>+ retrieveEnte(String email, String Password): EnteSanitario |
| **Invariante di classe** | |

| Nome Metodo | + creaReparto(String nome ,String Email, String Password,String specializzazione, String città, String ente ) |
|---|---|
| Descrizione | Questo metodo permette all'Ente di creare un account di reparto per un medico pubblico |
| Pre-condizione | / |
| Post-condizione | **Context**:EnteService:: creaReparto(String nome ,String Email, String Password,String specializzazione, String città, String ente )<br>**Post:** result==True \|\| MedicoService.retrieveMedico(email, password)!=None and result==False\|\| result==False |

| Nome Metodo | + deleteReparto(String Email) |
|---|---|
| Descrizione | Questo metodo permette all'Ente di eliminare un account di reparto per un medico pubblico |
| Pre-condizione | **Context**:EnteService:: deleteReparto(String Email)<br>**Pre**:MedicoService.retrieveMedico(email,password)!=None |
| Post-condizione | **Context**:EnteService:: deleteReparto(String Email)<br>**Post**: database.session.delete(medico)==None |

| Nome Metodo | + retrieveEnte(String email, String password) |
|---|---|
| Descrizione | Questo metodo permette di vedere se un ente è registrato sulla piattaforma |

| | |
|---|---|
| Pre-condizione | |
| Post-condizione | **Context**:EnteService:: retrieveEnte(String email, String password)<br>**Post**: ente!=None \|\| ente==None |

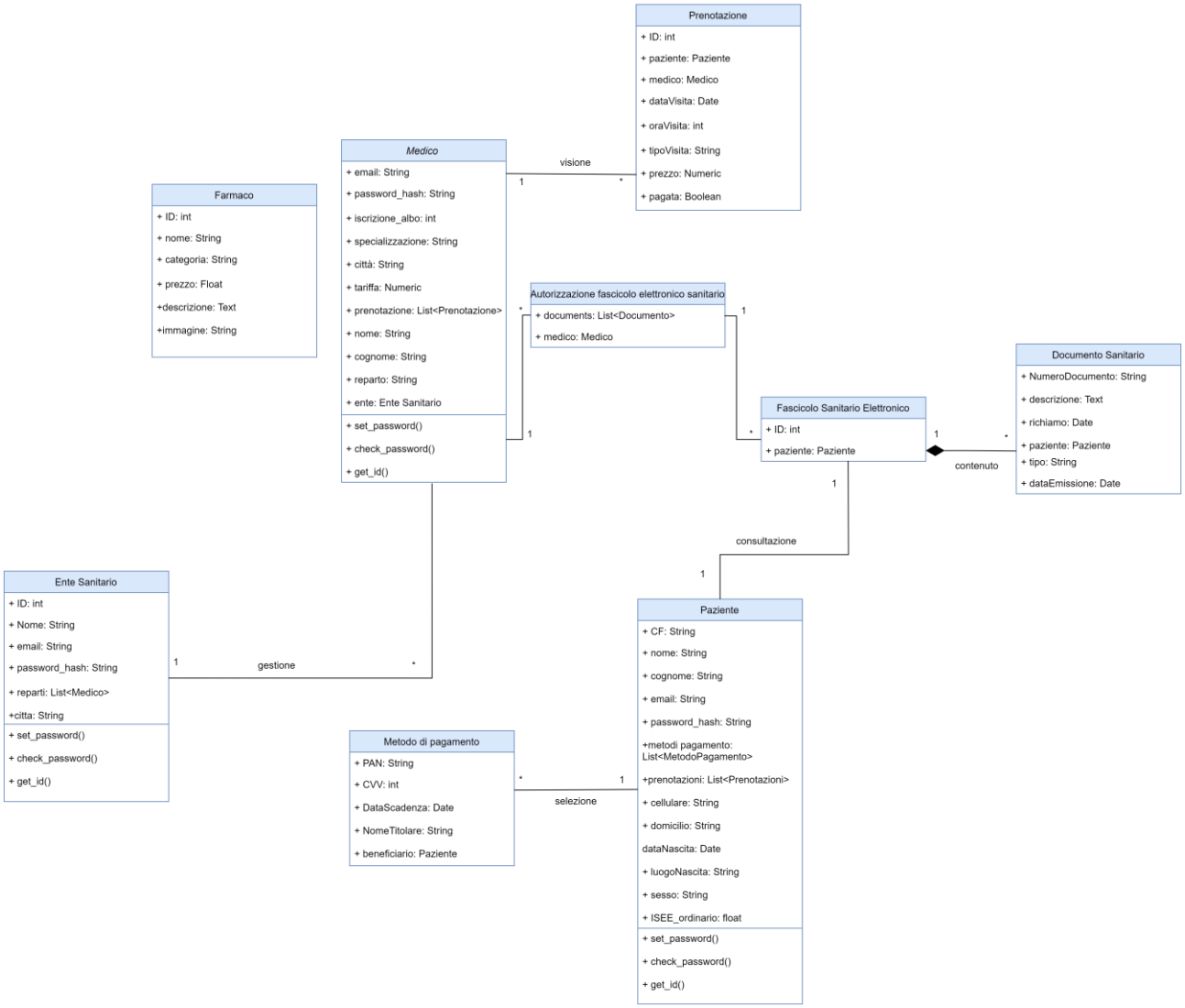
## Package AI

| **Nome classe** | AIService |
|---|---|
| **Descrizione** | Questa classe fornisce tutte le funzionalità relative alla diagnostica di malattia cardiaca |
| **Metodi** | + diagnosi_accurata(int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak, int slp, int caa, int thall): Boolean<br>+ diagnosi_simple(int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak): Boolean |
| **Invariante di classe** | / |

| Nome Metodo | + diagnosi_accurata(int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak, int slp, int caa, int thall) |
|---|---|
| Descrizione | Questo metodo permette all'utente di ottenere il risultato della diagnosi cardiaca. |
| Pre-condizione | **Context**:AIService::diagnosi_accurata (int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak, int slp, int caa, int thall)<br>**Pre**:Current_user.is_authenticated==True |
| Post-condizione | **Context**:AIService:::diagnosi_accurata (int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak, int slp, int caa, int thall)<br>**Post**:result!=Null |

| Nome Metodo | + diagnosi_simple(int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak) |
|---|---|
| Descrizione | Questo metodo permette all'utente di ottenere il risultato della diagnosi cardiaca solo tramite i dati strettamente necessari. |
| Pre-condizione | **Context**:GestioneAI::diagnosi_simple (int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak)<br>**Pre**:Current_user.is_authenticated==True |
| Post-condizione | **Context**:GestioneAIService:: diagnosi_simple (int age, int sex, int cp, int trtbps, int chol, int fbs, restecg, int thalach, int exng, int oldpeak)<br>**Post**:result!=Null |

# 4. Class Diagram Ristrutturato



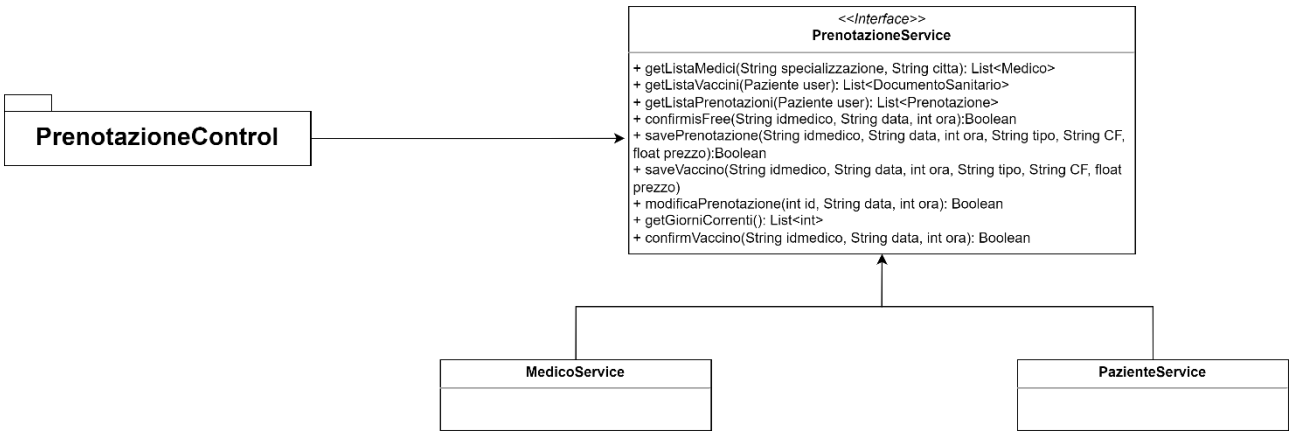

# 5. Elementi di Riuso

## 5.1 Design Pattern usati

### Facade

Facade è un design pattern strutturale che fornisce un'interfaccia unificata per un insieme di interfacce in un sottosistema. Verrà utilizzato per fornire un'interfaccia per le prenotazioni che faciliterà la gestione di esse all'utente, dato che sono presenti varie tipologie di prenotazioni (analisi, visite e vaccini), infatti esso renderà semplice l'accesso alle funzioni relative al medico ed al paziente oltre al salvataggio di esse. Questo

favorisce il disaccoppiamento e rende più manutenibile ed estendibile il codice sorgente del sistema favorendo i design goal stabiliti in fase di System Design. Medicare, quindi, utilizzerà il Facade per rendere più semplice l'interazione tra l'utente e l'interfaccia.




## 5.2 Componenti terzi

All'interno del sistema sono utilizzate le seguenti componenti COTS:

1. **MySQL**: MySQL è un sistema di gestione di database relazionali (RDBMS) open-source. Utilizzato nel sistema per la gestione dei dati persistenti.

# 6. Glossario

| Sigla / Termine | Definizione |
|---|---|
| **Package** | Raggruppamento di classi ed interfacce |
| **Facade** | Facade è un design pattern strutturale che fornisce un'interfaccia unificata per un insieme di interfacce in un sottosistema. |
| **MySQL** | MySQL è un sistema di gestione di database relazionali open source ampiamente utilizzato per la creazione, la gestione e l'interrogazione di basi di dati. Rappresenta una soluzione affidabile e veloce per lo storage e il recupero efficiente delle informazioni |
| **Componenti COTS** | Il termine componenti COTS si riferisce a quelle componenti hardware e software disponibili sul mercato al fine di poter essere utilizzati in progetti di aziende di sviluppo |

| | |
|---|---|
| **DAO** | Design pattern architetturale per la gestione della persistenza |
| **Controller** | Classe che gestisce le richieste del client |
| **Service** | Classe che implementa la logica di business |
| **Model** | Insieme dei metodi che permettono l'accesso ai dati |