



**Object Design Document**

| **Riferimento** | C15\_ODD\_0.3 |
| --- | --- |
| **Versione** | 0.3 |
| **Data** | 23/12/2022 |
| **Destinatario** | Prof.ssa F.Ferrucci, Prof F.Palomba |
| **Presentato da** | V.M.Arnone, M.Cicalese, L.Todisco, P.C.Valletta, C.Venditto, A.Zoccola, M.Calenda, R.Rapacciuolo |
| **Approvato da** | M.Calenda, R.Rapacciuolo |

Storia delle revisioni

| **Data** | **Versione** | **Descrizione** | **Autori** |
| --- | --- | --- | --- |
| 02/12/2022 | 0.1 | Prima stesura | Tutto il team |
| 22/12/2022 | 0.2 | Identificazione Packages | VA, MCI, LT, PV, CV, AZ |
| 23/12/2022 | 0.3 | Identificazione Design Patterns e COTS | VA, MCI, LT, PV, CV, AZ |
| 27/12/2022 | 0.4 | Aggiunta diagrammi Design Patterns | VA, MCI, LT, CV, AZ |
| 06/02/2023 | 0.5 | Aggiunta di ulteriori dettagli ai capitoli | VA, MCI, LT, PV, CV, AZ |
| 12/02/2023 | 1.0 | Review finale | Tutto il team |

Membri del team

| **Nome** | **Acronimo** | **Informazioni di contatto** |
| --- | --- | --- |
| Marco Calenda | MC | m.calenda10@studenti.unisa.it |
| Raimondo Rapacciuolo | RR | r.rapacciuolo1@studenti.unisa.it |
| Vincenzo Maria Arnone | VA | v.arnone4@studenti.unisa.it |
| Mario Cicalese | MCI | m.cicalese21@studenti.unisa.it |
| Leopoldo Todisco | LT | l.todisco4@studenti.unisa.it |
| Paolo Carmine Valletta | PV | p.valletta2@studenti.unisa.it |
| Carlo Venditto | CV | c.venditto@studenti.unisa.it |
| Alessandro Zoccola | AZ | a.zoccola2@studenti.unisa.it |

Sommario

[**1 - Introduzione**](#_heading=h.2pjkazwv3i4r) **5**

[1.1 - Object Design Goals](#_heading=h.8pvtehfw783v) 5

[1.2 - Linee guida](#_heading=h.97lmb99rtp) 5

[1.3 - Definizioni, acronimi e abbreviazioni](#_heading=h.8bhdwq7dto8) 6

[1.3 - Riferimenti](#_heading=h.yy25gbm4gzi5) 6

[**2 - Packages**](#_heading=h.hno12m5ur0f6) **7**

[2.1 - HeartCare](#_heading=h.xnlmmyazzhuq) 8

[2.2 - registrazione](#_heading=h.6uhzhbx2mca) 8

[2.3 - gestioneUtente](#_heading=h.xi2184f9hu9j) 9

[2.4 - gestioneMisurazione](#_heading=h.9dbs73h1zb3p) 10

[2.5 - gestioneVisita](#_heading=h.4413v5jtw49o) 11

[2.6 - gestioneComunicazione](#_heading=h.upemrjrgymny) 11

[**3 - Class Interfaces**](#_heading=h.m8udt7y23ywv) **12**

[3.1 - registrazione](#_heading=h.8qo7gvcxt012) 12

[3.2 - gestioneUtente](#_heading=h.u8qoif1wngld) 16

[3.3 - gestioneMisurazione](#_heading=h.hpdi1889d5j8) 25

[3.4 - gestioneVisita](#_heading=h.l0eytv7gfzq0) 29

[3.5 - gestioneComunicazione](#_heading=h.jdtr43cdkvvi) 31

[**4 - Class Diagram Ristrutturato**](#_heading=h.89k5p4bzof9s) **33**

[**5 - Elementi di Riuso**](#_heading=h.cu8c8iqvn1ka) **34**

[5.1 - Design Pattern Usati](#_heading=h.u7qvn3jmcr0p) 34

[5.1.1 - Facade](#_heading=h.u4k3jopgn4rh) 34

[5.1.2 - Adapter](#_heading=h.karx18bw40xe) 35

[5.1.3 - Observer](#_heading=h.4h96a8bhil39) 36

[5.2 - Identificazione COTS](#_heading=h.o41jgw26zfef) 37

[**6 - Glossario**](#_heading=h.2p3txhv9gkm8) **38**

# 1 - Introduzione

HeartCare è una piattaforma che nasce con lo scopo di fornire uno strumento di telemonitoraggio ai pazienti affetti da malattie cardiache in modo da poter tenere sotto osservazione i loro valori e notificare loro preventivamente in merito a possibili problemi grazie all'applicazione dell'Intelligenza Artificiale. Il paziente potrà, tramite le misurazioni, aggiornare il proprio Fascicolo Sanitario Elettronico che sarà accessibile al medico. Medico e paziente potranno comunicare tramite dei messaggi asincroni.

## 1.1 - Object Design Goals

Per il progetto HeartCare gli obiettivi di design posti sono:

* **Affidabilità & Safety**: Per evitare malfunzionamenti del sistema saranno implementati controlli sui dati lato backend e frontend per i requisiti funzionali più importanti come l’avvio del monitoraggio. Infine, prima di rilasciare l’applicativo, andrebbero superate delle fasi di validazione di terze parti allo scopo di ottenere una certificazione.
* **Sicurezza**: il sistema fornirà un elevato grado di sicurezza tramite setup di Spring Security e utilizzo di Token di autenticazione per le operazioni critiche.
* **Robustezza**: Verrà eseguita una test-suite per assicurarsi che il sistema non presenti malfunzionamenti in situazioni limite.
* **Manutenibilità**: La manutenibilità del codice viene assicurata tramite utilizzo di adeguati Design Patterns, la scrittura della javadoc e dei documenti in generale e la leggibilità del codice viene assicurata dall’adozione delle regole del checkstyle di SUN.

## 1.2 - Linee guida

Le linee guida per la scrittura del codice si attengono alla seguente convenzione, di cui sono riportati i dettagli al link che segue:

* **Sun Checkstyle:** <https://checkstyle.sourceforge.io/sun_style.html>

## 1.3 - Definizioni, acronimi e abbreviazioni

* **DAO**: Data Access Object
* **UML:** Unified Modeling Language
* **Architettura Three Tier**
* **Javadoc**
* **Package**
* **Design Pattern**
* **Java Spring MVC**
* **ReactJS**
* **JWT:** Json Web Token
* **API**: Application Programming Interface

## 1.3 - Riferimenti

* [Statement of Work](https://docs.google.com/document/d/1Z1Aepr4wo-y5f4-UBjetHFvpiYkanJSPHXE1pwrZsG4/edit?usp=share_link)
* [Business Case](https://docs.google.com/document/d/1h45Hi5hHq6iG3cD-crkg-n6efDCVH4UQLA6nzUAzuR8/edit?usp=share_link)
* [Matrice di tracciabilità dei requisiti](https://docs.google.com/spreadsheets/u/0/d/16rDP_kti7dTKZiCxFbLzf8ads_Y0AvSnRsLxr4SiXno/edit)
* [System Design Document](https://docs.google.com/document/d/10p-c3s2oG-JiTeYvjPgLaJooLPLA4ux4/edit)
* Manuale Utente

# 

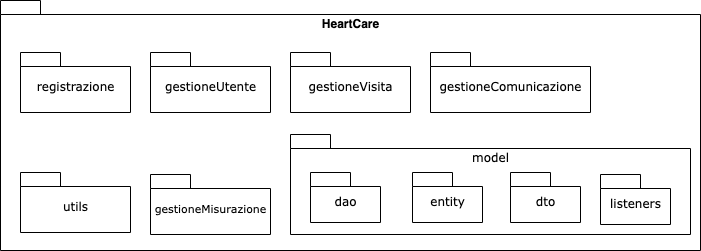
# 2 - Packages

In questa sezione del documento viene mostrata la divisione del sistema in packages.

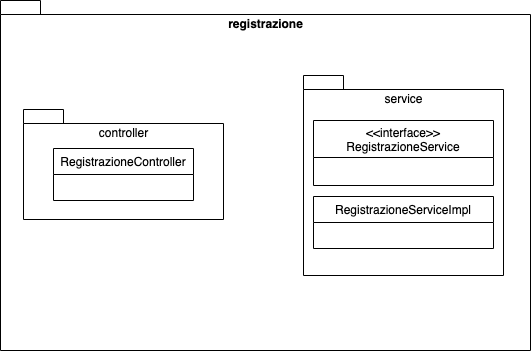
* HeartCare
  + .idea
  + .mvn
  + target
  + src, contiene i file sorgente
    - main
      * java
      * resources
        + static
        + templates
    - test
      * java
  + frontend
    - src
      * components
      * css
      * images
      * pages
    - public
    - node\_modules
    - wrapper
  + system\_testing

## 2.1 - HeartCare

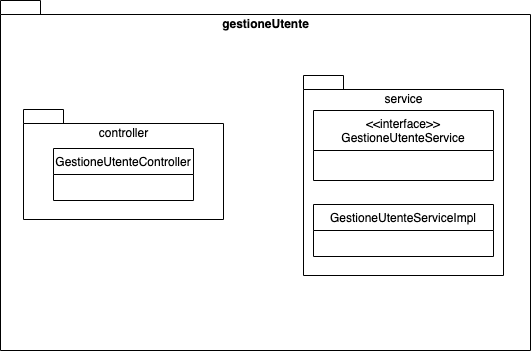
HeartCare è Il package principale e contiene al suo interno tutti gli altri packages che derivano dai sottosistemi individuati nel System Design Document e i packages in cui si trovano funzioni di utilità.



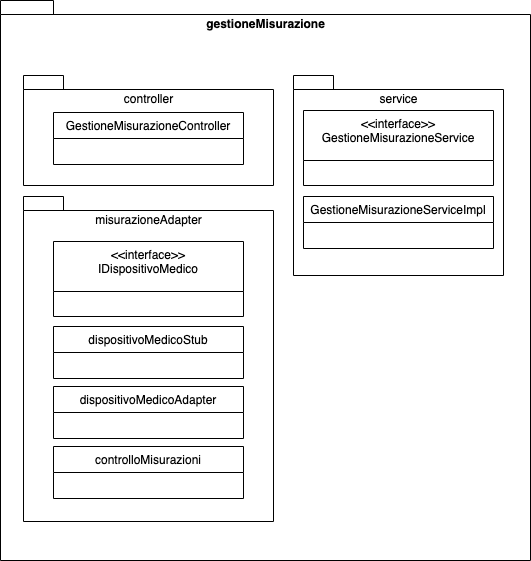
## 2.2 - registrazione

****

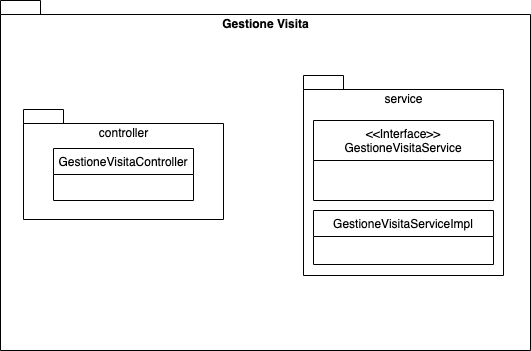
## 2.3 - gestioneUtente



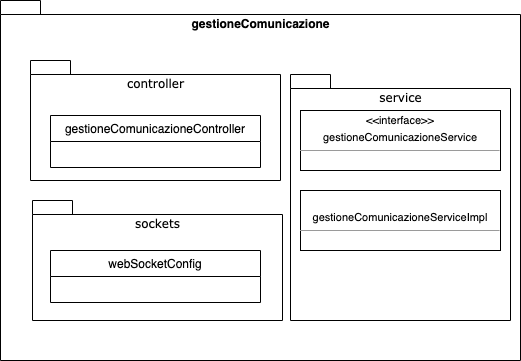
## 2.4 - gestioneMisurazione



## 2.5 - gestioneVisita

****

## 2.6 - gestioneComunicazione



# 3 - Class Interfaces

In questa sezione sono esposte le interfacce di ogni package.

## 3.1 - registrazione

| **Nome classe** | RegistrazioneService |
| --- | --- |
| **Descrizione** |  |
| **Metodi** | +registraPaziente(Paziente paziente): AuthenticationResponse throws Exception  +registraMedico(Medico medico): AuthenticationResponse throws Exception  +saveIndirizzo(Indirizzo ind): void  +login(AuthenticationRequest req): AuthenticationResponse  +findByemail(String email): Paziente  +findBycodiceFiscale(String codiceFiscale): Paziente  +registraAdmin(Admin admin): AuthenticationResponse |
| **Invariante di classe** |  |

| **Nome Metodo** | +registraPaziente(Paziente paziente, String confermaPassword): AuthenticationResponse throws Exception |
| --- | --- |
| **Descrizione** | Metodo che permette la registrazione di un nuovo paziente alla piattaforma. |
| **Pre-condizione** | **context**: RegistrazioneService::registraPaziente(Paziente paziente)  **pre**:  Paziente <> null && confermaPassword.size()>0 |
| **Post-condizione** | **context**: RegistrazioneService::registraPaziente(Paziente paziente)  **post**: UtenteRegistrato.findAll().contains(paziente) |

| **Nome Metodo** | +registraMedico(Medico medico): AuthenticationResponse throws Exception |
| --- | --- |
| **Descrizione** | Metodo che permette la registrazione di un nuovo medico alla piattaforma. |
| **Pre-condizione** | **context**: RegistrazioneService::registraMedico(Medico medico)  **pre**: medico <> null |
| **Post-condizione** | **context**: RegistrazioneService::registraMedico(Medico medico)  **post**: UtenteRegistrato.findAll().contains(medico) |

| **Nome Metodo** | +saveIndirizzo(Indirizzo ind): void |
| --- | --- |
| **Descrizione** | Metodo che permette di salvare un indirizzo. |
| **Pre-condizione** | **context**: RegistrazioneService::saveIndirizzo(Indirizzo ind): void  **pre**: ind <> null |
| **Post-condizione** | **context**: RegistrazioneService::saveIndirizzo(Indirizzo ind): void  **post**: Indirizzo.findAll().contains(ind) |

| **Nome Metodo** | +login(AuthenticationRequest req): AuthenticationResponse |
| --- | --- |
| **Descrizione** | Metodo che permette di accedere alla piattaforma. |
| **Pre-condizione** | **context**: RegistrazioneService::login(AuthenticationRequest req): AuthenticationResponse  **pre**: req <> null |
| **Post-condizione** | **context**: RegistrazioneService::login(AuthenticationRequest req): AuthenticationResponse  **post**:  AuthenticationResponse <> null |

| **Nome Metodo** | +findByemail(String email): Paziente |
| --- | --- |
| **Descrizione** | Metodo che permette di trovare un paziente tramite email. |
| **Pre-condizione** | **context**: RegistrazioneService::findByemail(String email): Paziente  **pre**:  email <> null |
| **Post-condizione** | **context**: RegistrazioneService::findByemail(String email): Paziente  **post**: / |

| **Nome Metodo** | +findBycodiceFiscale(String codiceFiscale): Paziente |
| --- | --- |
| **Descrizione** | Metodo che permette di trovare un paziente tramite il codice fiscale. |
| **Pre-condizione** | **context**: RegistrazioneService::findBycodiceFiscale(String codiceFiscale): Paziente  **pre**:  codiceFiscale <> null |
| **Post-condizione** | **context**: RegistrazioneService::findBycodiceFiscale(String codiceFiscale): Paziente  **post**: / |

| **Nome Metodo** | +registraAdmin(Admin admin): AuthenticationResponse |
| --- | --- |
| **Descrizione** | Metodo che permette di trovare un paziente tramite il codice fiscale. |
| **Pre-condizione** | **context**: RegistrazioneService::registraAdmin(Admin admin): AuthenticationResponse  **pre**:  admin <> null && |
| **Post-condizione** | **context**: RegistrazioneService::registraAdmin(Admin admin): AuthenticationResponse  **post**:  UtenteREgistrato.findAll().contains(admin) |

### 

## 3.2 - gestioneUtente

| **Nome Classe** | GestioneUtenteService |
| --- | --- |
| **Descrizione** |  |
| **Metodi** | +assegnaCaregiver(Long idPaziente, String emailCaregiver, String nomeCaregiver, String cognomeCaregiver): void  +rimuoviMedico(Long idUtente): void  +isPaziente(Long idUtente): boolean  +isMedico(Long idUtente): boolean  +isAdmin(Long idUtente): boolean  +assegnaPaziente(Long idMedico, Long idPaziente): boolean  +findPazienteById(Long id): Paziente  +findMedicoById(Long id): Medico  +findUtenteByCf(String codiceFiscale): boolean  +updatePaziente(Paziente paz): void  +updateUtente(UtenteRegistrato u): void  +getTuttiMedici(): List <UtenteRegistrato>  +getTuttiPazienti(): List <UtenteRegistrato>  +getTuttiUtenti(): List <UtenteRegistrato>  +getPazientiByMedico(Long idMedico): List<Paziente>  +findUtenteById(Long id): UtenteRegistrato  +findUtenteByEmail(String email): UtenteRegistrato  +checkByEmail(String email): boolean  +registraIndirizzo(Indirizzo ind): boolean  +assegnaIndirizzoAdUtente(Long idUtente, Indirizzo ind): boolean  +assegnaMedicoAPaziente(Long idMedico, Long idUtente): boolean  +getDispositiviByPaziente(Long idPaziente): Set<DispositivoMedico>  +updateIndirizzo(Indirizzo ind): void  +findMedicoByPaziente(Long idPaziente): long  +findAllIndirizzi(): List<Indirizzo>  +rimuoviUtente(Long idUtente): void  +controllaPassword(String pwd, Long idAdmin): boolean  +String encryptPassword(String nuovaPassword): String |
| **Invariante di classe** | / |

| **Nome Metodo** | **+assegnaCaregiver(Long id, String emailCaregiver, String nomeCaregiver, String cognomeCaregiver)** |
| --- | --- |
| **Descrizione** | Metodo che permette all’utente di assegnarsi un Caregiver |
| **Pre-condizione** | **context:**GestioneUtenteService::assegnaCaregiver(Long id,String nomeCaregiver,String cognomeCaregiver, String emailCaregiver)  **pre:** id <> null |
| **Post-condizione** | **context:**GestioneUtenteService::assegnaCaregiver( Long id,String nomeCaregiver,String cognomeCaregiver, String emailCaregiver)  **post:** Paziente.getNomeCaregiver==nomeCaregiver &&  UtenteRegistrato.getCognomeCaregiver==cognomeCaregiver && UtenteRegistrato.getEmailCaregiver==emailCaregiver |
| **Nome metodo** | **+rimuoviMedico(Long idUtente): void** |
| **Descrizione** | Metodo che permette di rimuovere un medico. |
| **Precondizione** | **context:**GestioneUtenteService::rimuoviMedico(Long idUtente)  **Pre:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::rimuoviMedico(Long idUtente)  **post:** medico.delete(utente) |
| **Nome metodo** | **+isPaziente(Long idUtente): boolean** |
| **Descrizione** | Metodo che permette di controllare se un utente è un paziente. |
| **Pre-condizione** | **context:** GestioneUtenteService::isPaziente(Long IdUtente)  **context:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::isPaziente(Long idUtente)  **post: /** |
| **Nome metodo** | **+isMedico(Long idUtente): boolean** |
| **Descrizione** | Metodo che permette di controllare se un utente è un medico. |
| **Precondizione** | **context:** GestioneUtenteService::isMedico(Long IdUtente)  **post:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::isMedico(Long idUtente)  **post:** / |
| **Nome metodo** | **+isAdmin(Long idUtente): boolean** |
| **Descrizione** | Metodo che permette di controllare se un utente è un admin. |
| **Pre-condizione** | **context:** GestioneUtenteService::isAdmin(Long IdUtente)  **context:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::isAdmin(Long idUtente)  **post:** / |
| **nome metodo** | **+getTuttiUtenti(): List<UtentiRegistrato>** |
| **Descrizione** | Metodo che restituisce tutti gli utenti |
| **pre-condizione** | **context:** GestioneUtenteService::getTuttiUtenti()  **pre:** / |
| **post-condizione** | **context:** GestioneUtenteService::getTuttiUtenti()  **post:** listaUtenti<Utenti> |
| **Nome metodo** | **+getPazientiByMedico(Long idMedico)List<Paziente>** |
| **Descrizione** | Metodo che restituisce tutti i pazienti assegnati ad un medico. |
| **Precondizione** | **context:**GestioneUtenteService::getPazientiByMedico(Long idMedico)  **pre:** idMedico <> null |
| **Post-condizione** | **context:**GestioneUtenteService::getPazientiByMedico(Long idMedico)  **post:** listaPazienti<> null |
| **Nome Metodo** | **+assegnaPaziente(Long idMedico, Long idPaziente): boolean** |
| **Descrizione** | Metodo che dato un id del medico e un id del paziente restituisce true solo se riesce ad assegnare il paziente al medico. |
| **Pre-condizione** | **context:**GestioneUtenteService::assegnaPaziente(Long idMedico, Long idPaziente)  **pre:** idMedico <> null && idPaziente <> null |
| **Post-condizione** | **context:**GestioneUtenteService::assegnaPaziente(Long idMedico, Long idUtente)  **post:** @pre medico.getElencoPazienti.size()+1 |
| **Nome metodo** | **+findPazienteById(long id): Paziente** |
| **Descrizione** | Metodo che dato un id restituisce un paziente corrispondente a quell’id. |
| **Pre-condizione** | **context:** GestioneUtenteService::findPazienteById(Long id)  **pre:** id <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findPazienteById(Long id)  **post:** paziente<>null |
| **Nome metodo** | **+findMedicoById(long id): Medico** |
| **Descrizione** | Metodo che dato un id restituisce un medico corrispondente a quell’id. |
| **Precondizione** | **context:** GestioneUtenteService::findMedicoById(Long id)  **pre:** id <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findMedicoById(Long id)  **post:** medico <> null |
| **Nome metodo** | **+findUtenteByCf(String codiceFiscale): boolean** |
| **Descrizione** | Metodo che dato un codice fiscale restituisce true se trova un utente corrispondente a quell’cf. |
| **Precondizione** | **context:** GestioneUtenteService::findUtenteByCf(String codiceFiscale)  **pre:** codiceFiscale <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findUtenteByCf(String codiceFiscale)  **post:** paziente <> null |
| **Nome metodo** | **+updatePaziente(Paziente paz): void** |
| **Descrizione** | Metodo che aggiorna il paziente passato in input. |
| **Precondizioni** | **/** |
| **Post-condizione** | **context:** GestioneUtenteService::UpdatePaziente(Paziente paz)  **post:** / |
| **Nome metodo** | **+updateUtente(UtenteRegistrato u): void** |
| **Descrizione** | Metodo che aggiorna l’utente passato in input. |
| **Precondizioni** | **/** |
| **Post-condizione** | **context:** GestioneUtenteService::UpdateUtente(UtenteRegsitrato u)  **post:** / |
| **nome metodo** | **+getTuttiPazienti(): List<Pazienti>** |
| **Descrizione** | Metodo che restituisce tutti i pazienti |
| **pre-condizione** | **context:** GestioneUtenteService::getTuttiPazienti()  **pre:** / |
| **post-condizione** | **context:** GestioneUtenteService::getTuttiPazienti()  **post:** listaPazienti<Pazienti> |
| **nome metodo** | **+getTuttiMedici(): List<Medici>** |
| **Descrizione** | Metodo che restituisce tutti i medici |
| **pre-condizione** | **context:** GestioneUtenteService::getTuttiMedici()  **pre:** / |
| **post-condizione** | **context:** GestioneUtenteService::getTuttiMedici()  **post:** listaMedici<Medici> |
| **nome metodo** | **+getPazientiByMedico(long idMedico): List<Paziente>** |
| **Descrizione** | Metodo che restituisce tutti i pazienti di un medico |
| **pre-condizione** | **context:** GestioneUtenteService::getPazientiByMedico(Long idMedico)  **pre:** idMedico <> null |
| **post-condizione** | **context:** GestioneUtenteService::getPazientiByMedico(Long idMedico)  **post:** listaPazienti<pazienti> |
| **Nome metodo** | **+findUtenteById(long id): UtenteRegistrato** |
| **Descrizione** | Metodo che dato un id restituisce un utente corrispondente a quell’id. |
| **Pre-condizione** | **context:** GestioneUtenteService::findUtenteById(Long id)  **pre:** id <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findutenteById(Long id)  **post:** utenteRegistrato |
| **Nome metodo** | **+findUtenteByEmail(String email): UtenteRegistrato** |
| **Descrizione** | Metodo che dato un email restituisce un utente corrispondente a quella email. |
| **Pre-condizione** | **context:** GestioneUtenteService::findUtenteByEmail(string email)  **pre:** utente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findUtenteByEmail(String email)  **post:** utenteRegistrato |
| **Nome metodo** | **+checkByEmail(String email): boolean** |
| **Descrizione** | Metodo che dato un email restituisce true se trova un utente corrispondente a quella email. |
| **Pre-condizione** | **context:** GestioneUtenteService::checkByEmail(string email)  **pre:** email <> null |
| **Post-condizione** | **context:** GestioneUtenteService::checkByEmail(String email)  **post:** / |
| **Nome metodo** | **+registraIndirizzo(indirizzo ind): boolean** |
| **Descrizione** | Metodo che salva l’indirizzo. |
| **Precondizioni** | **context:** GestioneUtenteService::registraIndirizzo(Indirizzo ind)  **pre:** ind <> null |
| **Post-condizione** | **context:** GestioneUtenteService::registraIndirizzo(Indirizzo ind)  **post:** / |
| **Nome metodo** | **+assegnaIndirizzoAdUtente(long idUtente, Indirizzo ind): boolean** |
| **Descrizione** | Metodo che assegna l’indirizzo all’utente. |
| **Precondizioni** | **context:** GestioneUtenteService::assegnaIndirizzoAdUtente(long idUtente, indirizzo ind)  **pre:** id <> null && ind <> null |
| **Post-condizione** | **context:** GestioneUtenteService::assegnaIndirizzoAdUtente(long idUtente, indirizzo ind)  **post:** UtenteRegistrato.getIndirzzo()<> null |
| **Nome Metodo** | **+assegnaMedicoAPaziente(Long idMedico, Long idPaziente): boolean** |
| **Descrizione** | Metodo che dato un id del medico e un id del paziente restituisce true solo se riesce ad assegnare il medico al paziente. |
| **Pre-condizione** | **context:**GestioneUtenteService::assegnaMedicoAPaziente(Long idMedico, Long idPaziente)  **pre:** idMedico <> null && idPaziente <> null |
| **Post-condizione** | **context:**GestioneUtenteService::assegnaMedicoAPaziente(Long idMedico, Long idUtente)  **post:** medico.getAllPazienti().size+1 |
| **Nome metodo** | **+getDispositiviByPaziente(Long idPaziente): Set<DispositivoMedico>** |
| **Descrizione** | Metodo che restituisce tutti i dispositivi di un paziente. |
| **Precondizione** | **context:**GestioneUtenteService::getDispositviByPazienti(Long idPaziente)  **pre:** idPaziente <> null |
| **Post-condizione** | **context:**GestioneUtenteService::getDispositiviByPazienti(Long idPaziente)  **post:** hashSet<> null |
| **Nome metodo** | **+updateIndirizzo(Indirizzo ind): void** |
| **Descrizione** | Metodo che aggiorna l’indirizzo passato in input. |
| **Precondizioni** | **context:** GestioneUtenteService::registraIndirizzo(Indirizzo ind)  **pre:** ind <> null |
| **Post-condizione** | **context:** GestioneUtenteService::UpdateIndirizzo(Indirizzo ind)  **post:** / |
| **Nome metodo** | **+findMedicoByPaziente(long idPaziente): Long** |
| **Descrizione** | Metodo che dato un id del paziente restituisce l’id del suo medico. |
| **Pre-condizione** | **context:** GestioneUtenteService::findMedicoByPaziente(Long idPaziente)  **pre:** idPaziente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::findMedicoByPaziente(Long id Paziente)  **post:** idMedico |
| **nome metodo** | **+findAllIndirizzi(): List<Indirizzo>** |
| **Descrizione** | Metodo che restituisce tutti gli indirizzi |
| **pre-condizione** | **context:** GestioneUtenteService::findAllIndirizzi()  **pre:** / |
| **post-condizione** | **context:** GestioneUtenteService::findAllIndirizzi()  **post:** listaIndirizzi<Indirizzi> |
| **Nome metodo** | **+rimuoviUtente(Long idUtente): void** |
| **Descrizione** | Metodo che permette di rimuovere un utente. |
| **Pre-condizione** | **context:**GestioneUtenteService::rimuoviUtente(Long idUtente)  **Pre:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::rimuoviUtente(Long idUtente)  **post:** / |
| **Nome metodo** | **+controllaPassword(String pwd, Long idUtente): boolean** |
| **Descrizione** | Metodo che permette di controllare che la password passata sia uguale a quella del db. |
| **Pre-condizione** | **context:**GestioneUtenteService::controllaPassword(String pwd, Long idUtente)  **Pre:** idUtente <> null && pwd <> null |
| **Post-condizione** | **context:** GestioneUtenteService::controllaPassword(String pwd, Long idUtente)  **post:** pwd <> null = true |
| **Nome metodo** | **+encryptPassword(String nuovaPassword): string** |
| **Descrizione** | Metodo che permette di decriptare la nuova password. |
| **Pre-condizione** | **context:**GestioneUtenteService::encryptPassword(String nuovaPassword)  **Pre:** idUtente <> null |
| **Post-condizione** | **context:** GestioneUtenteService::encryptPassword(String nuovaPassword)  **post:** nuovaPassword |

## 3.3 - gestioneMisurazione

| **Nome classe** | GestioneMisurazioneService |
| --- | --- |
| **Descrizione** | Classe che mette a disposizione le firme dei metodi principali del package gestioneMisurazione. |
| **Metodi** | +getAllMisurazioniByPaziente(Long idPaziente): List<MisurazioneDTO>  +findCategorieByPaziente(Long idPaziente): List<String>  +saveMisurazione(Misurazione misurazione): Misurazione  +getMisurazioneByCategoria(String categoria, Long idPaziente): List<Misurazione>  +AvvioMonitoraggio(Long idPaziente, DispositivoMedico dispositivo)  +registrazioneDispositivo(Dispositivo dispositivo, Long idPaziente): Boolean  +rimozioneDispositivo(Dispositivo dispositivo, Long idPaziente): Boolean  +getAllDispositiviByPaziente(Long idPaziente): List<Paziente>:  +getDispositivoById(Long idDispositivo):  Dispositivo |
| **Invariante di classe** | / |

| **Nome Metodo** | **+getAllMisurazioniByPaziente(Long idPaziente)** |
| --- | --- |
| **Descrizione** | Metodo che restituisce tutte le misurazioni di un determinato paziente. |
| **Pre-condizione** | **context**:GestioneMisurazioneservice::getAllMisurazioniByPaziente(Long idPaziente)  **pre**:  isPaziente(UtenteDAO.getById(idPaziente))==true |
| **Post-condizione** | **context**:GestioneMisurazioneservice::getAllMisurazioni(Long idPaziente) |
| **Nome Metodo** | **+getMisurazioneByCategoria(String categoria, Long id): List<Misurazione>** |
| **Descrizione** | Metodo che restituisce tutte le misurazioni di un determinato tipo. |
| **Pre-condizione** | **context**:GestioneMisurazioneservice::getAllMisurazioniByTipo(String tipo,Long idPaziente)  **pre:**  isPaziente(GestioneUtenteService::getById(idPaziente)) == true  &&  (tipo == MisurazioneEnzimiCardiaci) ||  (tipo == MisurazioneSaturazione) ||  (tipo == MisurazionePressione) ||  (tipo == MisurazioneGlicemica) ||  (tipo == MisurazioneHolterECG) ||  (tipo == MisurazioneCoagulazione) |
| **Post-condizione** | **/** |
| **Nome Metodo** | **+registrazioneDispositivo(Dispositivo disp, Long idPaziente): boolean** |
| **Descrizione** | Metodo che permette di registrare un nuovo dispositivo e assegnarlo al paziente. |
| **Pre-condizione** | **context**:GestioneMisurazioneservice::registrazioneDispositivo(Dispositivo dispositivo, Long idPaziente)  **pre**:  isPaziente(UtenteDAO.getById(idPaziente)) == true && dispositivo <> null |
| **Post-condizione** | **context**:GestioneMisurazioneservice::registrazioneDispositivo(Dispositivo dispositivo, Long idPaziente)  **post:**  GestioneMisurazione::getAllDispositiviByPaziente(Long idPaziente).size()  ==@pre::getAllDispositiviByPaziente(Long idPaziente).size()+1 |
| **Nome Metodo** | **+rimozioneDispositivo(Dispositivo dispositivo, Long idPaziente): Boolean** |
| **Descrizione** | Metodo che permette di rimuovere un dispositivo di un paziente. |
| **Pre-condizione** | **context:** GestioneMisurazioneservice::rimozioneDispositivo(Dispositivo dispositivo, Long idPaziente)  **pre:** isPaziente(UtenteDAO.getById(idPaziente)) == true && getDispositivoById(Dispositivo.getId()) <> null |
| **Post-condizione** | **context:** GestioneMisurazioneservice::rimozioneDispositivo(Dispositivo dispositivo, Long idPaziente)  **post:**  GestioneMisurazione::getAllDispositiviByPaziente(Long idPaziente).size()  ==@pre::getAllDispositiviByPaziente(Long idPaziente).size()-1 |
| **Nome Metodo** | **+getById(Long idDispositivo): DispositivoMedico** |
| **Descrizione** | Metodo che permette di restituire un dispositivo dato un id. |
| **Pre-condizione** | **context:** GestioneMisurazioneService::getDispositivoById(Long idDispositivo)  **pre:** idDispositivo <> null |
| **Post-condizione** | **context:** GestioneMisurazioneService::getDispositivoById(Long idDispositivo)  **post:** Dispositivo <> null |
| **Nome Metodo** | **+save(Misurazione m): Misurazione** |
| **Descrizione** | metodo che consente di inserire una misurazione nel database. |
| **Pre-condizione** | **context:**  GestioneMisurazione.save(misurazione): Misurazione  **pre:**  misurazione <> null |
| **Post-condizione** | **context:**  GestioneMisurazione.save(misurazione): Misurazione  **post:**  GestioneMisurazione::getAllMisurazioniByPaziente(Long idPaziente).size()==@pre::getAllMisurazioniByPaziente(Long id).size+1 |
| **Nome metodo** | **findCategorieByPaziente(long id): List<String>** |
| **Descrizione** | Metodo che permette, dato un paziente, di ottenere tutte le categorie delle sue misurazioni. |
| **Pre-condizione** | **context:**  GestioneMisurazione.findCategorieByPaziente(long id): List<String>  **pre:**  id <> null |
| **Post-condizione** | **context:**  GestioneMisurazione.findCategorieByPaziente(long id): List<String>  **post: /** |

### 

## 3.4 - gestioneVisita

| **Nome classe** | GestioneVisitaService |
| --- | --- |
| **Descrizione** |  |
| **Metodi** | +aggiungiVisita(Visita visita): boolean  +findVisiteProgrammateByUser(String email): List<Visita>  +findIndirizzoById(Long id): Indirizzo  +findById(Long id): Visita  +cambiaData(Visita visita, LocalDate data): void |
| **Invariante di classe** | / |

| **Nome Metodo** | **+aggiungiVisita(Visita visita): boolean** |
| --- | --- |
| **Descrizione** | Metodo che permette l’aggiunta di una visita per un utente. |
| **Pre-condizione** | **context:** GestioneVisitaService::aggiungiVisita(Visita visita)  **pre:**  visita <> null |
| **Post-condizione** | **context**: GestioneVisitaService::aggiungiVisita(Visita visita)  **post**:  GestioneVisita::findVisiteProgrammateByUser(String email).size() == @pre::findVisiteProgrammateByUser(String email).size() + 1 |
| **Nome metodo** | **+findVisiteProgrammateByUser(String email): List<Visita>** |
| **Descrizione** | Metodo che restituisce tutte le visite programmate per un paziente. |
| **Pre-condizione** | **context:** GestioneVisitaService::findVisiteProgrammateByUser(String email)  **pre:** email <> null |
| **Post-condizione** | **context:**GestioneVisitaService::findVisiteProgrammateByUser(String email)  **post:** listaVisite<Visita> <> null |
| **Nome metodo** | **+findIndirizzoById(Long id): Indirizzo** |
| **Descrizione** | Metodo che restituisce l’indirizzo associato ad una visita. |
| **Pre-condizione** | **context:** GestioneVisitaService::findIndirizzoById(Long id)  **pre:** id <> null |
| **Post-condizione** | **context:** GestioneVisitaService::findIndirizzoById(Long id)  **post:** Indirizzo <> null |
| **Nome metodo** | **+findById(Long id): Visita** |
| **Descrizione** | Metodo che, dato un id, restituisce la visita corrispondente. |
| **Pre-condizione** | **context:** GestioneVisitaService::findById(Long idVisita)  **pre:** idVisita <> null |
| **Post-condizione** | **context:** GestioneVisitaService::findById(Long idVisita)  **post:** visita <> null |
| **Nome metodo** | **+cambiaData(Visita visita, LocalDate data): void** |
| **Descrizione** | Metodo che permette di cambiare la data della visita. |
| **Pre-condizione** | **context:** GestioneVisitaService::cambiaData(Visita visita, LocalDate data)  **pre:** visita <> null && data > data odierna |
| **Post-condizione** | **context:** GestioneVisitaService::cambiaData(Visita visita, LocalDate data)  **post:** / |

### 

## 3.5 - gestioneComunicazione

| **Nome classe** | ComunicazioneService |
| --- | --- |
| **Descrizione** | Questa classe fornisce i metodi per la comunicazione |
| **Metodi** | +invioEmail(String messaggio, String emailDestinatario): void  +invioNota(String messaggio, Long idDestinatario, Long idMittente) :void  +findAllNote(): List<Nota>  +findNoteByIdUtente(Long id): List<NotaDTO>  +findNoteNonLetteByUser(String email):List<Nota>  +sendNotifica<String message,Long idDest>:void |
| **Invariante di classe** | / |

| **Nome Metodo** | **+invioEmail(String messaggio, String emailDestinatario)** |
| --- | --- |
| **Descrizione** | Metodo che permette di inviare una e-mail. |
| **Pre-condizione** | **context:** ComunicazioneService::invioEmail(String messaggio, String emailDestinatario)  **pre:** messaggio.size() > 0 && emailDestinatario.size() > 0 |
| **Post-condizione** | / |
| **Nome Metodo** | **+sendNotifica(String messaggio, Long idDest)** |
| **Descrizione** | Metodo che permette di inviare una notifica a un utente registrato. |
| **Pre-condizione** | **context:** ComunicazioneService::sendNotifica(String messaggio, Long idDest)  **pre:**idDest<>null && messaggio<>null |
| **Post-condizione** | / |
| **Nome Metodo** | **+findAllNote(): List<Nota>** |
| **Descrizione** | Metodo che restituisce tutte le note. |
| **Pre-condizione** | **context:** ComunicazioneService::findAllNote() |
| **Post-condizione** | **context:**ComunicazioneService::findAllNote(): List<Nota>  **post:** list <> null |
| **Nome Metodo** | **+findNoteByIdUtente(Long id): List<Nota>** |
| **Descrizione** | Metodo che permette di restituire tutte le note destinate a uno specifico utente |
| **Pre-condizione** | **context:** ComunicazioneService::findNoteByIdUtente(Long id): List<Nota>  **pre:**  id <> null |
| **Post-condizione** | **context:**ComunicazioneService::findNoteByIdUtente(Long id): List<Notifica>  **post:** list <> null |
| **Nome Metodo** | **+findNoteNonLetteByUser(String email): List<Nota>** |
| **Descrizione** | Metodo che permette di restituire tutte le note non lette da uno specifico utente |
| **Pre-condizione** | **context:** ComunicazioneService::findNoteNonLetteByUser(String email): List<Nota>  **pre:** email <> null |
| **Post-condizione** | **context:**ComunicazioneService::findNoteNonLetteByUser(String email): List<Nota>  **post:** list <> null |
| **Nome Metodo** | **+invioNote(String messaggio, Long idDestinatario, Long idMittente)** |
| **Descrizione** | Metodo che permette di inviare una nota. |
| **Pre-condizione** | **context:** ComunicazioneService::invioNota(String messaggio, Long idDestinatario, Long idMittente): List<Nota>  **pre:** messaggio <> null && GestioneUtenteService.findUtenteById(idDetinatario)&& GestioneUtenteService.findUtenteById(idDetinatario) |
| **Post-condizione** | / |

# 4 - Class Diagram Ristrutturato

# 

Il diagramma in alta risoluzione è disponibile in allegato tra gli artefatti di prodotto.

# 5 - Elementi di Riuso

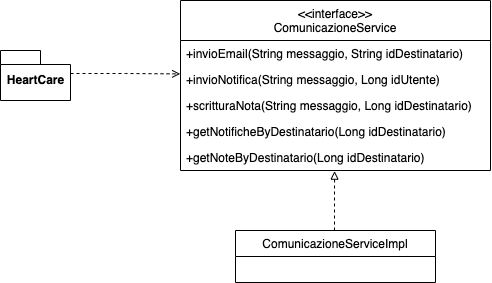
## 5.1 - Design Pattern Usati

In questa sezione sono descritti i Design Patterns usati nello sviluppo di HeartCare.

### **5.1.1 -** Facade

Il Facade è un Design Pattern che permette di minimizzare la comunicazione e la dipendenza tra sottosistemi.

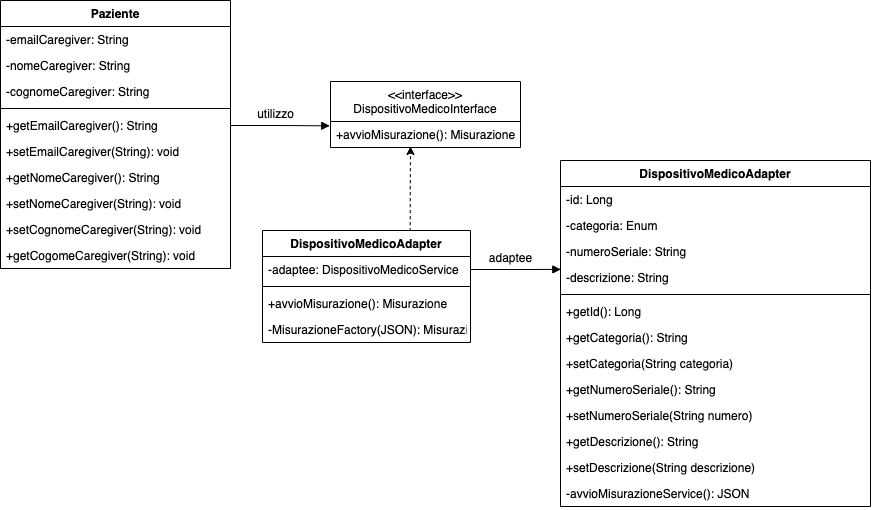
Consente di avere scarso accoppiamento tra sottosistema che offre servizi e gli utilizzatori, questo consente di modificare le componenti del sottosistema senza influenzare l’utilizzatore.  
Nel sistema Heartcare il pattern Facade è implementato per ogni sottosistema attraverso interfacce chiamate Service.



### **5.1.2 -** Adapter

L’Adapter è un Design Pattern che consente a sottosistemi/classi con interfacce incompatibili di collegarsi. Nel sistema HeartCare il pattern Adapter permetterà la creazione, a partire da un dispositivo medico, di una misurazione.

Un dispositivo medico fisico, in effetti, potrebbe strutturare i dati in modo diverso, come ad esempio JSON, in questo modo un adapter trasforma il JSON in una misurazione vera e propria.



### **5.1.3 -** Observer

L’Observer è un Design Pattern che consente di definire un meccanismo di sottoscrizione per notificare più oggetti circa l’avvenimento di eventi correlati ad uno specifico oggetto che viene “osservato”. Nel sistema HeartCare il pattern Observer sarà responsabile dell’invio di Email e notifiche quando si verificano determinati eventi. Viene implementato mediante la gestione degli eventi di Hibernate siccome se avessimo usato le classiche interfacce Subject e Observer non avremmo potuto usufruire del principio di Dependency Injection di Java Spring Boot.

## 

## 5.2 - Identificazione COTS

1. Java Spring MVC
2. ReactJS
3. Maven
4. JUnit
5. Mockito
6. Selenium
7. Jacoco
8. Google Charts API
9. react-responsive-modal
10. SockJS
11. Lombok
12. react-push-notification
13. Gson
14. Hibernate
15. JWT
16. Jackson serializer
17. React-Routes
18. Checkstyle

# 6 - Glossario

| **Termine / Sigla** | **Definizione** |
| --- | --- |
| **Piattaforma** | Infrastruttura in grado di connettere tra loro i sistemi diversi ed esporli agli utenti attraverso interfacce semplificate ed integrate, generalmente un'app mobile o un sito web. |
| **Sun Checkstyle** | Suite di strumenti per la verifica della qualità del codice Java in modo che gli sviluppatori seguano delle linee di sviluppo durante la fase di implementazione. |
| **DAO** | Pattern per la gestione della persistenza. |
| **UML** | Linguaggio di modellazione grafico usato per documentare e progettare sistemi software. |
| **Architettura Three Tier** | Architettura software che organizza un sistema in tre livelli distinti. |
| **COTS** | Component Off The Shelf, si riferisce a componenti hardware e software disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate a utilizzarli nei loro progetti. |
| **Design Pattern** | Una soluzione progettuale generale ad un problema ricorrente. |
| **Java Spring MVC** | Framework Java che è usato per sviluppare applicazioni web. |
| **Package** | Insieme di classi e altri elementi di codice che lavorano insieme per svolgere un compito specifico. |
| **React JS** | React è una libreria JavaScript per la creazione di interfacce utente basate su componenti dell'interfaccia utente. |
| **Frontend** | Parte dell’applicazione che è visibile all’utente finale o che interagisce con l’utente tramite l’interfaccia grafica. |
| **Backend** | Parte dell’applicazione non visibile all’utente e che si occupa delle attività che avvengono sul lato server. |
| **Hibernate** | E’ un framework ORM per il linguaggio Java. Si occupa della gestione della persistenza dei dati. |
| **gson** | Libreria Java per la serializzazione e deserializzazione degli oggetti JSON. |
| **Maven** | Piattaforma di automazione della build per progetti Java. |
| **JUnit** | Framework open source per il testing di unità. |
| **Mockito** | Framework open source per la creazione di oggetti simulati (Mock) usati per testare il codice. |
| **Selenium** | Framework open source per le automazioni di test di applicazioni web in modo da simulare l’interazione degli utenti con la piattaforma. |
| **Jacoco** | Strumento open source per la misurazione della copertura del codice Java. |
| **jwt** | Json Web Token, è uno standard che definisce un modo sicuro per trasmettere in modo sicuro informazioni tra diverse parti del sistema. |
| **Componente React** | Porzione riutilizzabile di codice che descrive come dovrebbe apparire e comportarsi una porzione specifica di interfaccia. |
| **Notifica Push** | Brevi messaggi inviati all’utente in maniera asincrona. |
| **Socket** | Endpoint di una connessione bidirezionale sulla rete che mette in comunicazione due macchine diverse. |
| **API** | Application Programming Interface, insieme di routine, protocolli e strumenti usati per la costruzione di software. |
| **Modal** | Una modale è una finestra di dialogo che appare sopra alla finestra principale. |
| **Google Charts** | Servizio offerto da Google che permette di creare e integrare grafici all’interno di applicazioni web. |