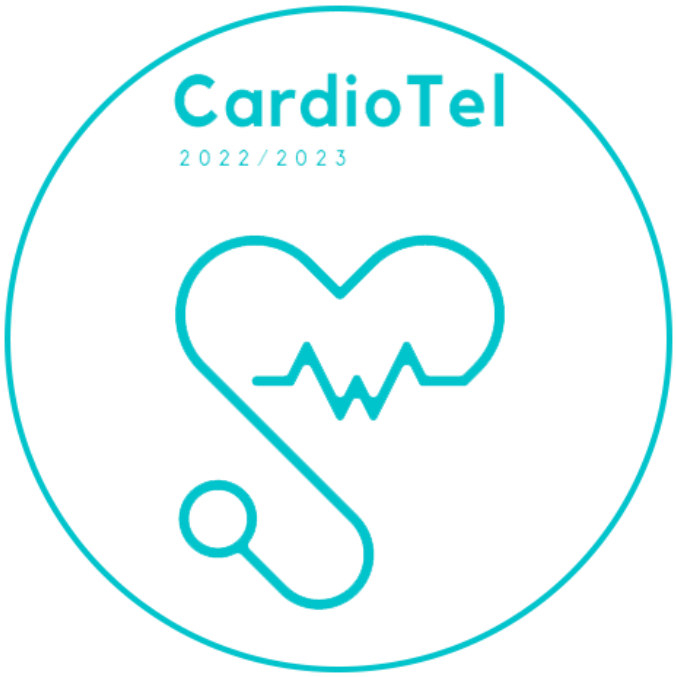


ODD

Object Design Document

Riferimento	C10_ODD_ver.1.0
Versione	1.0
Data	28/12/2022
Destinatario	Prof.ssa Filomena Ferrucci AND Prof. Fabio Palomba
Presentato da	Team C10
Approvato da	La Monica Tiziano, Bacco Alessandro

Revision History

Data	Versione	Descrizione	Autore
15/12/2022	0.1	Prima Stesura	SP
26/12/2022	0.2	Aggiunta sezione “Packages”, “Class Interfaces”	SP
28/12/2022	1.0	Aggiunta sezione “Object Design Goals”, “Design Pattern”, “COTS”, “Glossario”	SP
09/02/2023	2.0	Revisione generale	SP

Team Members

Nome	Ruolo nel progetto	Acronimo	Informazioni di contatto
Alessandro Bacco	Project Manager	AB	a.bacco10@studenti.unisa.it
Tiziano La Monica	Project Manager	TLM	t.lamonica@studenti.unisa.it
Alessio Romaniello	Team Member	AR	a.romaniello9@studenti.unisa.it
Carmine Pascale	Team Member	CP	c.pascale15@studenti.unisa.it
Francesco Laurenzano	Team Member	FL	f.laurenzano1@studenti.unisa.it
Mattia Giuseppe Giella	Team Member	MGG	m.giella4@studenti.unisa.it
Sabrina Pannullo	Team Member	SP	s.pannullo1@studenti.unisa.it

Indice

1. Introduzione

Lo scopo del software CardioTel è di fornire uno strumento di supporto ai medici assicurando che il carico di lavoro dei singoli sia focalizzato solo sui pazienti con gravi patologie in modo da gestirne il più possibile.
Di seguito vengono descritti i trade-offs e le linee guida per la fase di implementazione.

1.1 Object Design Goals

Robustezza
Il sistema deve assestare condizioni e input imprevisti attraverso il controllo degli errori e la gestione delle eccezioni.

Incapsulamento
Le interfacce del sistema nascondono l'implementazione dei dettagli, rendendo possibile l'utilizzo delle varie funzionalità sottoforma di black-box.

Modularità
Il sistema è organizzato in più blocchi funzionali che, connessi tra loro, formano un software più elaborato.

1.2 Linee guida per la documentazione dell'interfaccia

Le linee guida riportano regole da rispettare in fase di progettazione delle interfacce. Per la costruzione di queste si è fatto riferimento alla convenzione **Sun Java Coding Convention** [Sun, 2009].

1.3 Definizioni, acronimi e abbreviazioni

Vengono riportati di seguito alcune definizioni presenti nel documento:

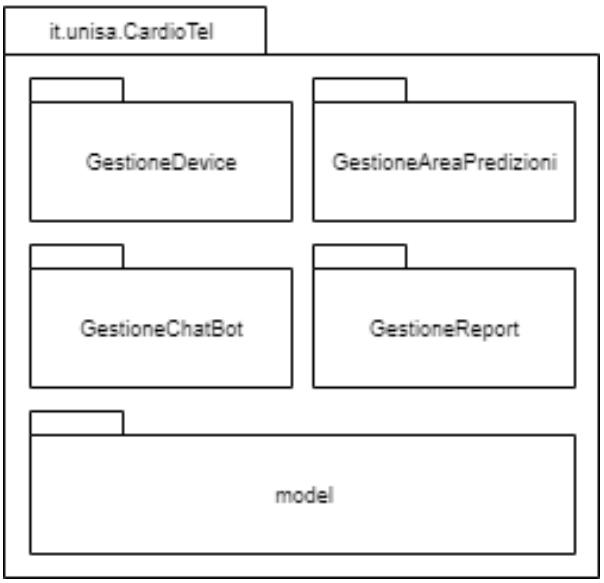
- **Package:** raggruppamento di classi, interfacce o file correlati tra loro;
- **COTS**: componenti disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate a utilizzarli nei loro progetti;
- **Design Pattern:** soluzione progettuale per problemi ricorrenti;
- **Interfaccia**: insieme di firma delle operazioni offerte dalla classe;
- **DB**: base di dati.

1.4. Riferimenti

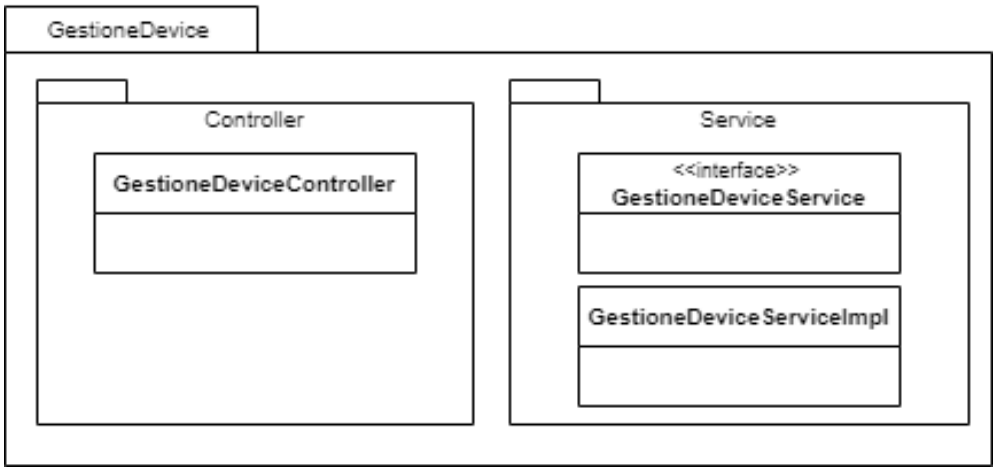
- Statement Of Work
- Business Case
- Requirements Analysis Document
- System Design Document
- Test Plan
- Matrice di tracciabilità
- Materiale di installazione
- Manuale utente

2. Packages

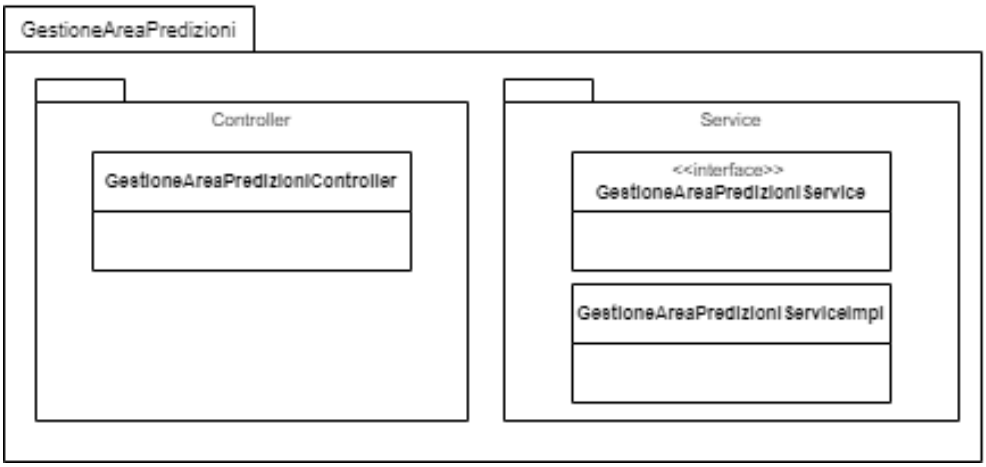
In questa e nella prossima sezione viene presentata la suddivisione del sistema in package seguendo la partizione riportata nel System Design Document.
Si è scelto di strutturare CardioTel in quattro package seguendo la suddivisione del software in quattro endpoint. Ognuno di questi sarà costituito dalle classi Control e Service. Per quanto concerne le classi Model, queste saranno implementate in un package distinto.



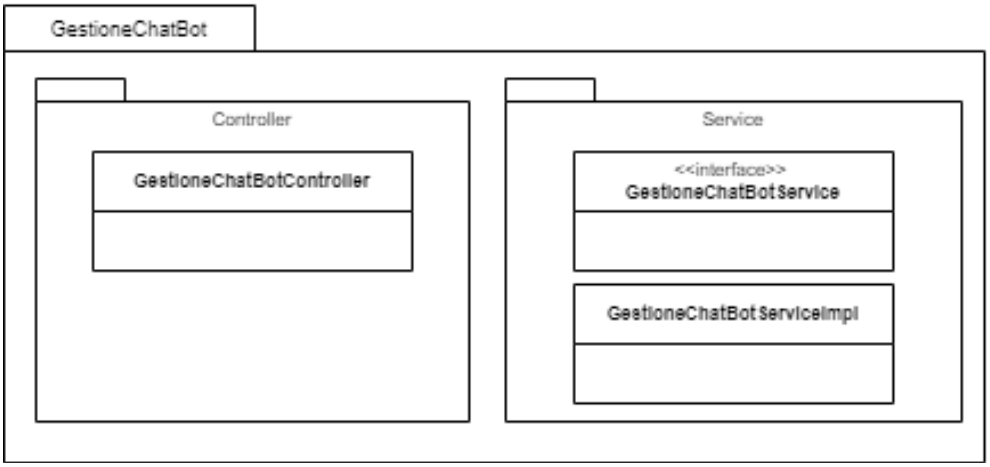
Package Gestione Device



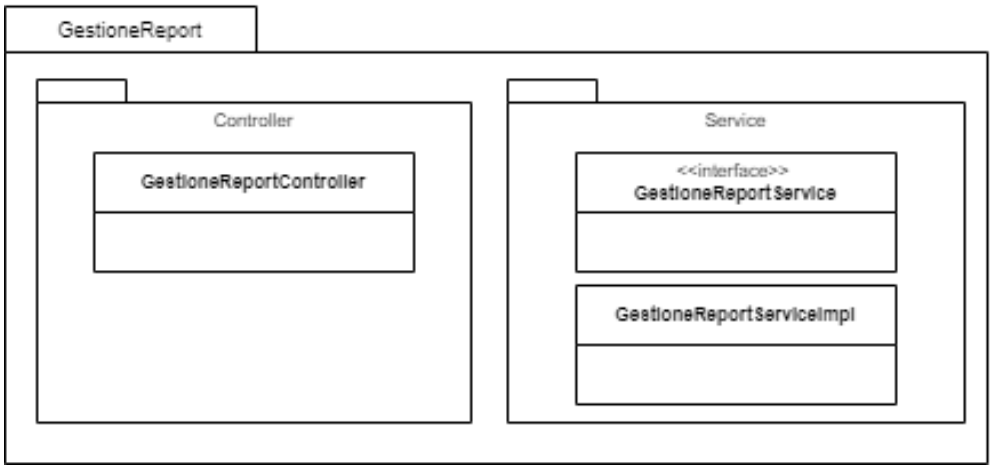
Package Gestione Area Predizioni



Package Gestione ChatBot



Package Gestione Report



3. Class Interfaces

3.1. Package Gestione Device

Nome classe	**GestioneDevice**
Descrizione	Classe che permette di gestire il funzionamento dei sensori
Metodi	+segnalaGuasto (Sensore sensore): Sensore. +registraAnomalia (Parametro parametro). +registraDati (Parametro parametro).
Invariante di classe	/

Nome metodo	**+segnalaGuasto (Sensore sensore)**
Descrizione	Metodo che permette di segnalare un guasto del Sensore
Pre-condizione	/
Post-condizione	**context:** GestioneDevice::segnalaGuasto (sensore) **post: se**nsore.Malfunzionamento() == true;
Nome metodo	**+registraAnomalia (Parametro parametro)**

Descrizione	Metodo che permette registrare una misurazione come anomalia
Pre-condizione	**context:** GestioneDevice:: registraAnomalia (parametro) **pre:** Misurazione.isAnomalia() == true;
Post-condizione	**context:** GestioneDevice:: registraAnomalia (parametro) **post:** Model.saveAnomalia() == true;
Nome metodo	**+registraDati (Parametro parametro).**
Descrizione	Metodo che permette registrare una misurazione
Pre-condizione	**context:** GestioneDevice:: registraDati (parametro) **pre:** Misurazione.isAnomalia() == false;
Post-condizione	**context:** GestioneDevice:: registraDati (parametro) **post:** Model.saveParametro() == true;

Specifica Eccezioni

Nome eccezione	SensorStateNotChanged
Metodo	+segnalaGuasto
Condizione non verificata	Post-condizione
Descrizione	sensore.Malfunzionamento() != true
Nome eccezione	MeasurementIsNotAnomaly
Metodo	+registraAnomalia
Condizione non verificata	Pre-condizione
Descrizione	Misurazione.isAnomalia() != true
Nome eccezione	AnomalyNotSaved
Metodo	+registraAnomalia
Condizione non verificata	Post-condizione

Descrizione	Model.saveAnomalia() != true
Nome eccezione	MeasurementIsAnomaly
Metodo	+registraDati
Condizione non verificata	Pre-condizione
Descrizione	Misurazione.isAnomalia() != false
Nome eccezione	ParameterNotSaved
Metodo	+registraDati
Condizione non verificata	Post-condizione
Descrizione	Model.saveParametro() != true

Package Gestione Area Predizioni

Nome classe	**GestioneAreaPredizioni**
Descrizione	Classe che permette la visualizzazione delle predizioni
Metodi	+visualizzaPredizioniInfarto (DatasetInfarto dataset): double predizione. +visualizzaPredizioniAterosclerosi (DatasetAteroclerosi dataset):: double predizione
Invariante di classe	

Nome metodo	**+visualizzaPredizioniInfarto (DatasetInfarto dataset)**
Descrizione	Metodo che permette di calcolare la predizione di rischio di infarto
Pre-condizione	/
Post-condizione	**context:** GestioneAreaPredizioni:: visualizzaPredizioniInfarto (dataset)

	post: predizione > 0 && predizione < 100;
Nome metodo	**+visualizzaPredizioniAterosclerosi (DatasetAterosclerosi dataset)**
Descrizione	Metodo che permette di calcolare la predizione di rischio di aterosclerosi
Pre-condizione	/
Post-condizione	**context:** GestioneAreaPredizioni: visualizzaPredizioniAterosclerosi (dataset) **post:** predizione > 0 && predizione < 100;

Specifica Eccezioni

Nome eccezione	PredictedValueOutOfRangeForHeartAttack
Metodo	+visualizzaPredizioniInfarto
Condizione non verificata	Post-condizione
Descrizione	predizione <= 0 || predizione >= 100
Nome eccezione	PredictedValueOutOfRangeForAtherosclerosis
Metodo	+visualizzaPredizioniAterosclerosi
Condizione non verificata	Post-condizione
Descrizione	predizione <= 0 || predizione >= 100

Package Gestione ChatBot

Nome classe	**GestioneChatBot**
Descrizione	Classe che permette di gestire le interazioni con il Chatbot
Metodi	+selezionaProblema (String problema): List<String>.
Invariante di classe	/

Nome metodo	**+selezionaProblema (String problema)**
Descrizione	Metodo che permette di selezionare un problema e ottenere le soluzioni
Pre-condizione	/
Post-condizione	**context:** GestioneChatbot::segnalaGuasto (sensore) **post:** listaSoluzioni.size() > 0;

Specifica Eccezioni

Nome eccezione	SolutionsNotFound
Metodo	+selezionaProblema
Condizione non verificata	Post-condizione
Descrizione	listaSoluzioni.size() <= 0

Package Gestione Report

Nome classe	GestioneReport
Descrizione	Classe che permette di gestire la stampa del Report
Metodi	+selezionaIntervallo (DateRange data): List<Misurazione>.
Invariante di classe	/

Nome metodo	**+selezionaIntervallo (LocalDate data)**
Descrizione	Metodo che permette di calcolare la predizione di rischio di infarto
Pre-condizione	/
Post-condizione	**context:** GestioneReport:: selezionaIntervallo (data) **post:** medieMisuazioni.size() > 0

Specifica Eccezioni

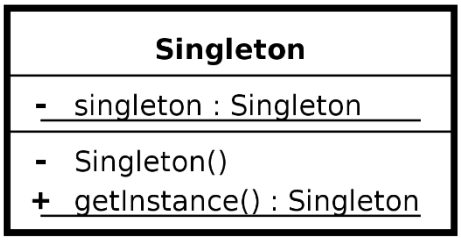
Nome eccezione	WrongAverageMesurements
Metodo	+selezionaIntervallo
Condizione non verificata	Post-condizione
Descrizione	medieMisurazioni.size() <= 0

4. Design Patterns

In questa sezione sono descritti i design pattern utilizzati nello sviluppo di CardioTel.

Singleton

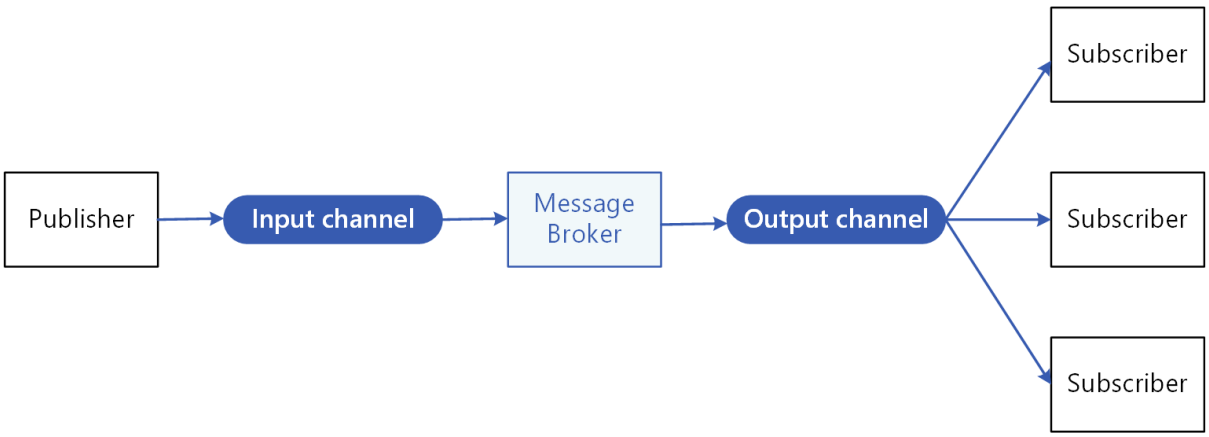
È un design pattern creazionale che limita la creazione di una classe ad una singola istanza a cui è possibile accedere globalmente.
Useremo il singleton per la connessione al DB per permettere al publisher di ottenere una singola istanza di collegamento tramite la quale accedere ai dati.



Publisher - Subscriber

È un design pattern per lo scambio di messaggi: un publisher rende disponibili i messaggi tramite un broker senza saperne il destinatario, dopodiché i subscriber possono accedere ai messaggi a cui sono interessati in modo asincrono.

Nel nostro caso il broker è una classe che ottiene i dati dal database e li rende disponibili ai vari subscriber, i nostri endpoint.



5. Componenti Off-The-Shelf (COTS)

Per lo sviluppo del sistema è previsto l'utilizzo di diversi componenti off-the-shelf. Questi ultimi, sono componenti hardware e software disponibili sul mercato per l'acquisto da parte di aziende di sviluppo interessate ad implementarli nei propri progetti ed utili a risolvere specifici problemi. Le componenti OTS previste per la realizzazione del sistema CardioTel sono le seguenti:

- **Sensori**, dispositivi hardware che fisicamente effettuano la misurazione e la trasformazione della grandezza d'ingresso, nel nostro caso parametri vitali del paziente, in un segnale di natura digitale così da poter essere memorizzato nel sistema. CardioTel impiega dei sensori virtuali per generare valori aleatori utili nella fase di testing;
- **Docker**, software progettato per eseguire processi informatici in ambienti isolati chiamati "container Linux" così da renderli semplici e facilmente distribuibili. La sua implementazione in CardioTel permette di semplificare i processi di deployment del sistema stesso;
- **MongoDB**, è un DBMS non relazionale di tipo NoSQL orientato ai documenti;
- **Mockito**, un tool utilizzato per i mock degli oggetti durante il testing di unità;
- **Quarkus**, un framework Java realizzato per le macchine virtuali Java (JVM) e per la compilazione nativa che ottimizza Java specificamente per i container.

6. Glossario

Sigla/Termine	Definizione
Aterosclerosi	Malattia degenerativa che colpisce le arterie di medio e grosso calibro, infiammandole e irrigidendole a causa del deposito di grassi e globuli bianchi nella loro parete
CardioTel	Nome della web application proposta, composto dalle parole chiavi “Cardiaco” e “Telemonitoraggio”
Chatbot	Programma informatico capace di interagire con l’utente
Device	Insieme di sensori IoT
Frequenza cardiaca	Misura del numero di battiti del cuore in un minuto
Infarto	Morte di un tessuto (cardiaco) che non riceve un adeguato apporto di sangue e ossigeno dalla circolazione arteriosa a lui dedicata
Mockup	Bozza di un oggetto o di un sistema, priva delle funzioni del prodotto finale
Report	Resoconto riepilogativo di una raccolta di dati