Laurea Triennale in Informatica - Università di Salerno
Corso di *Ingegneria del Software* - Prof.ssa F. Ferrucci, Prof. F. Palomba

CHEMO SMART
SCHEDULER FOR CHEMOTERAPY

# T.P.D.
# Test Plan Document
# ChemoSmart

| | |
|---|---|
| Versione | 2.0 |
| Data | 10/02/2023 |
| Destinatario | Prof.ssa F. Ferrucci, Prof. F. Palomba |
| Presentato da | C. Troiano, M. Purice, L. Miranda, A. Nappi, G. Basile, C. De Palma |
| Approvato da | A. Bergamo, F. P. Ianuzziello |

# Revision History

| Data | Versione | Descrizione | Autori |
|---|---|---|---|
| 06/12/2022 | 0.1 | Prima stesura | AB, FPI |
| 06/12/2022 | 0.2 | Inseriti Introduzione, Relazione con altri documenti, Panoramica del Sistema, Funzionalità da non testare, Pass/fail criteria. | AB, FPI |
| 07/12/2022 | 0.3 | Inseriti Approccio, Sospensione e ripristino, Materiale di testing e la sezione riguardante i Test Case. | AB, FPI |
| 09/12/2022 | 0.8 | Inseriti i Test Case divisi per package di gestione. | LM, CT, AN, GB, MP, CDP |
| 09/12/2022 | 0.9 | Revisione del Documento | AB, FPI |
| 12/12/2022 | 1.0 | Revisione del Documento per Consegna Intermedia | AB, FPI |
| 10/02/2023 | 2.0 | Revisione documento con aggiornamento casi di test | LM, CT, AN, GB, MP, CDP |

# Team members

| Nome | Ruolo nel progetto | Acronimo | Informazioni di contatto |
|---|---|---|---|
| Alessandro Bergamo | Project Manager | AB | a.bergamo2@studenti.unisa.it |
| Francesco Pio Ianuzziello | Project Manager | FPI | f.ianuzziello1@studenti.unisa.it |
| Luigi Miranda | Team Member | LM | l.miranda11@studenti.unisa.it |
| Ciro Troiano | Team Member | CT | c.troiano17@studenti.unisa.it |
| Antonio Nappi | Team Member | AN | a.nappi47@studenti.unisa.it |
| Giuseppe Basile | Team Member | GB | g.basile36@studenti.unisa.it |
| Mihail Purice | Team Member | MP | m.purice@studenti.unisa.it |
| Claudio De Palma | Team Member | CDP | c.depalma5@studenti.unisa.it |

# Sommario

# 1 Introduzione

Il sistema che si vuole realizzare ha come obiettivo principale quello di semplificare la schedulazione e l'organizzazione dell'uso dei farmaci chemioterapici e del calendario delle terapie, in modo tale da diminuire i tempi necessari, organizzare al meglio il personale adibito ed evitare lo spreco di farmaci.
In particolare, esso permetterà di ottimizzare e automatizzare il lavoro di schedulazione degli appuntamenti, tenendo conto della disponibilità di farmaci, della patologia di un paziente e dello stadio di quest'ultima, attraverso l'ausilio di una componente di Intelligenza Artificiale.
Il sistema ha, inoltre, il compito di ottimizzare il lavoro degli utenti che si interfacciano con quest'ultimo, semplificando l'iter richiesto, attualmente, per la schedulazione di un appuntamento e la successiva notifica del paziente.
Il sistema, in futuro, cercherà di integrare tutti i sistemi già presenti fondendoli in un unico sistema valido per tutto l'ambiente ospedaliero. Esso punterà a migliorare l'efficienza della gestione del personale, del lavoro e, in particolare, la gestione dei dati, fondamentali in un contesto ospedaliero. Il sistema, inoltre, cercherà di ridurre sensibilmente tutti gli iter burocratici ed amministrativi massimizzando l'efficienza lavorativa del personale addetto. Tutto ciò non sarà presente in una prima release del sistema.

# 2 Relazione con altri documenti

Per una corretta e coerente individuazione dei casi di test si è fatto riferimento ad altri documenti prodotti.

**Relazione con il Requirement Analysis Document (RAD)**

I test case pianificati nel Test Plan sono elaborati in relazione ai requisiti funzionali e non funzionali individuati e presentati nel RAD.

**Relazione con il System Design Document (SDD)**

I test case pianificati nel Test Plan devono rispettare la suddivisione in sottosistemi presentata nell'SDD.

**Relazione con l'Object Design Document (ODD)**

Per ciò che concerne il test di unità e di integrazione, poichè maggiormente legati allo ODD e alla divisione in package del sistema, essi saranno scritti e documentati unicamente all'interno del codice dell'applicativo. Per tale motivo, nel presente documento, non vi saranno riferimenti al loro design.

# 3 Panoramica del sistema

Il sistema sarà realizzato tramite un'architettura a microservizi. Tale architettura ci offre la possibilità di realizzare un sistema efficace ed efficiente con alta scalabilità, requisito fondamentali per l'obiettivo a lungo termine del sistema.

Nello sviluppo del sistema verranno utilizzati **HTML5**, **TAILWIND CSS**, **React.JS** per la realizzazione del front-end. Per il back-end verrà utilizzato **Node.JS** ed **Express.JS**.

Il login alla piattaforma sarà gestito esternamente dal **Sistema Pubblico di Identità Digitale (SPID)** tramite l'API fornito.

Lo scheduler "intelligente" degli appuntamenti verrà realizzato tramite un modulo IA basato su Algoritmi Genetici (GA) tramite linguaggio **Python**, in particolare mediante l'utilizzo delle librerie **Scikit-Learn e Pandas**.

Per la gestione della persistenza, e quindi di un database, verranno utilizzati:

- **Mongoose**, per l'interfacciamento tra database ed i Microservizi;
- **MongoDB** per lo storage di dati.

# 4 Features da non testare

Di seguito la lista delle features di cui si effettuerà il testing per le varie gestioni:

- Gestione Paziente
  - Schedulazione Nuova Terapia
  - Aggiornamento Terapia
- Gestione Appuntamento
  - Schedulazione Appuntamento
  - Cancellazione Appuntamento
  - Rischedulazione Appuntamenti
- Gestione Farmaco
  - Aggiorna quantità Farmaco

Le funzionalità di cui non si andrà ad effettuare le attività di testing riguardano requisiti funzionali di bassa o media priorità; sono inoltre escluse le funzionalità che non prevedono input manuale da parte dell'utente - ad esempio attività riguardanti esclusivamente la visualizzazioni di dati.

Il testing della parte di assegnazione di priorità ai pazienti che comprende una parte riguardante modelli di Machine Learning verrà testata in seguito utilizzando tecniche diverse.

# 5 Pass/fail criteria

Le attività di testing sono mirate ad identificare la presenza di faults (errori) all'interno del sistema, per poi individuare una soluzione.
L'esito di un test case è valutato mediante un oracolo, inteso come il risultato atteso della sua esecuzione, basandosi sui requisiti.
Un test ha successo (pass) se, dato un input al sistema, l'output ottenuto è diverso dall'output atteso dall'oracolo.
Un test fallisce (fail) se, dato un input al sistema, l'output ottenuto è uguale all'output atteso dall'oracolo.
Tutto il testing sarà considerato valido se tutti i seguenti vincoli saranno rispettati:

- Testare tutti i requisiti funzionali ad alta priorità;
- Effettuare test di regressione ogni volta che si introducono nuove caratteristiche al sistema o vengono modificate quelle presenti;
- Raggiungere un branch coverage non inferiore al 75%.

# 6 Approccio

Il testing dell'intero sistema si compone di tre fasi: testing di sistema, testing di integrazione e testing di unità. Verranno progettati nell'ordine appena definito, ma verranno eseguiti in ordine inverso.
Prima della fase di implementazione del sistema avverrà la progettazione dei casi di test di sistema, perfezionati in seguito nella loro fase di esecuzione; durante la fase implementativa avverrà la progettazione dei casi di test di unità.
Durante lo sviluppo saranno eseguite periodiche attività di revisione sul codice prodotto.
Poiché la progettazione è organizzata seguendo un modello simile al modello a V, il testing di sistema è stato pianificato in seguito alla stesura del documento Requirements Analysis Document, mentre la pianificazione del testing di integrazione avverrà dopo la stesura del System Design Document.

## Testing di sistema

Per il testing di sistema sarà utilizzato il tool Selenium IDE, che permette di registrare le azioni che un utente può intraprendere sul browser, in modo da poter implementare ed eseguire i test case di sistema. Il server, per la fase di testing, verrà deployato in localhost.

### *Functional Testing*

Il functional testing ha il fine di validare i requisiti funzionali. Consiste nell'individuare i possibili faults generati dagli input degli utenti.

### *Performance Testing*

A causa del basso budget a disposizione, non si assicura l'esecuzione del performance testing.

### *Pilot Testing*

A causa del basso budget a disposizione, non si assicura l'esecuzione del pilot testing.

### *Acceptance Testing*

L'acceptance testing verrà effettuato solo sul functional testing, ed i Project Manager simuleranno la figura del cliente.

### *Installation Testing*

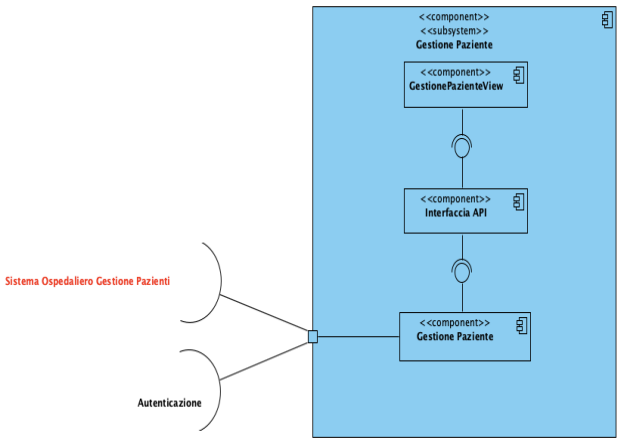
A causa del basso budget a disposizione, non si assicura l'esecuzione dell'installation testing.

## Testing di integrazione

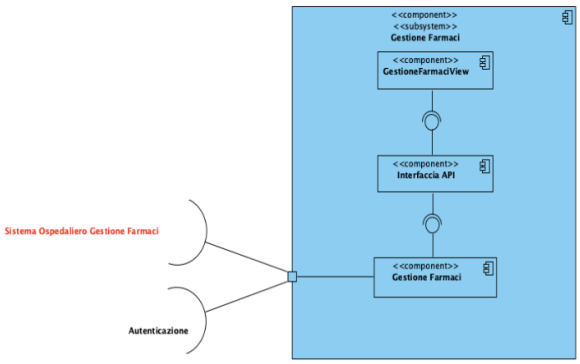
Verrà utilizzato un approccio bottom-up, metodo ritenuto più adatto per un software implementato tramite microservizi. La definizione dei test case avverrà tramite il framework Jest; analogamente con lo stesso framework si avranno a disposizione funzionalità per il mock e funzionalità che permettono di controllare automaticamente il report coverage dei casi di test. Verrà valutato l'utilizzo di Github Actions o Travis CI per realizzare la Continuous Integration.

Il test di integrazione sarà il medesimo per tutte le componenti da testare. Nello specifico, si procederà prima con il test delle classi rappresentanti ogni microservizio come cooperano con il relativo database, e successivamente con il test della classe che funge da interfaccia con il client. Durante questa seconda esecuzione, la chiamata all'interfaccia sarà mockata. Alcuni microservizi ottengono dati da database esterni per cui il testing di tale integrazioe non è mostrata nel diagramma architetturale riportato di seguito; per completezza saranno riportati di seguito i diagrammi che mostrano il collegamento tra il microservizio ed il servizio esterno da cui vengono ottenuti i dati; in particolare sono coinvolti in tale situazione il microservizio relativo alla "Gestione Paziente" ed il microservizio relativo alla "Gestione Farmaci".

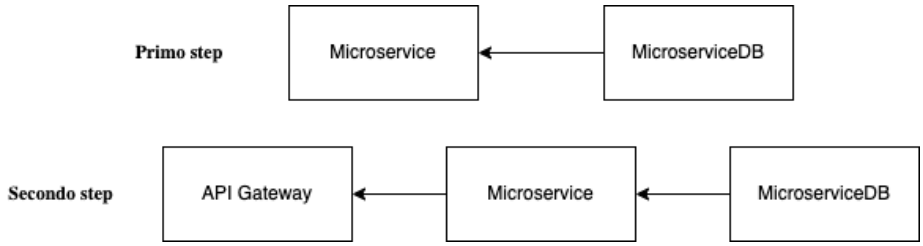
**Gestione Paziente**



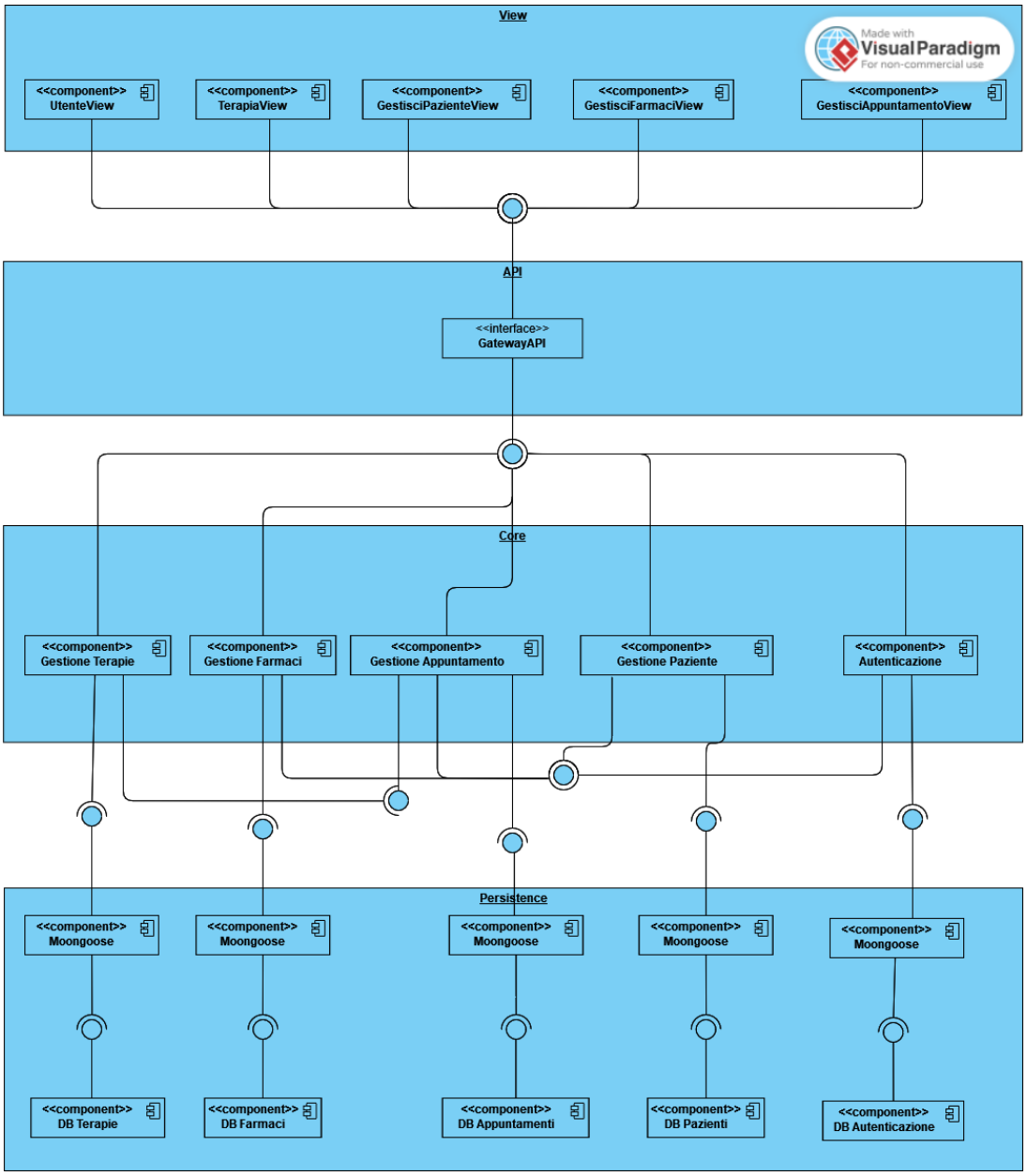

**Gestione Farmaci**




Di seguito viene presentato un esempio grafico di test di integrazione diviso nei due steps.




Per ciò che concerne le dipendenze tra i sottosistemi, si riporta di seguito il diagramma architetturale.




Il test di integrazione verrà fatto a partire dal layer Persistance fino a salire al layer API.

## Testing di unità

Per il testing di unità la strategia prevista consiste nel testare ogni metodo delle classi del sistema. Da esse, sono escluse le interfacce e le classi entity, poiché quest'ultime presentano solo metodi getters e setters. I casi di test saranno definiti attraverso un approccio black-box e saranno documentati direttamente nel codice, attraverso l'uso del framework per il testing di JavaScript Jest.

Per ogni Production Class sarà definita una Test Class che rispetterà il formato NomeProductionClassTest. Tali classi saranno scritte in parallelo alle Production Class, per garantire una più facile copertura del codice. Le stesse classi saranno poi revisionate e modificate da sviluppatori differenti.
Utilizzando Jest come framework per il testing avremo a disposizione tutte le funzionalità che riguardano il mocking, il report coverage ed il test automatic execution, quindi non è necessario utilizzare altri framework che dispongono di funzionalità simili.

### Ispezione del codice

Per aumentare la qualità del codice ci si affiderà principalmente ai controlli del tool Checkstyle e della CI/CD con Jest. Sebbene vi sia l'intenzione di fare ispezione del codice, anche se non costante e approfondita, a causa del basso budget non si può assicurare tale pratica.

# 7 Sospensione e ripristino

In questa sezione verranno specificati i criteri di sospensione del test e le attività di test che dovranno essere ripetute quando si riprende il test.

### Criteri di sospensione

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento dell'esecuzione, un errore nella definizione di uno dei test stessi.

### Criteri di ripristino

Il testing verrà ripreso dopo aver risolto i fault individuati.

# 8 Materiale di testing

L'hardware necessario per l'attività di test è un semplice computer, necessariamente dotato di connessione ad internet, in quanto il sistema dovrà comunque collegarsi a database esterni da cui ottenere dati utili al funzionamento del sistema.

# 9 Test cases

L'approccio per la definizione dei test frame sarà il category partition. Come specificato nei capitoli precedenti, alcuni dati che vengono utilizzati, provengono da sistemi esterni e di conseguenza da database esterni, per tale motivo si andrà a considerare e conseguentemente testare solamente la loro persistenza all'interno dei database a cui si fa riferimento. Al fine di minimizzare il numero di test case, gli input saranno partizionati in classi di equivalenza. Per definire l'output atteso si userà un oracolo umano, per via dell'assenza di specifiche formali/semi-formali.

## 9.1 Gestione Paziente

### 9.1.1 Filtra terapia per paziente

| **Autore: AN** | |
|---|---|
| **Parametro: CF** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [VCF] | 1. Il CF non è valido [ERROR]<br>2. Il CF è valido [PCF_OK] |
| **Parametro: Stato** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PCF] | 1. Il CF non è presente nel db [ERROR]<br>2. Il CF è presente nel db [PN_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|
| TC_1.1_1 | VCF1 | Errore: Codice Fiscale non valido |
| TC_1.1_2 | VCF2, PCF1 | Errore: Codice Fiscale valido ma non presente nel db |
| TC_1.1_3 | VCF2, PCF2 | Corretto. |

### 9.1.2 Aggiornamento Terapia

| **Autore: LM** | |
|---|---|
| **Parametro: CF** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PCF] | 1. Il codice fiscale non è presente nel Database [ERROR]<br>2. Il codice fiscale è presente nel Database [PCF_OK] |
| **Parametro: Farmaco** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PF] | 1. Farmaco non è presente nel Database [ERROR]<br>2. Farmaco è presente nel Database [PF_OK] |
| **Parametro: Numero Appuntamenti** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Correttezza [CNA] | 1. NumeroAppuntamenti <= 0 [ERROR]<br>2. NumeroAppuntamenti >=1 [CNA_OK] |
| **Parametro: Frequenza Appuntamenti** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VF] | 1. Frequenza != 7 && Frequenza != 14 && Frequenza != 21 [ERROR]<br>2. Frequenza == 7 \|\| Frequenza == 14 \|\| Frequenza == 21 [VF_OK] |
| **Parametro: Stato** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Correttezza [CS] | 1. Stato != “In corso” && stato != “Terminata” && stato != “Non schedulata” [ERROR]<br>2. Stato == “In corso” \|\| stato == “Terminata” \|\| stato == “Non schedulata” |

| | |
|---|---|
| | [CS_OK] |
| **Parametro: ID Terapia** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PIDT] | 1. L'ID terapia non è presente nel Database [ERROR]<br>2. L'ID terapia è presente nel Database [PIDT_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|
| TC_1.2_1 | PCF1 | Errore: Il codice fiscale non è presente nel DB. |
| TC_1.2_2 | PCF2, PF1 | Errore: Il farmaco non è presente nel DB. |
| TC_1.2_3 | PCF2, PF2, CNA1 | Errore: Il numero appuntamenti non è valido. |
| TC_1.2_4 | PCF2, PF2, CNA2, VF1 | Errore: la frequenza appuntamenti non è valida |
| TC_1.2_5 | PCF2, PF2, CNA2, VF2, CS1 | Errore: Lo stato inserito non è valido. |
| TC_1.2_6 | PCF2, PF2, CNA2, VF2, CS2, PIDT1 | Errore: La terapia non è presente nel DB. |
| TC_1.2_7 | PCF2, PF2, CNA2, VF2, CS2, PIDT2 | Corretto |

## 9.2 Gestione Appuntamento

### 9.2.1 Schedulazione Appuntamento

| **Autore: CT** | |
|---|---|
| **Parametro: CF** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PCF] | 1. Il CF non è presente nel database [ERROR]<br>2. Il CF è presente nel database [PCF_OK] |

| **Parametro: Nome** | |
|---|---|
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PN] | 1. Il nome non è presente nel database [ERROR]<br>2. Il nome è presente nel database [PN_OK] |
| **Parametro: Cognome** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PC] | 1. Il cognome non è presente nel database [ERROR]<br>2. Il cognome è presente nel database [PC_OK] |
| **Parametro: DataInizio** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VDI] | 1. DataScelta <= DataOdierna [ERROR]<br>2. DataScelta > DataOdierna [VDI_OK] |
| **Parametro: DataFine** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VDF] | 1. DataScelta <= DataOdierna [ERROR]<br>2. DataScelta > DataOdierna [VDF_OK] |
| **Parametri: DataInizio, DataFine** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Coerenza [CO] | 1. DataFine.ora < DataInizio.ora [ERROR]<br>2. DataFine.ora > DataInizio.ora [CO_OK] |
| **Parametro: FarmaciAppuntamento** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Match [MFA] | 1. I farmaci dell'appuntamento non rispettano quelli previsti nella terapia. [ERROR]<br>2. I farmaci dell'appuntamento rispettano quelli previsti nella terapia. [MFA_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|
| TC_2.1_1 | PCF1 | Errore: Il CF non è presente nel database |
| TC_2.1_2 | PCF2, PN1 | Errore: Il nome non è presente nel database |
| TC_2.1_3 | PCF2, PN2, PC1 | Errore: Il cognome non è presente nel database |
| TC_2.1_4 | PCF2, PN2, PC2, VDI1 | Errore: La dataInizio scelta non è valida |
| TC_2.1_5 | PCF2, PN2, PC2, VDI2, VDF1 | Errore: La dataFine scelta non è valida |
| TC_2.1_6 | PCF2, PN2, PC2, VDI2, VDF2, CO1 | Errore: Le due date scelte non sono coerenti fra di loro |
| TC_2.1_7 | PCF2, PN2, PC2, VDI2, VDF2, CO2, MFA1 | Errore: il farmaco non è presente nel database |
| TC_2.1_8 | PCF2, PN2, PC2, VDI2, VDF2, CO2, MFA2 | Corretto |

### 9.2.2 Cancellazione Appuntamento

| **Autore: GB** | |
|---|---|
| **Parametro: IDAppuntamento** | |
| Persistenza [PIDA] | 1. L'appuntamento non è presente nel database [ERROR]<br>2. L'appuntamento è presente nel database [PIDA_OK] |
| **Parametro: CF** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PCF] | 1. Il CF non è presente nel database [ERROR]<br>2. Il CF è presente nel database [PCF_OK] |
| **Parametro: Nome** | |
| **Nome Categoria** | **Scelta per la categoria** |

| | |
|---|---|
| Persistenza [PN] | 1. Il nome non è presente nel database [ERROR]<br>2. Il nome è presente nel database [PN_OK] |
| **Parametro: Cognome** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Persistenza [PC] | 1. Il cognome non è presente nel database [ERROR]<br>2. Il cognome è presente nel database [PC_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|
| TC_2.2_1 | PIDA1 | Errore: Appuntamento non presente nel DB |
| TC_2.2_2 | PIDA2, PCF1 | Errore: Codice Fiscale non presente nel DB |
| TC_2.2_3 | PIDA2, PCF2, PN1 | Errore: Nome non presente nel DB |
| TC_2.2_4 | PIDA2, PCF2, PN2, PC1 | Errore: Cognome non presente nel DB |
| TC_2.2_5 | PIDA2, PCF2, PN2, PC2 | Corretto |

### 9.2.3 Rischedulazione Appuntamento

| | |
|---|---|
| **Autore: MP** | |
| **Parametro: idAppuntamento** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PCI] | 1. idAppuntamento non è presente nel database [ERROR]<br>2. idAppuntamento è presente nel database [PCI_OK] |
| **Parametro: CF** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PCF] | 1. Il CF non è presente nel database [ERROR]<br>2. Il CF è presente nel database [PCF_OK] |

| **Parametro: Nome** | |
|---|---|
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PN] | 1. Il nome non è presente nel database [ERROR]<br>2. Il nome è presente nel database [PN_OK] |
| **Parametro: Cognome** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PC] | 1. Il cognome non è presente nel database [ERROR]<br>2. Il cognome è presente nel database [PC_OK] |
| **Parametro: DataInizio** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VDI] | 1. DataScelta <= DataOdierna [ERROR]<br>2. DataScelta > DataOdierna [VDI_OK] |
| **Parametro: DataFine** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VDF] | 1. DataScelta <= DataOdierna [ERROR]<br>2. DataScelta > DataOdierna [VDF_OK] |
| **Parametro: DataInizio, DataFine** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Coerenza [CO] | 1. DataFine.ora < DataInizio.ora [ERROR]<br>2. DataFine.ora > DataInizio.ora [CO_OK] |
| **Parametro: FarmaciAppuntamento** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Match [MFA] | 1. I farmaci dell'appuntamento non rispettano quelli previsti nella terapia. [ERORR]<br>2. I farmaci dell'appuntamento rispettano quelli previsti nella terapia. [MFA_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|
| TC_2.3_1 | PCI1 | Errato: Id non è presente nel database |
| TC_2.3_2 | PCI2, PCF1 | Errore: Il CF non è presente nel database |
| TC_2.3_3 | PCI2, PCF2, PN1 | Errore: Il Nome non è presente nel database |
| TC_2.3_4 | PCI2, PCF2, PN2, PC2 | Errore: Il Cognome non è presente nel database |
| TC_2.3_5 | PCI2, PCF2, PN2, PC2, VDI1 | Errato: La dataInizio scelta non è valida |
| TC_2.3_6 | PCI2, PCF2, PN2, PC2, VDI2, VDF1 | Errato: La dataFine scelta non è valida |
| TC_2.3_7 | PCI2, PCF2, PN2, PC2, VDI2, VDF2, CO1 | Errore: Le due date scelte non sono coerenti fra di loro |
| TC_2.3_8 | PCI2, PCF2, PN2, PC2, VDI2, VDF2, CO2, MFA1 | Errore: il farmaco non è presente nel database |
| TC_2.3_9 | PCI2, PCF2, PN2, PC2, VDI2, VDF2, CO2, MFA2 | Corretto |

## 9.3 Gestione Farmaco

### 9.3.1 Aggiorna quantità farmaco

| **Autore: CDP** | |
|---|---|
| **Parametro: Farmaco** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Presenza [PF] | 1. Farmaco non è presente nel Database [ERROR]<br>2. Farmaco è presente nel Database [PF_OK] |
| **Parametro: Stock** | |
| **Nome Categoria** | **Scelta per la categoria** |
| Validità [VS] | 1. Stock < 0 [ERROR]<br>2. Stock >= 0 [CF_OK] |

| Test Case ID | Test Frame | Esito |
|---|---|---|

| | | |
|---|---|---|
| TC_3.1_1 | PF1 | Errore: Il farmaco non è presente nel Database |
| TC_3.1_2 | PF2, VS1 | Errore: Il valore dello stock non è valido |
| TC_3.1_3 | PF2, VS2 | Corretto |

# 10 Testing schedule

Le attività di pianificazione del testing avverranno come definito nei capitoli precedenti, cioè subito dopo la fase di design necessaria per la pianificazione.

La scrittura dei casi di test avverrà in contemporanea con lo sviluppo del codice.

L'esecuzione dei test avverrà sia durante che dopo l'implementazione del sistema. Una volta concluso lo sviluppo, tutti i test saranno rieseguiti per garantirne il corretto funzionamento e produrre i report finali.