

β -Carotene



Membri del team:

Andrea Eugenio Cesaretti 0000937369

Mirko Legnini 0000923730

Luca De Dominicis 0000925531

Abstract

Si vuole sviluppare un software che permetta al nostro cliente di soddisfare le sue esigenze da nutrizionista. L'applicazione deve permettere di compilare e salvare in memoria diverse diete, tenendo traccia dei valori nutrizionali di ciascuna di esse. Il cliente deve inoltre essere in grado di mantenere una banca dati dei clienti, controllando l'evoluzione di parametri fisiologici (altezza, peso, pliche...) e lo storico delle diete. Le diete devono essere compilabili in maniera dinamica quindi il numero di pasti al giorno deve poter essere scelto, così come gli alimenti che compongono ciascun pasto; i macronutrienti devono essere monitorati giorno per giorno mentre i micronutrienti devono essere controllati su periodi più lunghi. L'applicazione deve disporre anche di un database degli alimenti per permettere il calcolo automatico delle kcal e dei nutrienti specificata la quantità di alimento desiderata. Si sono inoltre discusse col cliente due possibili estensioni:

- la prima prevede la possibilità di calcolare in automatico alcune combinazioni di alimenti per soddisfare un certo fabbisogno giornaliero;
- la seconda prevede lo sviluppo di un applicativo che permetta ai pazienti di caricare personalmente giorno per giorno alcuni parametri fisiologici (es. il peso) per tracciare l'andamento della dieta.

Analisi dei requisiti

Raccolta dei requisiti

- Il software deve permettere l'autenticazione del cliente
- Il nutrizionista deve disporre di una agenda settimanale in cui segnare gli appuntamenti
- Deve essere possibile visualizzare l'elenco dei pazienti registrati
- Deve essere possibile visualizzare la base dati degli alimenti
- Per ogni paziente deve essere possibile:
 - Inserire i dati di contatto (Nome, cognome, data di nascita, telefono, e-mail)
 - Inserire dati clinici e antropometrici
 - Inserire una nuova dieta
 - Visualizzare lo storico delle diete associate
- Nella creazione della dieta il nutrizionista deve poter inserire:
 - Le soglie in grammi di macronutrienti da assumere giornalmente
 - Un numero di pasti variabile
- Per ogni pasto l'applicazione deve permettere di:
 - Inserire il nome del pasto
 - Inserire un numero non definito di alimenti specificandone la quantità di ognuno
- L'applicazione controlla ad ogni alimento inserito se i macronutrienti sono oltre la soglia e in tal caso notifica la situazione
- L'applicazione tiene traccia dei micronutrienti settimanalmente
- L'applicazione deve poter permettere di esportare la dieta
- L'applicazione deve poter essere usata su Windows 10
- L'applicazione deve girare in locale, senza utilizzo di server

Requisiti del sistema

Id Requisito	Descrizione requisito	Tipo
R1F	L'applicazione deve disporre di un calendario per ogni paziente da riempire con un numero di pasti variabile per giorno.	Funzionale
R2F	I pasti possono essere popolati manualmente con diversi alimenti.	Funzionale
R3F	L'applicazione calcola in automatico i valori nutrizionali dati alimenti e quantità.	Funzionale
R4F	L'applicazione tiene traccia giornalmente delle kcal e dei macronutrienti assunti dal paziente	Funzionale
R5F	L'applicazione tiene traccia settimanalmente dei micronutrienti assunti dal paziente	Funzionale
R6F	Deve essere possibile tenere traccia dell'evoluzione dei parametri fisiologici del paziente	Funzionale
R7F	Deve essere possibile creare da zero la scheda di un nuovo paziente inserendo manualmente i dati.	Funzionale
R8F	Le diete create devono poter essere associate ai pazienti per uno specifico periodo.	Funzionale
R9F	Deve essere possibile inserire gli alimenti specificando l'apporto in macronutrienti.	Funzionale
R10F	Il cliente deve avere a disposizione un'agenda su cui segnare gli appuntamenti coi pazienti	Funzionale
R11F	L'applicazione deve disporre di un servizio di stampa per diete	Funzionale

R12F	L'applicazione deve presentare una schermata di login per il cliente	Funzionale
R13F	Le diete devono poter essere modificate	Funzionale
R14F	Deve essere possibile mantenere uno storico delle diete	Funzionale
R15F	Deve essere possibile ampliare il catalogo degli alimenti con alimenti personalizzati.	Funzionale
R16F	L'applicazione deve segnalare visivamente al cliente se alcune soglie sono state superate.	Funzionale
R17F	Deve essere possibile accedere direttamente agli storici dei pazienti.	Funzionale
R18F	Nella scheda del cliente si devono poter inserire anche dati clinici	Funzionale
R19F	Il cliente deve poter specificare degli intervalli raccomandati di valori nutrizionali per le diete.	Funzionale
R23F	Deve essere possibile registrare e aggiornare dati anagrafici e di contatto dei pazienti	Funzionale
R1NF	L'applicazione deve essere facilmente navigabile e chiara	Non funzionale
R2NF	L'inserimento di dati deve essere veloce	Non funzionale
R3NF	L'esportazione di dati deve essere veloce	Non funzionale
R4NF	La dieta di un paziente non deve essere associata ad un altro paziente	Non funzionale
R5NF	I parametri fisiologici di un paziente non devono essere associati ad altri pazienti	Non funzionale
R6NF	L'applicazione deve gestire eventuali omonimie fra pazienti	Non funzionale
R7NF	L'applicazione deve poter essere eseguita su Windows 10.	Non funzionale

R8NF	Le diete correnti per ogni cliente non devono essere più di una	Non funzionale
------	---	----------------

Analisi del dominio

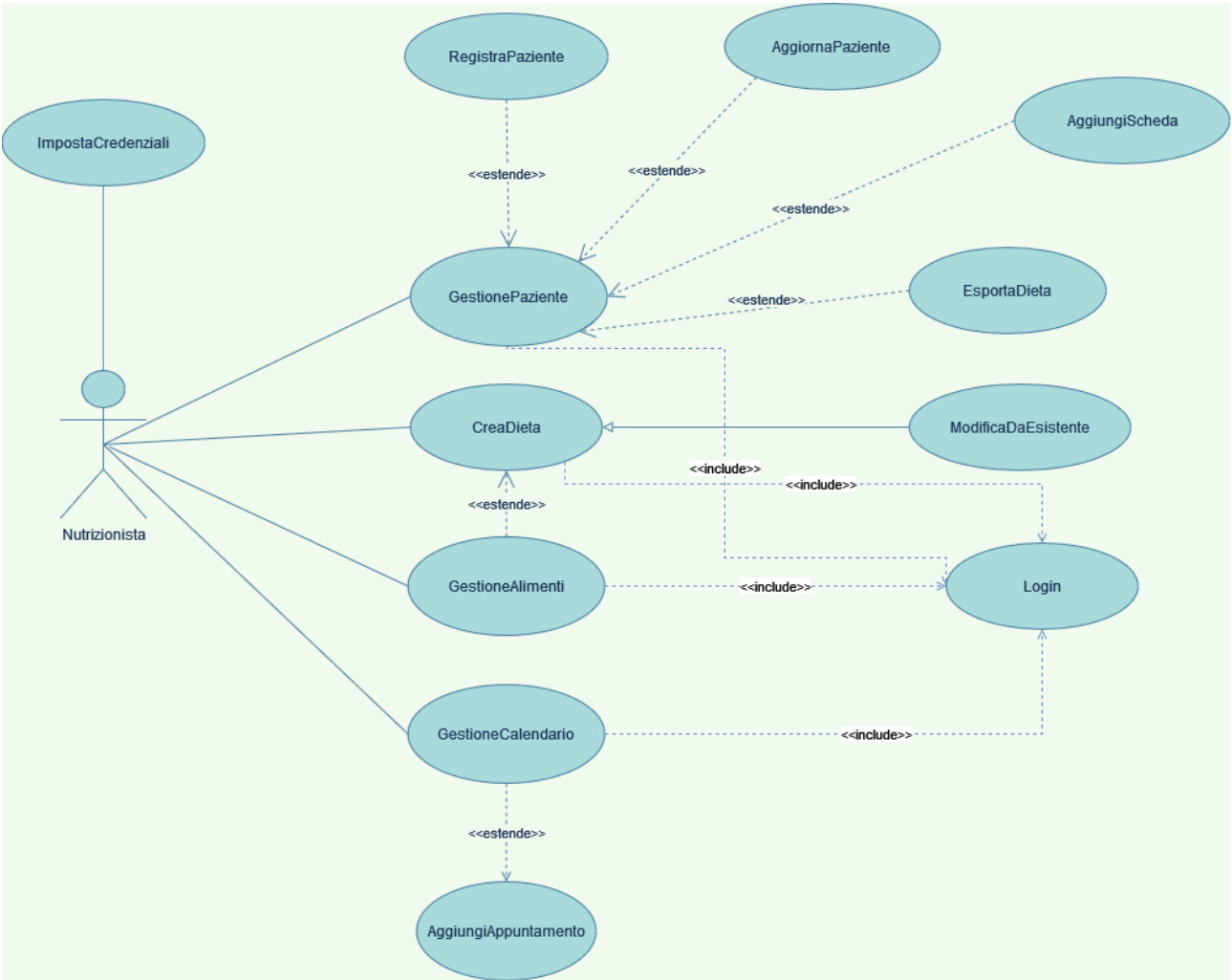
Glossario

Voce	Definizione	Sinonimi
Cliente	Il nutrizionista che sfrutta l'applicazione	Nutrizionista, medico, professionista.
Dati Clinici	Informazioni sulla salute del paziente ricavate da esami clinici (analisi del sangue, urine, ...)	
Paziente	Cliente del nutrizionista	
Scheda	Visualizzazione istantanea di un paziente, che include i parametri corporei e la dieta corrente	
Storico	Insieme dei dati presenti e passati di un paziente	
Calendario	Visualizzazione di un insieme di giorni	
Pasto	Momento della giornata in cui il paziente si nutre	
Alimenti	Sostanze nutritive che compongono un pasto	
Valori Nutrizionali	Valori che denotano la composizione dell'alimento di interesse al nutrizionista	
Macronutrienti	Macromolecole che compongono gli alimenti: proteine, glucidi, lipidi	Macro
Micronutrienti	Molecole degli alimenti che non sono macronutrienti	Micro
Kcal	Unità di misura dell'apporto energetico di un alimento	
Misure Antropometriche	Parametri che descrivono lo stato fisico del paziente (Altezza, peso, età, circonferenze)	Parametri Corporei
Peso	Massa del paziente	
Circonferenza vita	Misura in centimetri della vita del paziente	
Circonferenza fianchi	Misura in centimetri dei fianchi del paziente	

Plica tricipitale	Misura della plica tricipitale presa col plicometro.	
Plica sottoscapolare	Misura della plica sotto la scapola presa col plicometro	
Plica sovrailiaca	Misura presa col plicometro	
Plica addominale	Misura della plica dell'addome presa col plicometro	
Plica bicipitale	Misura della plica del bicipite presa col plicometro	
Plica quadricipitale	Misura della plica del quadricipite presa col plicometro	
Agenda	Agenda del cliente che dispone di un calendario su cui annotare gli appuntamenti	
Appuntamento	Visita ad un paziente	
Dieta	Prescrizione del medico al paziente formata da un calendario con i pasti	

Analisi dei Requisiti

Casi d'uso



Scenari

Titolo	Login
Descrizione	Autenticazione del nutrizionista
Attori	Nutrizionista
Relazioni	GestionePaziente, GestioneCalendario, (GestioneBaseDati)
Precondizioni	L'attore deve essere registrato
Postcondizioni	L'attore è autenticato nel sistema
Scenario principale	<ol style="list-style-type: none">1. Al nutrizionista viene richiesto di inserire username e password2. Il nutrizionista inserisce i dati richiesti3. Il nutrizionista invia i dati al sistema4. Il sistema verifica la correttezza dei dati
1. Scenari alternativi	Scenario A: Le credenziali non sono valide <ol style="list-style-type: none">1. L'errore viene notificato all'attore2. Il sistema ripresenta la schermata di autenticazione all'attore.
Requisiti non funzionali	
Punti aperti	

Titolo	GestionePaziente
Descrizione	Il medico gestisce la scheda del paziente
Attori	Nutrizionista
Relazioni	GestioneApplicazione, AggiornaPaziente, RegistraPaziente, EsportaDieta
Precondizioni	
Postcondizioni	
Scenario principale	<ol style="list-style-type: none">1. Login.2. Il Sistema permette al nutrizionista di selezionare il paziente desiderato3. Se il paziente non esiste, il sistema offre la possibilità di registrarlo tramite RegistraPaziente.4. Se è necessario inserire nuovi dati relativi al paziente il nutrizionista può farlo tramite AggiornaPaziente5. Se necessario, il nutrizionista può esportare su un documento esterno la dieta tramite EsportaDieta.

Scenari alternativi	
Requisiti non funzionali	R1NF, <u>R6NF</u>
Punti aperti	

Titolo	RegistraPaziente
Descrizione	Un paziente nuovo viene registrato all'interno del sistema
Attori	Nutrizionista
Relazioni	GestionePaziente
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Il Sistema presenta all'utente una schermata per inserire i dati anagrafici e i contatti del paziente 2. L'utente inserisce i dati 3. L'utente invia i dati al sistema 4. Il sistema crea un identificativo univoco e registra il paziente
Scenari alternativi	
Requisiti non funzionali	R2NF, R6NF
Punti aperti	

Titolo	AggiornaPaziente
Descrizione	Il medico può aggiornare i dati di un paziente
Attori	Nutrizionista
Relazioni	GestionePaziente
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Al nutrizionista vengono presentati gli attuali dati del paziente 2. Il nutrizionista aggiunge i dati aggiornati.
Scenari alternativi	
Requisiti non funzionali	R2NF, R5NF
Punti aperti	

Titolo	AggiungiScheda
Descrizione	Il medico può registratr i parametri fisiologici e I dati clinici del paziente
Attori	Nutrizionista
Relazioni	GestionePaziente
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Al nutrizionista viene presentata una scheda da riempire con I dati del paziente. 2. Il nutrizionista riempie la scheda con i parametri ottenuti dalle misurazioni 3. Il nutrizionista allega eventuali dati clinici ottenuti dal paziente. 4. Il nutrizionista salva la scheda.
Scenari alternativi	
Requisiti non funzionali	R2NF, R5NF

Titolo	GestioneAlimenti
Descrizione	Il nutrizionista può modificare la base dati di alimenti dell'applicazione
Attori	Nutrizionista
Caso d'uso	Questo
Relazioni	GestioneApplicazione, CreaDieta
Precondizioni	
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Login 2. Il Sistema propone all'utente una visualizzazione della base dati di alimenti 3. L'utente può aggiungere, rimuovere o modificare un alimento 4. Quando ha terminato, ritorno alla schermata precedente
Scenari alternativi	
Requisiti non funzionali	R2NF
Punti aperti	

Titolo	CreaDieta
Descrizione	Il cliente può creare una dieta che viene associata al paziente
Attori	Nutrizionista
Relazioni	ModificaDaEsistente, GestioneAlimenti
Precondizioni	L'utente deve essere già autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Il Sistema permette all'utente di specificare gli intervalli di valori nutrizionali di riferimento per la dieta 2. Il nutrizionista inserisce i valori 3. Il sistema permette di comporre la dieta visualizzando un calendario settimanale 4. Il nutrizionista può aggiungere pasti in ogni giornata. Per ogni pasto seleziona gli alimenti e le quantità. 5. Se un alimento non è presente nella banca dati può aggiungerlo via GestioneAlimenti. 6. Il sistema calcola automaticamente ad ogni inserimento i valori nutrizionali totali. 7. Il sistema notifica visivamente all'utente se i valori di riferimento vengono superati 8. La dieta viene finalizzata e associata al paziente come dieta corrente
Scenari alternativi	
Requisiti non funzionali	R2NF, R4NF, R8NF
Punti aperti	

Titolo	ModificaDaEsistente
Descrizione	Permette di creare una dieta modificandone una preesistente
Attori	Nutrizionista
Relazioni	CreaDieta
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Il Sistema mostra una scelta di diete compilate 2. L'utente seleziona la dieta di interesse 3. Il sistema la carica 4. CreaDieta
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	EsportaDieta
Descrizione	L'utente può richiedere al sistema l'esportazione su documento della dieta corrente di un paziente
Attori	Nutrizionista
Relazioni	GestionePaziente
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Il cliente richiede al sistema l'esportazione di una dieta 2. Il sistema produce automaticamente un documento dettagliato della dieta più recente associata al paziente 3. Il cliente può scegliere di salvare con nome il documento
Scenari alternativi	
Requisiti non funzionali	R3NF
Punti aperti	

Titolo	GestioneCalendario
Descrizione	L'utente visualizza ed eventualmente modifica la sua agenda personale
Attori	Nutrizionista
Caso d'uso	
Relazioni	GestioneApplicazione
Precondizioni	
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. Login 2. Il Sistema propone all'utente un calendario con la visualizzazione degli appuntamenti 3. L'utente può visualizzare il dettaglio di un appuntamento 4. Se necessario, l'utente può aggiungere un nuovo appuntamento tramite AggiungiAppuntamento 5. Ritorno alla schermata precedente
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	ImpostaCredenziali
Descrizione	Il nutrizionista può modificare le proprie credenziali di accesso
Attori	Nutrizionista
Caso d'uso	ImpostaCredenziali
Relazioni	
Precondizioni	Esistono delle credenziali
Postcondizioni	Esistono altre credenziali
Scenario principale	<ol style="list-style-type: none"> 1. L'utente inserisce le credenziali correnti 2. Il Sistema mostra all'utente una schermata dove scegliere una coppia di nuove credenziali 3. Il Sistema le memorizza al posto delle credenziali correnti
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Titolo	AggiungiAppuntamento
Descrizione	L'utente aggiunge un appuntamento con un paziente nel futuro
Attori	Nutrizionista
Relazioni	GestioneCalendario
Precondizioni	L'utente deve essere autenticato
Postcondizioni	
Scenario principale	<ol style="list-style-type: none"> 1. L'utente seleziona data e ora per l'appuntamento. 2. L'utente inserisce il dettaglio dell'appuntamento. 3. L'operazione termina con successo e l'appuntamento viene registrato. 4. Ritorno alla schermata precedente.
Scenari alternativi	<p>Data nel passato:</p> <ol style="list-style-type: none"> a) Il Sistema notifica all'utente che la data precede la data corrente b) Il Sistema propone all'utente di rilelezionare una data <p>Appuntamenti sovrapposti:</p> <ol style="list-style-type: none"> a) Il Sistema notifica all'utente che la data e l'ora selezionate non sono disponibili. b) Il Sistema propone all'utente di rilelezionare una data o un orario per l'appuntamento.
Requisiti non funzionali	
Punti aperti	

Analisi del Rischio

Tabella valutazione dei beni

Bene	Valore	Esposizione
Dati personali dei pazienti	Alto, sono dati riservati e in alcuni casi anche cartelle cliniche.	Bassa, l'applicazione ha un singolo punto di accesso.
Base dati degli alimenti	Medio/Basso, la base dati è complessivamente replicabile, ad eccezione degli inserimenti personalizzati che però sono in quantità ridotta.	Bassa, l'applicazione ha un singolo punto di accesso.
Sistema	Medio, il sistema in sé è utile al cliente ma in caso di malfunzionamenti non è un problema ripristinarlo.	Bassa, è vulnerabile solo a danneggiamenti fisici.

Tabella minacce e controlli

Minaccia	Probabilità	Controllo	Fattibilità
Furto dei dati di accesso	Bassa (il sistema è accessibile solo in locale)	Log degli accessi all'applicazione.	Basso costo
Furto dei dati dei pazienti	Bassa (il sistema è accessibile solo in locale)	Cifratura dei dati personali	Costo medio
Danni al supporto fisico del sistema	Media	Backup dell'applicazione	Costo medio

Analisi della tecnologia dal punto di vista della sicurezza

Tecnologia	Vulnerabilità
Sistema di autenticazione	<ul style="list-style-type: none">• Password banale• L'utente rivela la password con un attacco di ingegneria sociale• Password cracking• Password fissa
Supporto fisico	<ul style="list-style-type: none">• Il supporto fisico può essere soggetto a malfunzionamenti e potrebbe subire danni irreversibili
Cifratura	<ul style="list-style-type: none">• Cifratura Simmetrica:<ul style="list-style-type: none">• Tempo di vita della chiave• Più informazioni vengono cifrate con la stessa chiave, più materiale è offerto per l'analisi del testo a un attaccante• Memorizzazione della chiave• Lunghezza della chiave
Backup	<ul style="list-style-type: none">• Supporto del backup può essere in prossimità del supporto dell'applicazione• Supporto del backup può essere rubato

Security use case e misuse case

Titolo	ControlloAccesso	
Descrizione	Il sistema controlla gli accessi per identificare le attività di malintenzionati	
Misuse case	FurtoCredenziali	
Relazioni		
Precondizioni	Il Sistema ha memorizzato dati sensibili	
Postcondizioni	Il Sistema registra le attività interne per futura consultazione	
Scenario principale	Sistema	Attaccante
		L'attaccante naviga all'interno del sistema e raccoglie informazioni
	Il sistema registra le attività dell'attaccante	
Scenario di attacco avvenuto con successo	Sistema	Attaccante
		L'attaccante naviga all'interno del sistema e raccoglie informazioni
	Il sistema registra le attività dell'attaccante	
		L'attaccante riesce a rubare i dati sensibili e ad andarsene impunito

Titolo	ProtezioneDati	
Descrizione	Il sistema impedisce che i dati dei pazienti siano accessi al di fuori dello stesso.	
Misuse case	FurtoDati	
Relazioni		
Precondizioni	Il Sistema ha memorizzato dati sensibili	
Postcondizioni		
Scenario principale	Sistema	Attaccante
		L'attaccante ha accesso alla macchina e cerca di scaricare i dati dell'applicazione dall'esterno
	Il sistema ha preventivamente cifrato i dati in modo da renderli inaccessibili	
Scenario di attacco avvenuto con successo	Sistema	Attaccante
		L'attaccante ha accesso alla macchina e cerca di scaricare i dati dell'applicazione dall'esterno
	Il sistema ha preventivamente cifrato i dati in modo da renderli inaccessibili	
		L'attaccante riesce ad aggirare la cifratura e ottiene accesso ai dati.

Diagramma casi d'uso con Sicurezza

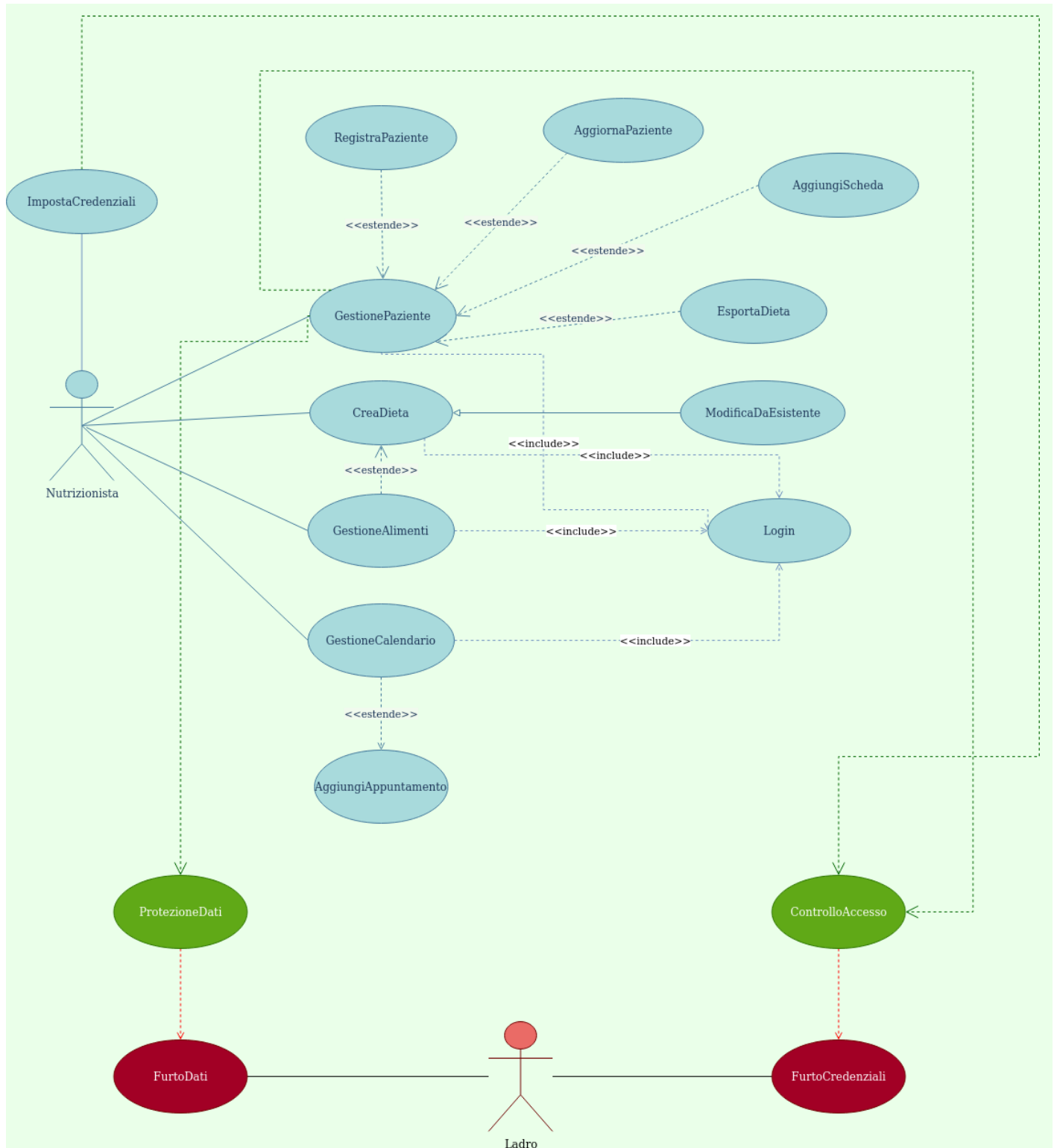


Tabella dei requisiti di sicurezza

Id Requisito	Descrizione requisito	Tipo
R1F	L'applicazione deve disporre di un calendario per ogni paziente da riempire con un numero di pasti variabile per giorno.	Funzionale
R2F	I pasti possono essere popolati manualmente con diversi alimenti.	Funzionale
R3F	L'applicazione calcola in automatico i valori nutrizionali dati alimenti e quantità.	Funzionale
R4F	L'applicazione tiene traccia giornalmente delle kcal e dei macronutrienti assunti dal paziente	Funzionale
R5F	L'applicazione tiene traccia settimanalmente dei micronutrienti assunti dal paziente	Funzionale
R6F	Deve essere possibile tenere traccia dell'evoluzione dei parametri fisiologici del paziente	Funzionale
R7F	Deve essere possibile registrare un nuovo paziente inserendo manualmente i dati.	Funzionale
R8F	Le diete create devono poter essere associate ai pazienti per uno specifico periodo.	Funzionale
R9F	Deve essere possibile inserire gli alimenti specificando l'apporto in macronutrienti.	Funzionale
R10F	Il cliente deve avere a disposizione un'agenda su cui segnare gli appuntamenti coi pazienti	Funzionale
R11F	L'applicazione deve disporre di un servizio di stampa per diete	Funzionale

R12F	L'applicazione deve presentare una schermata di login per il cliente	Funzionale
R13F	Le diete devono poter essere modificate	Funzionale
R14F	Deve essere possibile mantenere uno storico delle diete	Funzionale
R15F	Deve essere possibile ampliare il catalogo degli alimenti con alimenti personalizzati.	Funzionale
R16F	L'applicazione deve segnalare visivamente al cliente se alcune soglie sono state superate.	Funzionale
R17F	Deve essere possibile accedere direttamente agli storici dei pazienti.	Funzionale
R18F	Nella scheda del cliente si devono poter inserire anche dati clinici	Funzionale
R19F	Il cliente deve poter specificare degli intervalli raccomandati di valori nutrizionali per le diete.	Funzionale
R20FS	Il sistema deve registrare dei log delle attività legate ai pazienti	Funzionale Sicurezza
R21FS	Il sistema deve eseguire dei backup a cadenza periodica	Funzionale Sicurezza
R22FS	Il sistema deve mostrare data e ora dell'ultimo accesso nella schermata iniziale	Funzionale Sicurezza
R23F	Deve essere possibile registrare e aggiornare dati anagrafici e di contatto dei pazienti	Funzionale
R1NF	L'applicazione deve essere facilmente navigabile e chiara	Non funzionale
R2NF	L'inserimento di dati deve essere veloce	Non funzionale
R3NF	L'esportazione di dati deve essere veloce	Non funzionale
R4NF	La dieta di un paziente non deve essere associata ad un altro paziente	Non funzionale

R5NF	I parametri fisiologici di un paziente non devono essere associati ad altri pazienti	Non funzionale
R6NF	L'applicazione deve gestire eventuali omonimie fra pazienti	Non funzionale
R7NF	L'applicazione deve poter essere eseguita su Windows 10.	Non funzionale
R8NF	Le diete correnti per ogni cliente non devono essere più di una	Non funzionale
R9NFS	I dati dei pazienti devono essere cifrati a riposo	Non funzionale sicurezza
R10NFS	La password deve essere di almeno 8 caratteri e deve cambiare una volta ogni due mesi.	Non funzionale sicurezza

Complessivamente il sistema non risulta molto vulnerabile ad attacchi dall'esterno essendo un sistema concentrato, ma per la medesima ragione è esposto ad un alto rischio di malfunzionamento della macchina ospitante. Abbiamo pertanto dedotto che:

1. È opportuno che il sistema mantenga dei log delle attività legate a dati personali per poter risalire ad eventuali accessi da parte di estranei
2. Al fine di evitare che siano visibili al di fuori dell'applicazione i dati dei pazienti devono essere cifrati.
3. Per tutelare il sistema da eventuali malfunzionamenti della macchina ospitante riteniamo necessario mantenere un backup su un'altra risorsa.

Analisi del problema

Analisi del documento dei requisiti: Analisi della funzionalità

I colori sono significativi: i campi con sfondi più scuri rappresentano funzionalità semplici o complesse che si possono ritrovare identiche nel diagramma dei casi d'uso, in caso di funzionalità complesse ci sono di seguito dei campi di colori più chiari che vanno a specificare quali sono le sotto funzionalità che le compongono

Sì Marco lo abbiamo copiato, è così che funziona l'informatica.

Tabella funzionalità

Funzionalità	Id	Tipo	Grado complessità	Requisiti collegati
GestionePaziente	F1	Memorizzazione dati, gestione dati, Interazione con l'esterno	Complessa	[R1F-R9F], R11F, R13F, R14F, [R16F-R19]
RegistraPaziente	F1.1	Memorizzazione dati, gestione dati, interazione con l'esterno	Semplice	R7F
VisualizzaPaziente	F1.2	Interazione con l'esterno	Semplice	R6F, R14F, R17F
EsportaDieta	F1.3	Interazione con l'esterno, gestione dati	Semplice	R11F
AggiornaPaziente	F1.4	Memorizzazione dati, interazione con l'esterno, gestione dati	Semplice	R23F
AggiungiScheda	F1.5	Memorizzazione dati, interazione con l'esterno, gestione dati	Semplice (ricorda che puoi anche aggiungere hal imenti)	R6F, R18F
CreaDieta	F2	Memorizzazione dati, gestione dati, interazione con l'esterno	Complessa	[R1F-R5F], R8F, R9F, R14F, R16F, R19F
ModificaDietaDaEsistente	F2.1	Memorizzazione dati, gestione dati, interazione con l'esterno	Semplice	R13F
GestioneAlimenti	F3	Memorizzazione dati, gestione dati, interazione con l'esterno	Complessa	R15F

GestioneCalendario	F4	Interazione con l'esterno, gestione dati, memorizzazione dati	Complessa	R10F
VisualizzaCalendario	F4.1	Interazione con l'esterno	Semplice	R10F
Login	F5	Interazione con l'esterno, gestione dati	Semplice	R12F
AggiornaCredenziali	F6	Interazione con l'esterno, gestione dati, memorizzazione dati	Semplice	R10NFS
Log	F7	Interazione con l'esterno, gestione dati, memorizzazione dati	Semplice	R19FS

RegistraPaziente: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Nome	semplice	Protezione media	Input	Non più di 40 caratteri
Cognome	semplice	Protezione media	Input	Non più di 40 caratteri
DataNascita	semplice	Protezione media	Input	Non più di 10 caratteri
Telefono	semplice	Protezione media	Input	Non più di 20 caratteri
Email	semplice	Protezione media	Input	Non più di 40 caratteri

AggiornaPaziente:tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Nome	semplice	Protezione media	Input/Output	Non più di 40 caratteri
Cognome	semplice	Protezione media	Input/Output	Non più di 40 caratteri
DataNascita	semplice	Protezione media	Input/Output	Non più di 10 caratteri
Telefono	semplice	Protezione media	Input/Output	Non più di 20 caratteri
Email	semplice	Protezione media	Input/Output	Non più di 40 caratteri

AggiungiScheda: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Misure antropometriche Composto da Altezza Peso Circonferenze Pliche	Complesso	Protezione media	Input	
Dati clinici	complesso	Protezione alta	Input	

VisualizzaPaziente: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Nome	semplice	Protezione media	Output	Non più di 40 caratteri
Cognome	semplice	Protezione media	Output	Non più di 40 caratteri
DataNascita	semplice	Protezione media	Output	Non più di 10 caratteri
Telefono	semplice	Protezione media	Output	Non più di 20 caratteri
Email	semplice	Protezione media	Output	Non più di 40 caratteri
Storico Misure antropometriche Composto da: elenco in ordine cronologico dei valori delle misure antropometriche	complesso	Protezione media	Output	
Storico Dati clinici Composto da: elenco in ordine cronologico dei dati clinici	complesso	Protezione alta	Output	
Storico diete Composto da: elenco in ordine cronologico delle diete	Complesso	Protezione bassa	Output	

EsportaDieta: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Dieta Composto da: Pasti Kcal/giorno Macronutrienti/giorno Micronutrienti/settimana	Complesso	Protezione bassa	Output	No

CreaDieta: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Estremi Composto da: intervallo kcal intervallo grassi intervallo proteine intervallo carboidrati intervalli micronutrienti	Complesso	Protezione bassa	Input	No
Dieta Composto da: Pasti	Complesso	Protezione bassa	Input	No

ModificaDietaDaEsistente: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
DietaEsistente Composto da: Pasti Kcal/giorno Macronutrienti/giorno Micronutrienti/settimana	Complesso	Protezione bassa	Output	No
Estremi Composto da: intervallo kcal intervallo grassi intervallo proteine intervallo carboidrati intervalli micronutrienti	Complesso	Protezione bassa	Input	No
Dieta Composto da: Pasti	Complesso	Protezione bassa	Input	No

GestioneAlimenti: tabella delle informaizioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Nome Alimento	Semplice	Protezione bassa	Input/Output	No
Macro Alimento	Composto	Protezione bassa	Input/Output	No
Micro Alimento	Composto	Protezione bassa	Input/Output	No

GestioneCalendario: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Appuntamento Composto da: Data Oralnizio OraFine Paziente	Composto	Protezione bassa	Input/Output	No

Login:Tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Password	Semplice	Protezione alta	Input	Più di 8 caratteri

AggiornaCredenziali: tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Password corrente	Semplice	Protezione alta	Input	Più di 8 caratteri
Password nuova	Semplice	Protezione alta	Input	Più di 8 caratteri

Log:Tabella delle informazioni

Informazione	Tipo	Protezione/Privacy	I/O	Vincoli
Log Composto da: idOperazione Timestamp	Complesso	Protezione alta	Output	No

Analisi dei Vincoli:

Requisito	Categorie	Impatto	Funzionalità
Navigabilità (R1NF)	Usabilità	Cercare di migliorare	F1, F2, F3, F4
Velocità scrittura (R2NF)	Tempo di risposta	Cercare di migliorare	F1.1, F1.4, F2, F2.1, F3, F4
Velocità lettura(R3NF)	Tempo di risposta	Cercare di migliorare	F1.2, F1.3, F2.1, F3
Univocità dati (R4NF, R5NF, R6NF, R8NF)	Funzionamento	Cercare di migliorare	F1

Analisi delle Interazioni:

Tabella Maschere

Nome	Informazioni	Funzionalità
ViewLogin	Username e Password	F5
ViewHome	Possibilità di Navigazione a ViewCalendario, Possibilità di navigazione verso ViewPazienti, possibilità di navigazione verso ViewAlimenti	F1, F2, F3, F4
ViewPazienti	Lista dei pazienti, Input di ricerca pazienti, possibilità di passare a DettaglioPaziente, possibilità di registrare paziente	F1
DettaglioPaziente	Nome e cognome del paziente, Data di nascita, parametri antropometrici, dieta corrente, possibilità di visualizzare diete passate, possibilità di esportare dieta corrente, possibilità di creare dieta	F1.2, F1.5
ViewCreaDieta	Elenco parametri soglia, elenco pasti, elenco alimenti per pasto	F2
ViewRegistraPaziente	Nome e cognome, data di nascita, mail, telefono	F1.1
ViewAlimenti	Elenco alimenti con nome e valori nutritivi	F3
ViewCalendario	Calendario settimanale navigabile con gli orari	F4
DettaglioAppuntamento	Nome e cognome paziente, ora inizio, ora fine	F4

Tabella Ruoli/Responsabilità

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Nutrizionista	Amministrazione e utilizzo dell'applicazione	Sì	È richiesto un alto grado di riservatezza	1

Tabella Informazione/Tipo di accesso Nutrizionista

Informazione	Tipo di accesso
Nome Paziente	Lettura/Scrittura
Cognome Paziente	Lettura/Scrittura
DataNascita Paziente	Lettura/Scrittura
Telefono Paziente	Lettura/Scrittura
Email Paziente	Lettura/Scrittura
Storico Paziente	Lettura
Dati Correnti Paziente	Lettura/Scrittura
Calendario	Lettura/Scrittura
Base dati alimenti	Lettura/Scrittura

Si sceglie di dare accesso in scrittura anche a dati considerabili stabili poiché non compromette la sicurezza dell'applicazione e permette la correzione di eventuali errori.

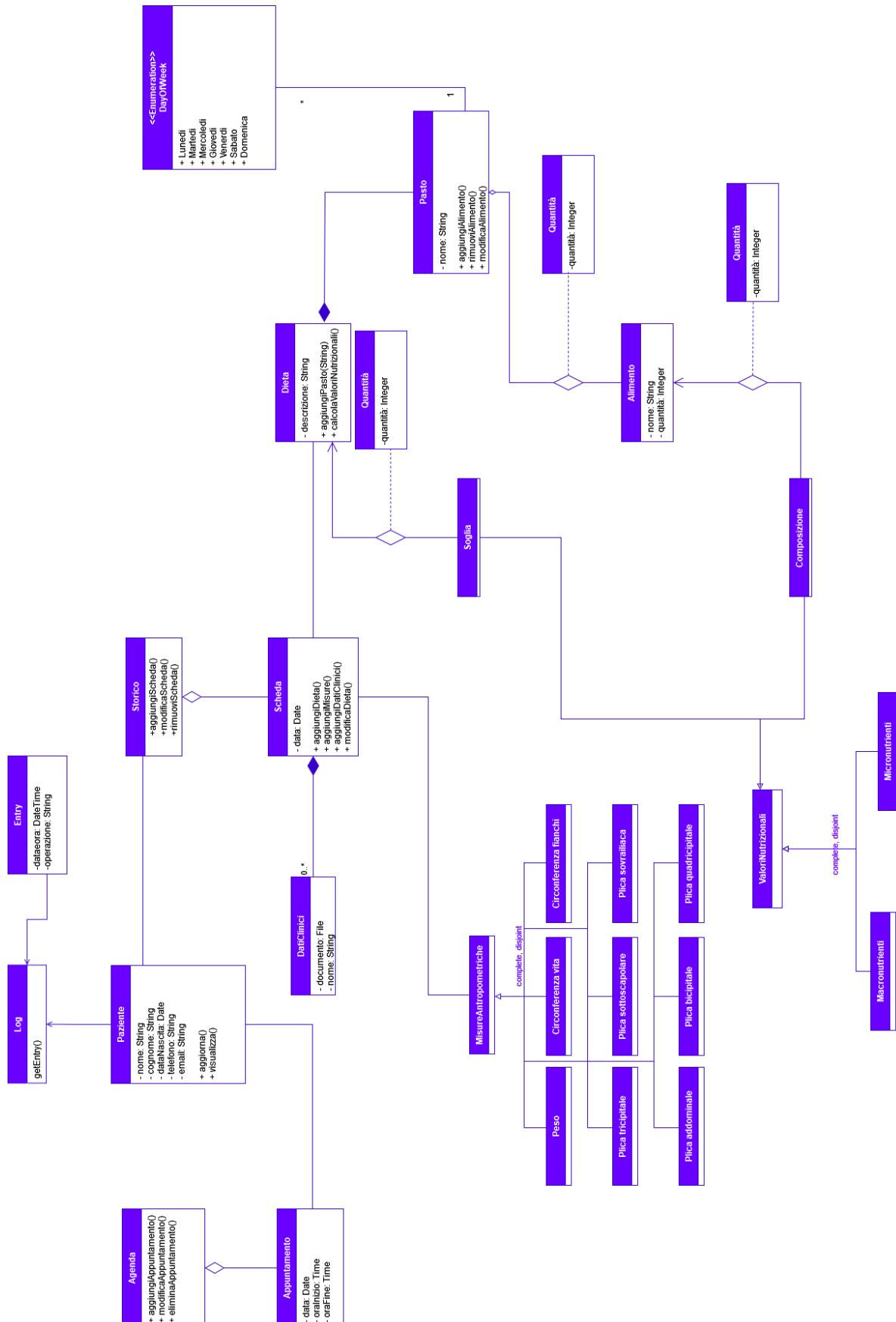
Scomposizione del Problema

Funzionalità	Sotto-funzionalità
CreaDieta	InserisciPasto, AssociaDieta, GestioneNutrienti
GestioneCalendario	VisualizzaCalendario, AggiungiAppuntamento, ModificaAppuntamento, EliminaAppuntamento
GestioneAlimenti	VisualizzaAlimento, ModificaAlimento, AggiungiAlimento, EliminaAlimento

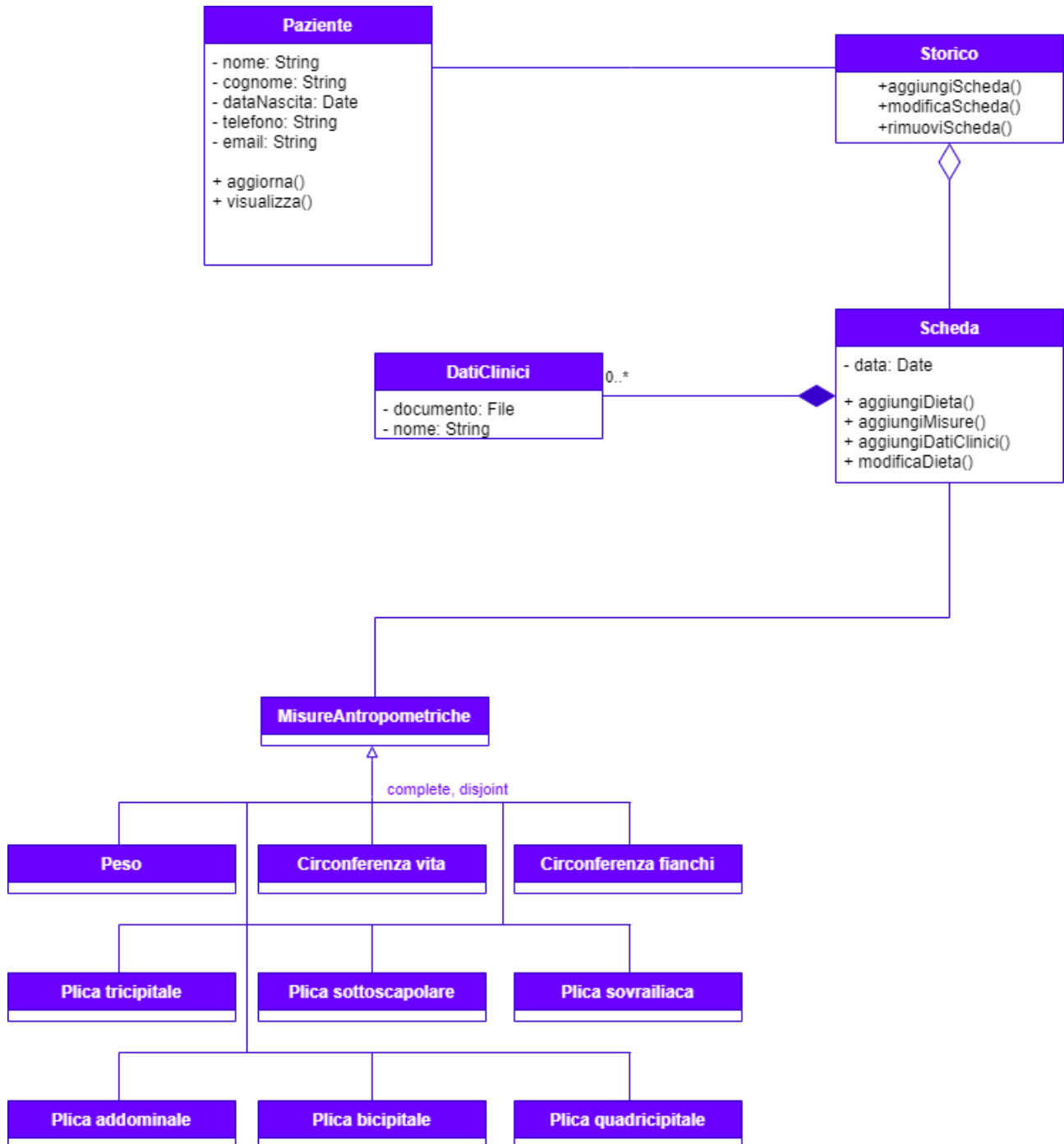
Sotto-funzionalità	Funzionalità	Legame	Informazioni
InserisciPasto	CreaDieta	Si possono inserire pasti solo durante la creazione di una dieta	Pasto (insieme di coppie (alimento, quantità))
AssociaDieta	CreaDieta	Solo alla fine della creazione di una dieta è possibile associarla al paziente	Identificativo del paziente
GestioneNutrienti	CreaDieta	Interazione con l'esterno, gestione dati	Soglie dei valori nutrizionali
VisualizzaAlimento	GestioneAlimenti	In gestione alimenti è possibile visualizzare i dettagli di un alimento	Nome e valori nutrizionali alimento
ModificaAlimento	GestioneAlimenti	In gestione alimenti è possibile modificare i dettagli di un alimento	Nome e valori nutrizionali alimento
InserisciAlimento	GestioneAlimenti	In gestione alimenti è possibile inserire un alimento nell'elenco	Nome e valori nutrizionali alimento
EliminaAlimento	GestioneAlimenti	In gestione alimenti è possibile eliminare un alimento dall'elenco	Nome alimento
VisualizzaCalendario	GestioneCalendario	In gestione calendario è possibile visualizzare gli appuntamenti	Appuntamenti
ModificaAppuntamento	GestioneCalendario	In gestione alimenti è possibile modificare i dettagli di un appuntamento	Identificativo cliente, data e ora appuntamento
InserisciAppuntamento	GestioneCalendario	In gestione alimenti è possibile inserire un appuntamento nell'elenco	Identificativo cliente, data e ora appuntamento
EliminaAppuntamento	GestioneCalendario	In gestione alimenti è possibile eliminare un appuntamento dall'elenco	Identificativo cliente e data e ora appuntamento

Modello del dominio

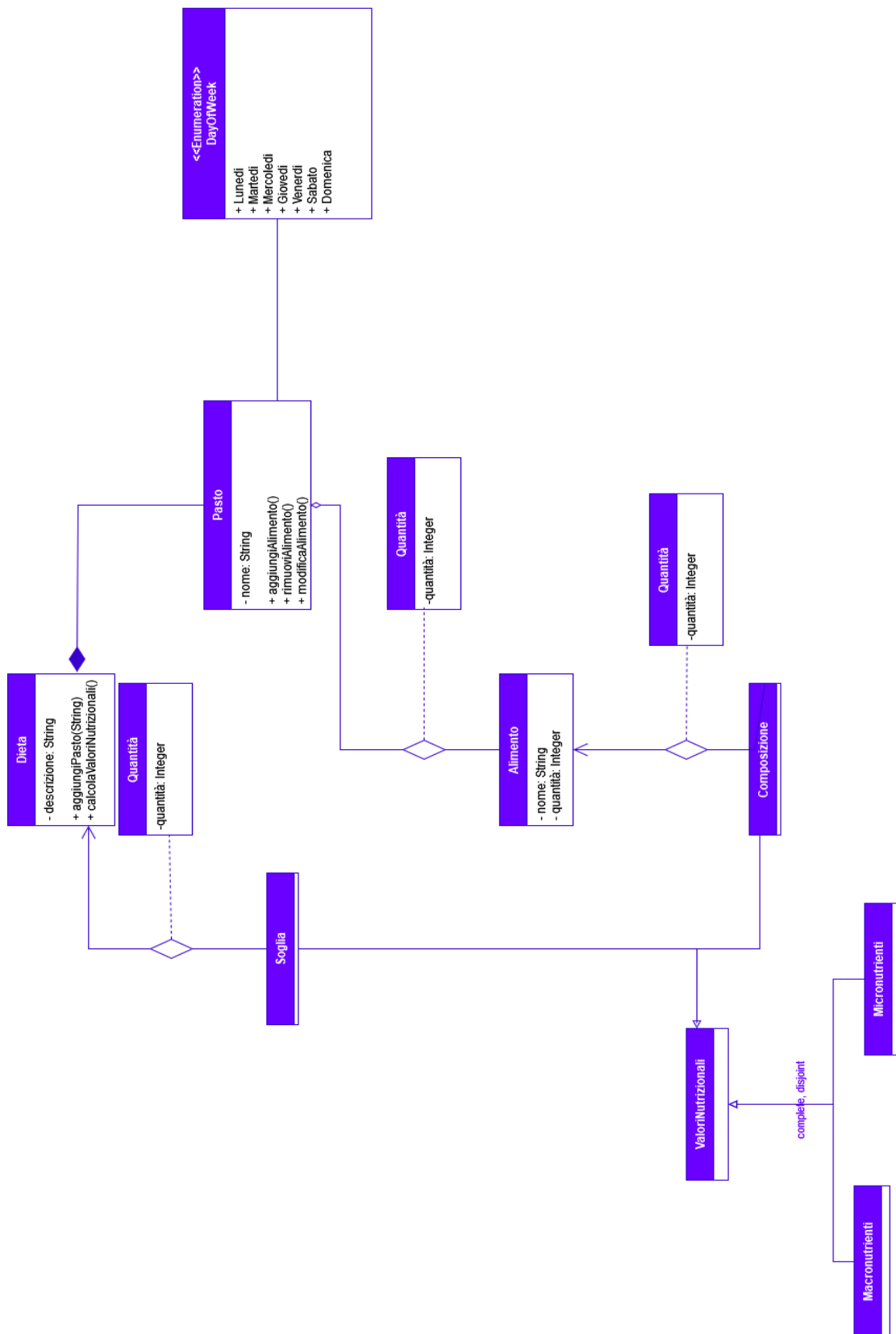
Di seguito è riportato il diagramma delle classi del modello del dominio.



Di seguito è mostrato il diagramma delle classi del modello del dominio relativo alla gestione del paziente.



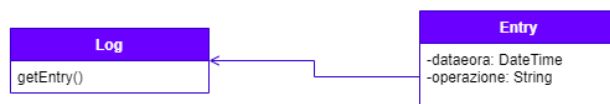
Di seguito è riportato il diagramma delle classe del modello del dominio relativo alla gestione della dieta.



Di seguito è riportato il diagramma delle classi del modello del dominio relativo alla gestione del calendario.



Di seguito è riportato il diagramma delle classi del modello del dominio relativo alla gestione dei log.



Architettura logica Struttura

Diagramma dei package

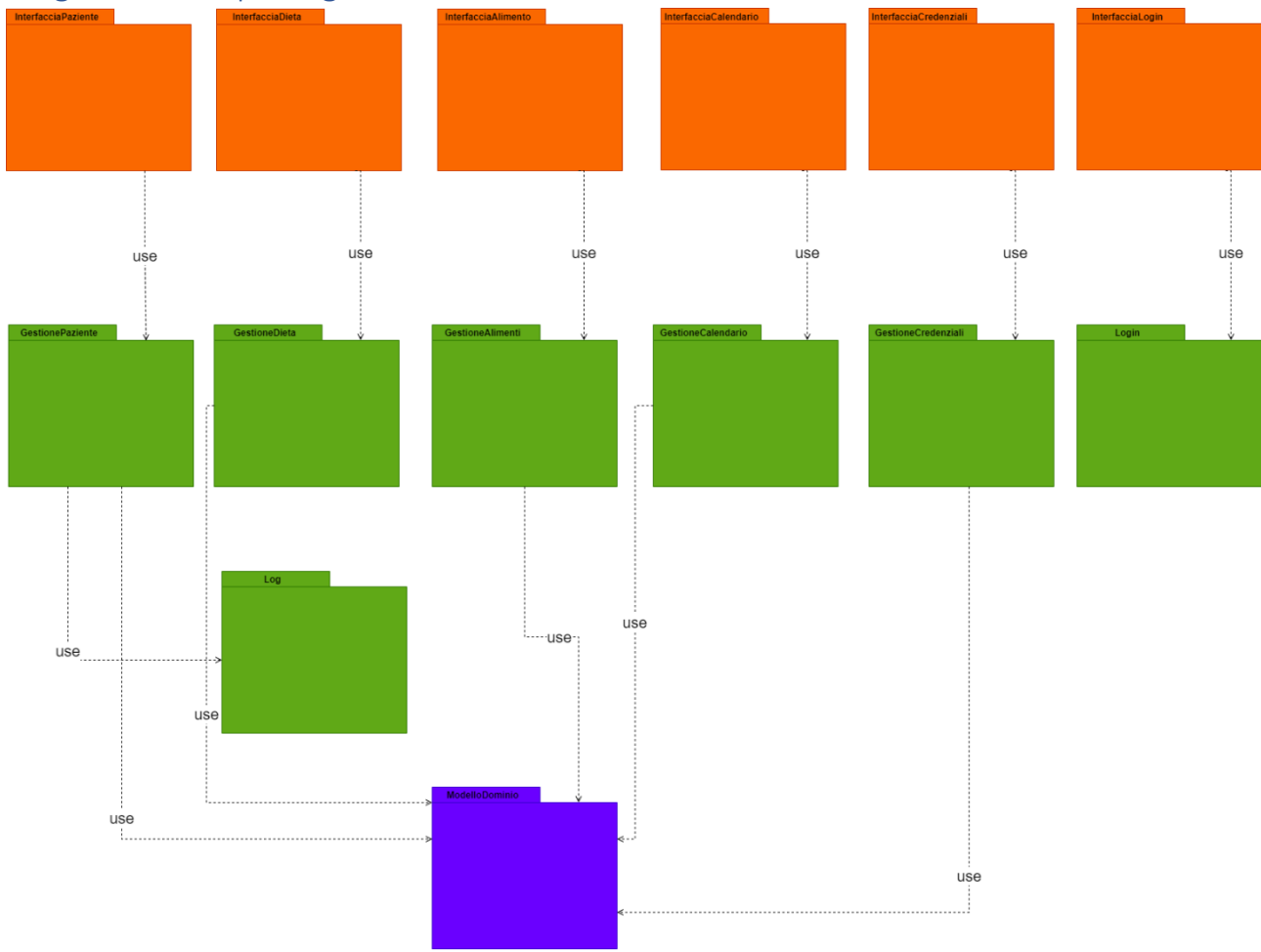


Diagramma delle classi: Dominio

Non viene riportato il diagramma delle classi associato al package Dominio in quanto è il modello del dominio creato nella fase precedente.

Diagramma delle classi: GestionePaziente e GestioneDieta

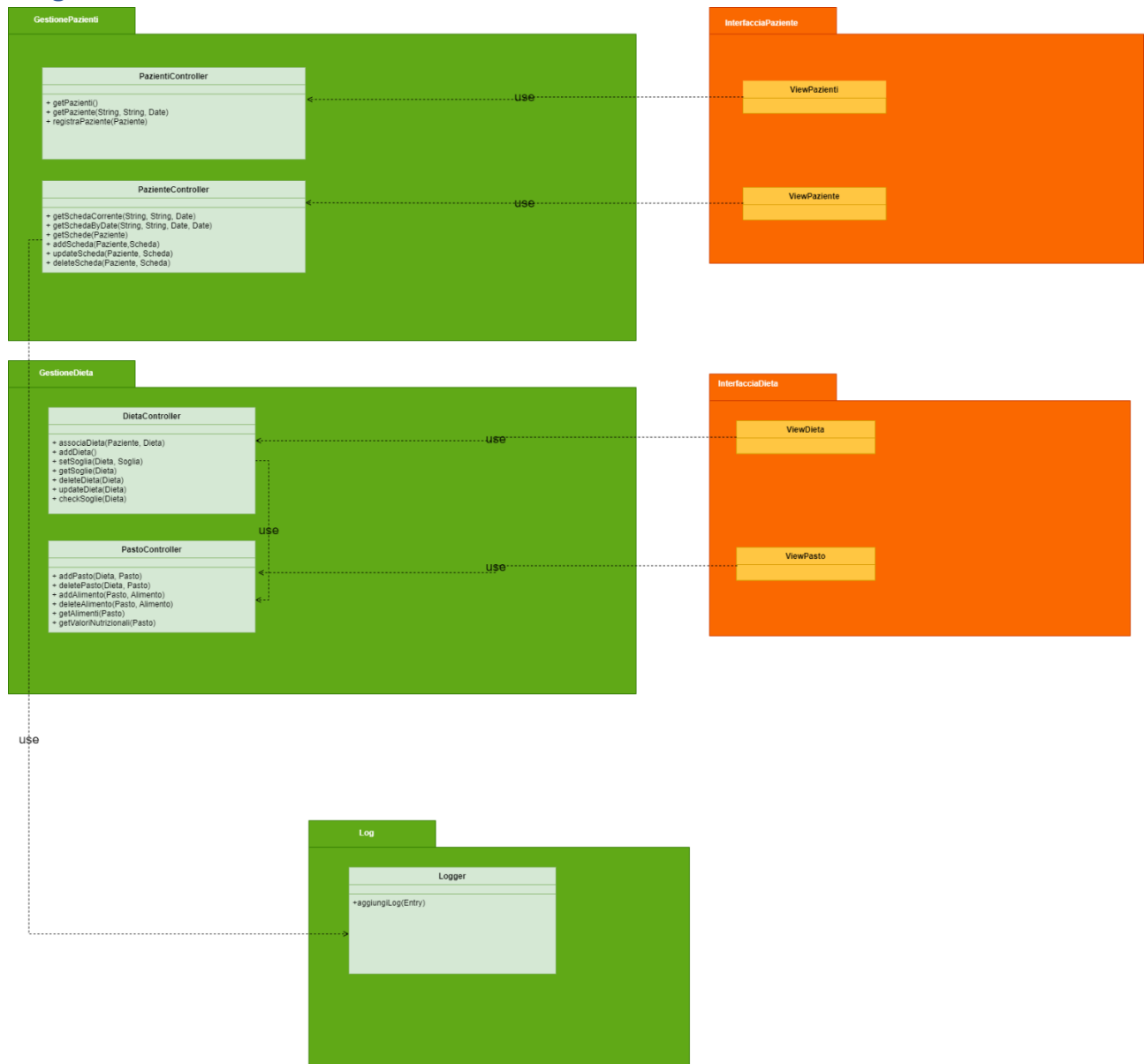


Diagramma delle classi: GestioneAlimenti

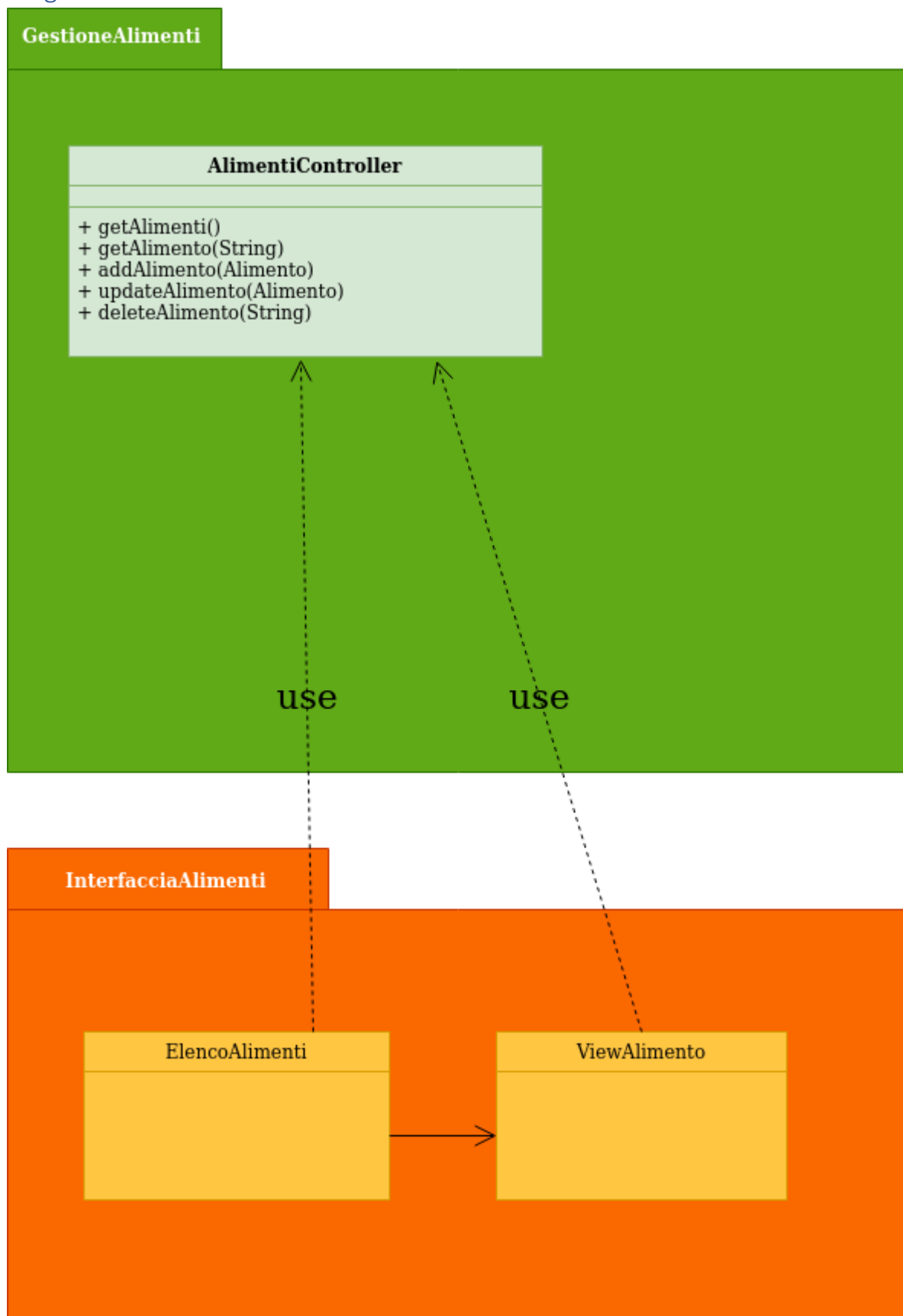


Diagramma delle classi: GestioneCalendario

GestioneCalendario

CalendarioController

- + getAppuntamenti(Date, Date)
- + getCalendario()
- + getAppuntamentiCliente(Cliente)
- + addAppuntamento(Cliente, DateTime, Time)
- + updateAppuntamento(Appuntamento)
- + deleteAppuntamento(Appuntamento)

use

InterfacciaCalendario

ViewCalendario

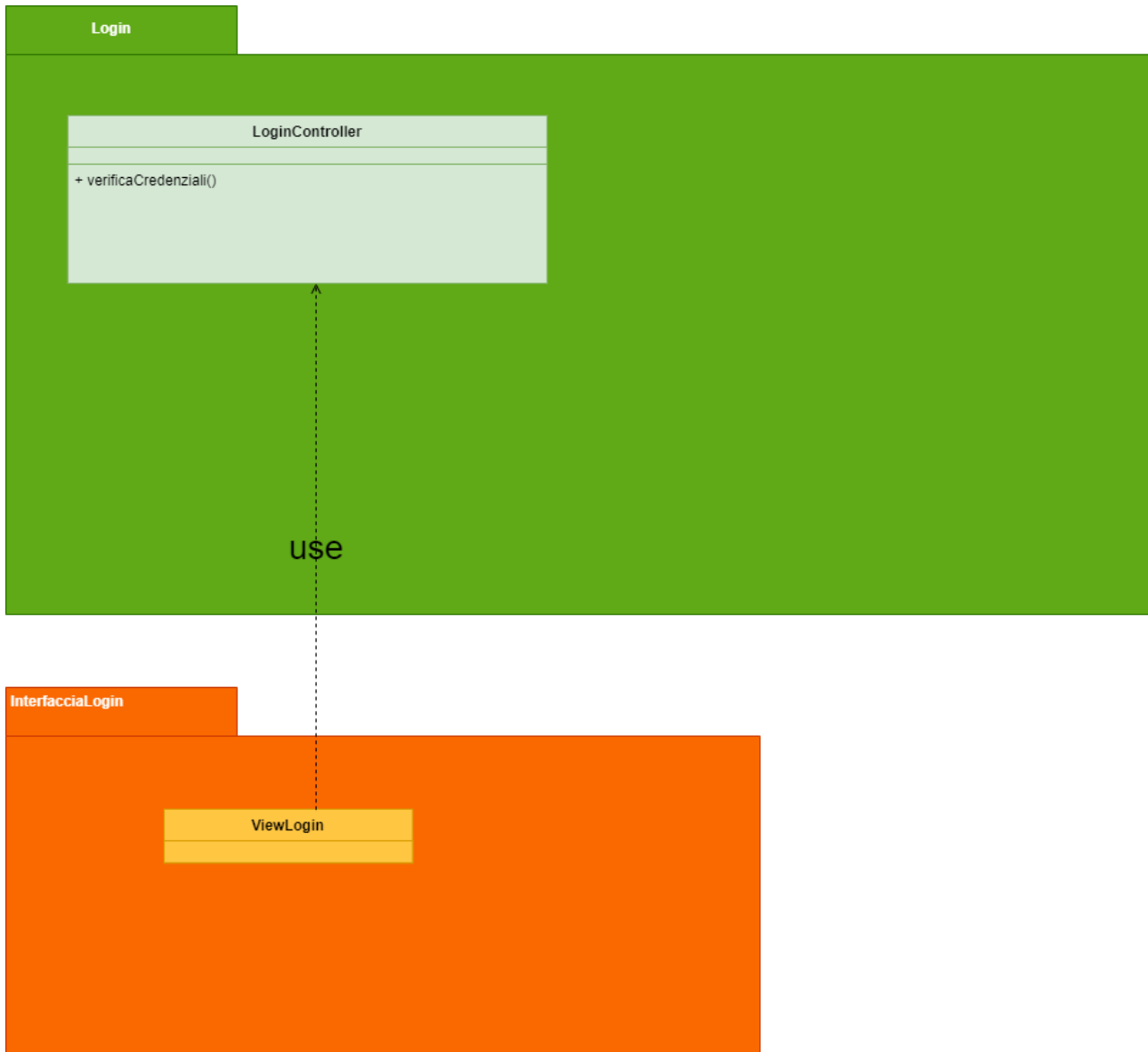
ViewAppuntamento



Diagramma delle classi: GestioneCredenziali



Diagramma delle classi: Login



Architettura Logica: Interazione

Diagramma di Sequenza Login avvenuto con successo

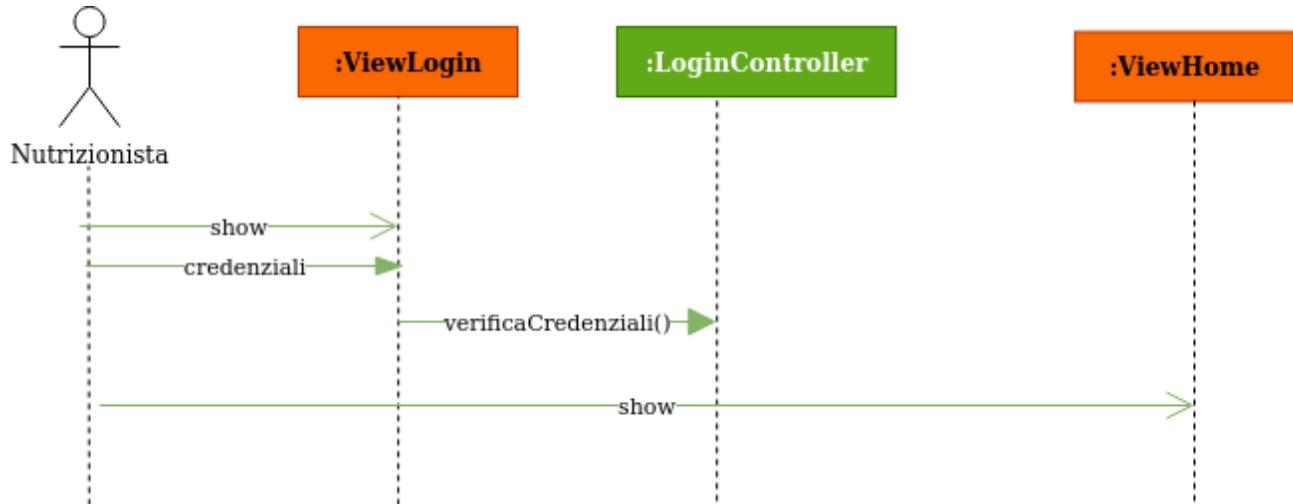


Diagramma di Sequenza Aggiunta Alimento alla base dati e visualizzazione

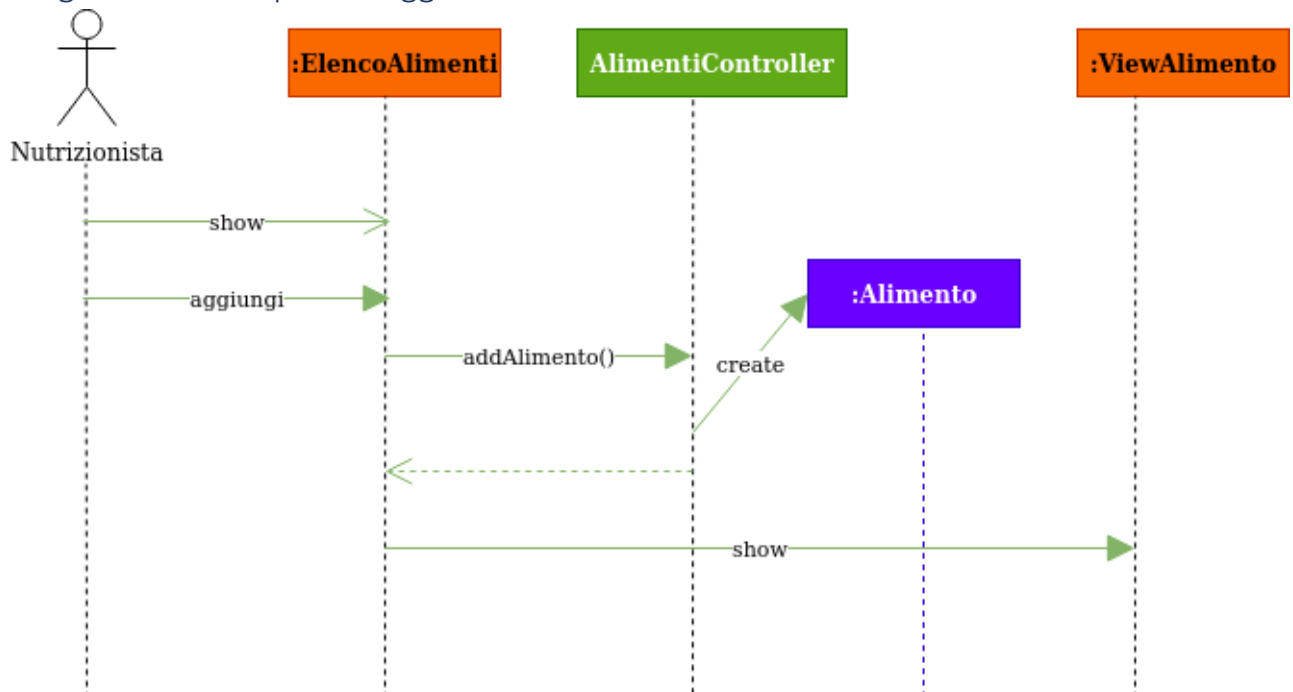


Diagramma di Sequenza: Aggiunta appuntamento e modifica

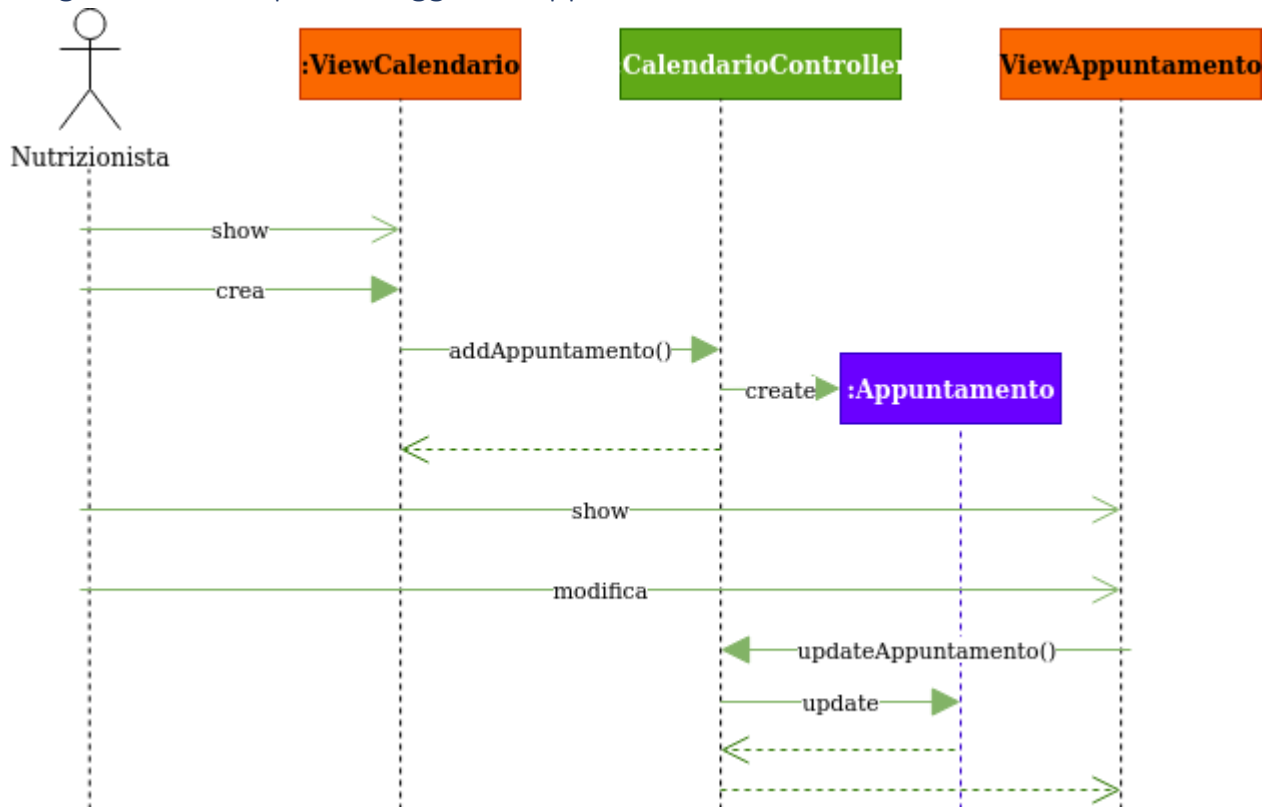


Diagramma di Sequenza Aggiunta Scheda a Paziente

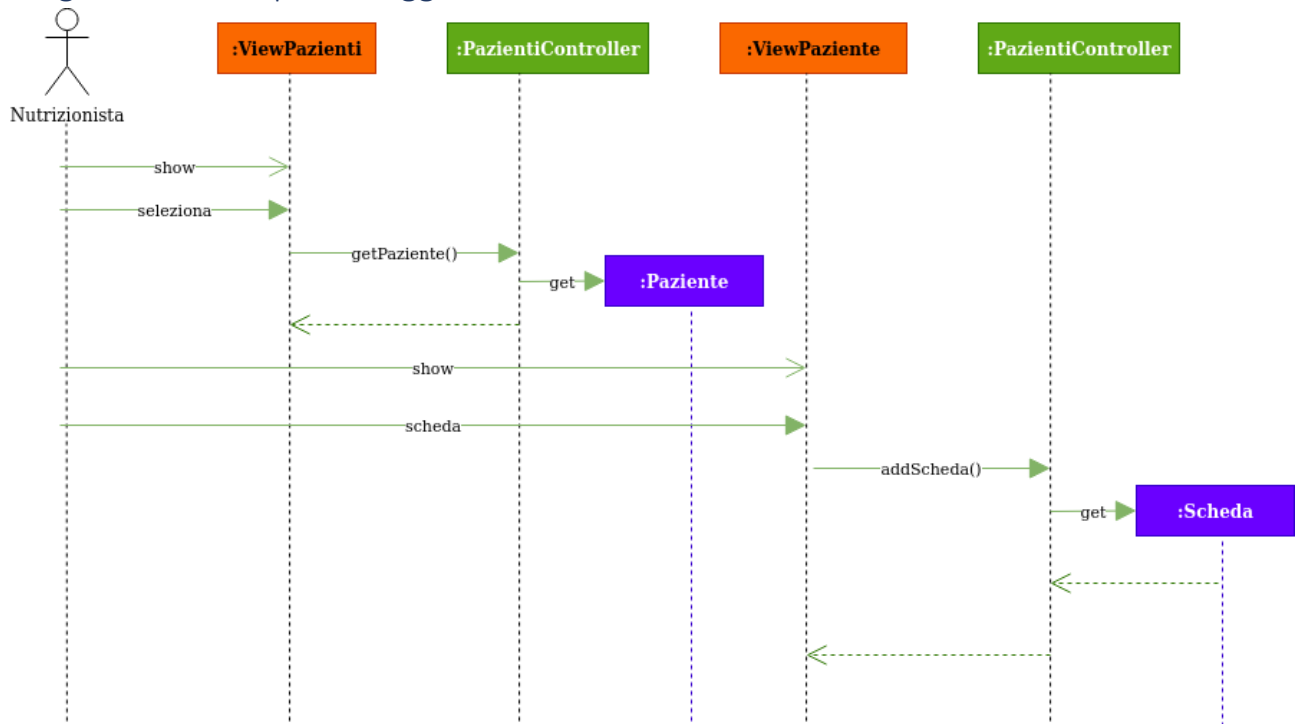
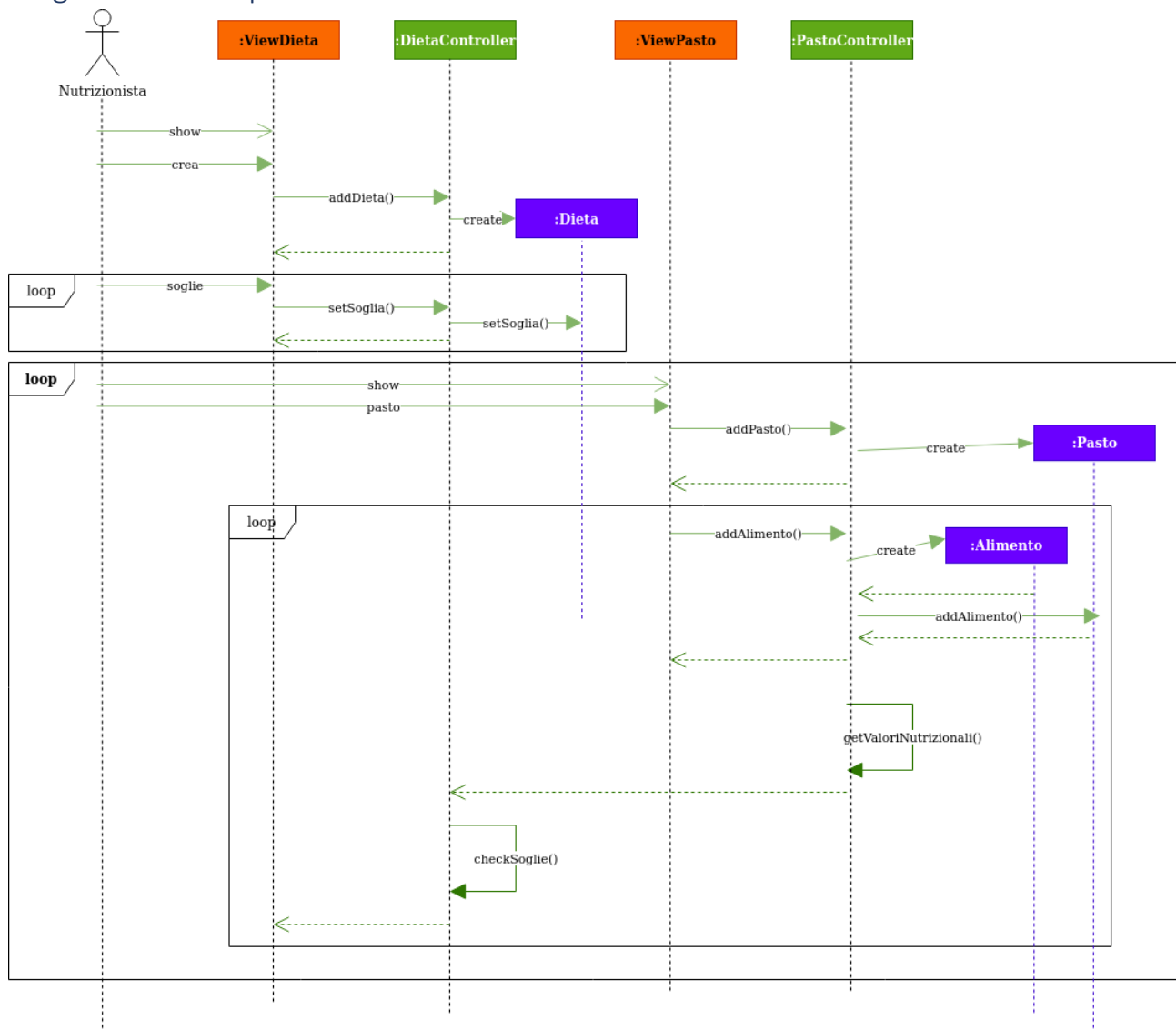
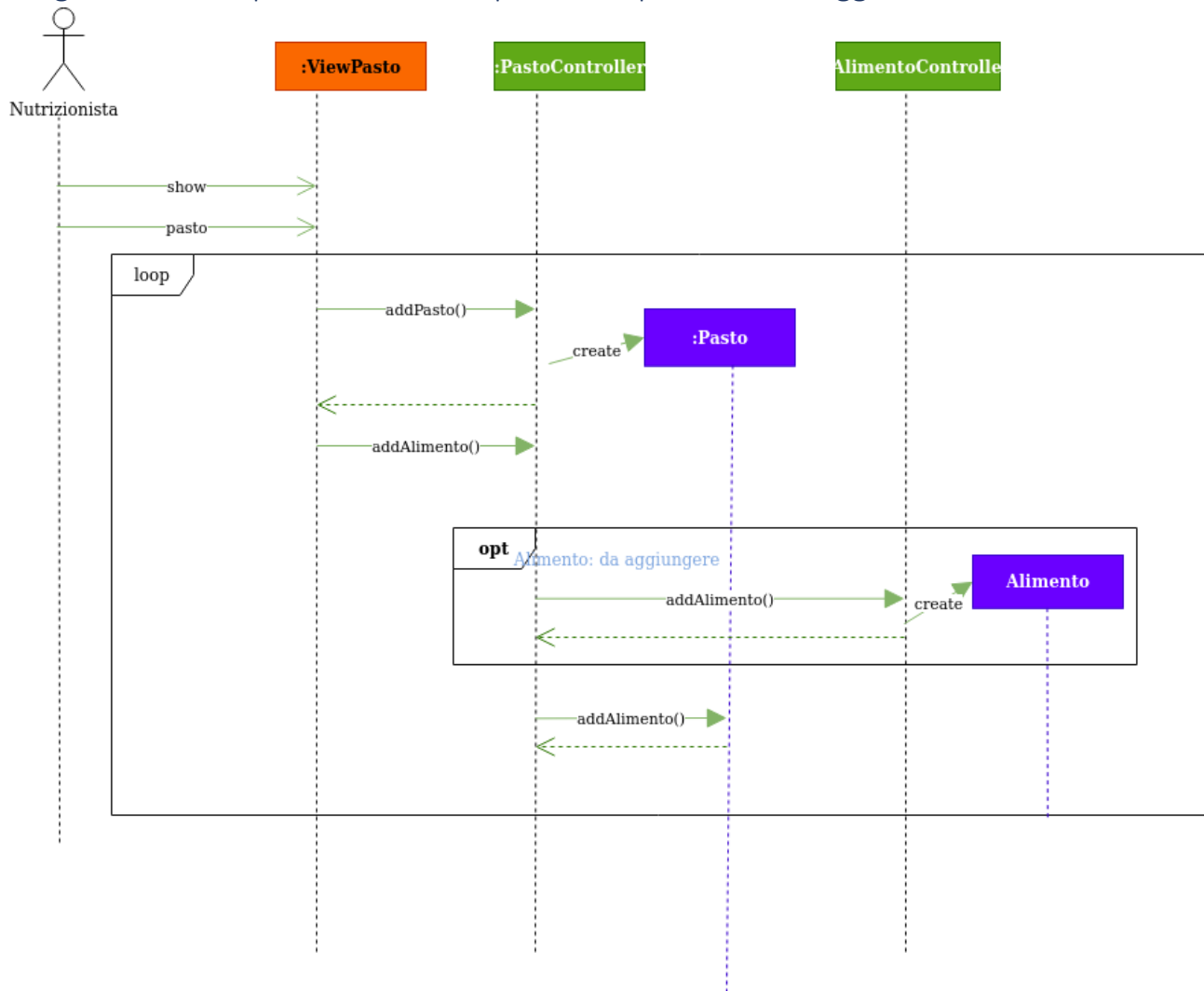


Diagramma di Sequenza Creazione Dieta



In particolare, data la possibilità di aggiungere alimenti alla base dati durante la creazione di una dieta, il comportamento all'interno del loop di aggiunta pasto è più nel dettaglio il seguente

Diagramma di sequenza creazione pasto con possibilità di aggiunta alimenti al db



Architettura Logica: Comportamento

Diagramma di Stato: Dieta

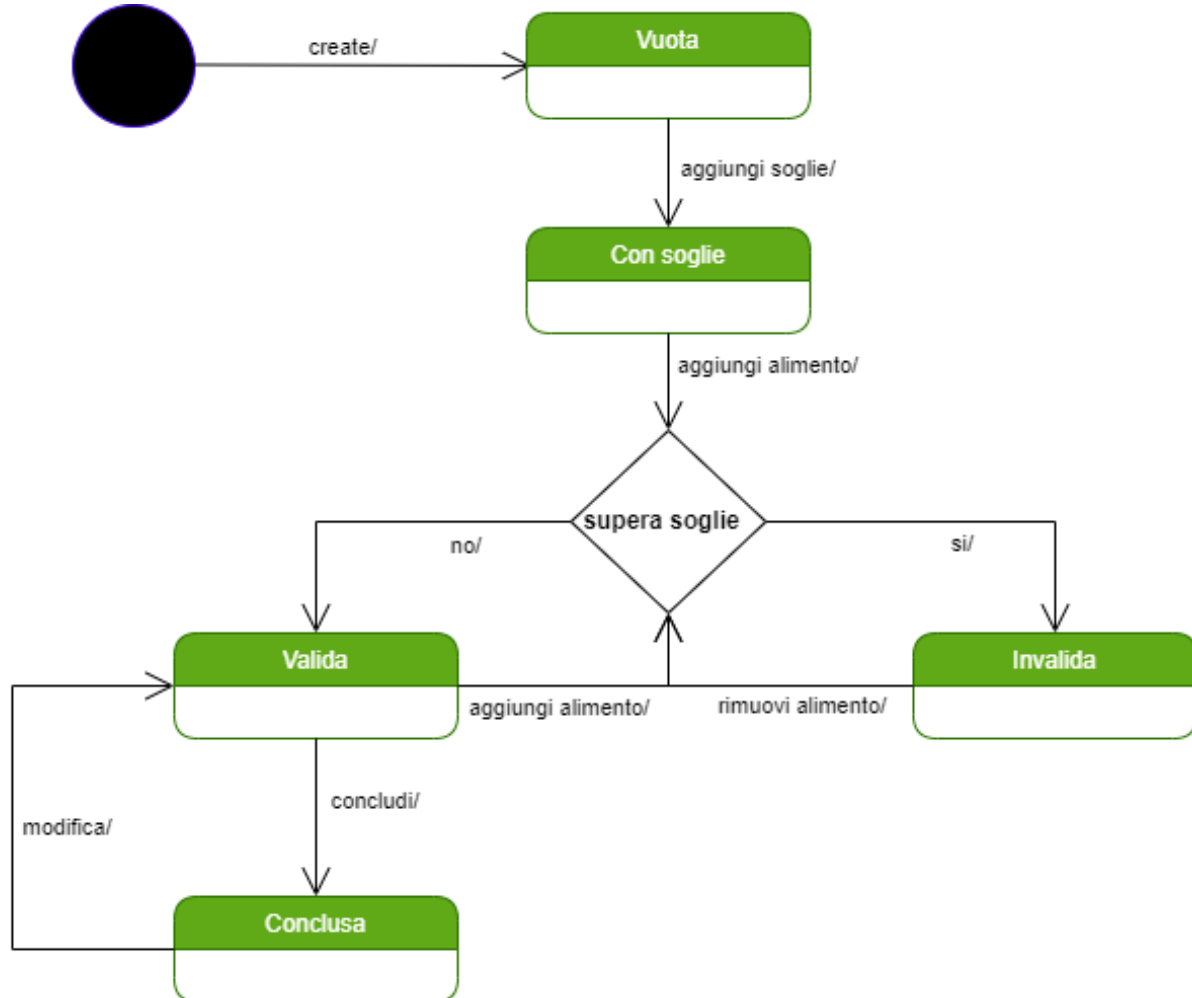
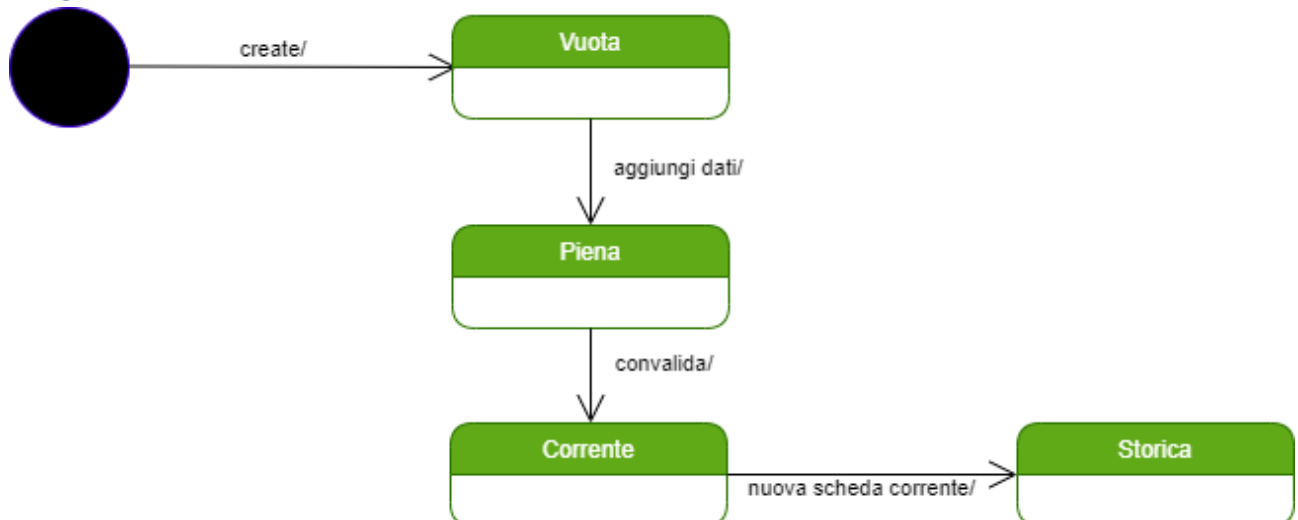


Diagramma di Stato: Scheda



Piano di Lavoro

Il progetto e lo sviluppo del sistema sono assegnati a diversi team come indicato nella tabella sottostante:

Nome team	Composizione
Team progettazione	Cesaretti, De Dominicis, Legnini
Team sviluppo	Cesaretti, De Dominicis, Legnini
Team DB	Cesaretti, De Dominicis, Legnini
Team sicurezza	Cesaretti, De Dominicis, Legnini
Team grafico	Cesaretti, De Dominicis, Legnini

Package	Progetto	Sviluppo
Dominio	Team progettazione + Team DB	Team sviluppo + Team DB
GestionePaziente	Team progettazione	Team sviluppo
GestioneDieta	Team progettazione	Team sviluppo
GestioneAlimenti	Team progettazione	Team sviluppo
GestioneCalendario	Team progettazione	Team sviluppo
GestioneCredenziali	Team sicurezza	Team sicurezza + Team sviluppo
Login	Team sicurezza	Team sicurezza + Team sviluppo
Log	Team sicurezza	Team sicurezza + Team sviluppo
InterfacciaPaziente	Team grafico + Team progettazione	Team sviluppo + Team grafico
InterfacciaDieta	Team grafico + Team progettazione	Team sviluppo + Team grafico
InterfacciaAlimenti	Team grafico + Team progettazione	Team sviluppo + Team grafico
InterfacciaCalendario	Team grafico + Team progettazione	Team sviluppo + Team grafico
InterfacciaCredenziali	Team grafico + Team progettazione	Team sviluppo + Team grafico
InterfacciaLogin	Team grafico + Team progettazione	Team sviluppo + Team grafico

I tempi di rilascio previsti sono i seguenti:

- Progettazione entro 2 settimane dalla data odierna
- Sviluppo delle singole parti con collaudo unitario entro 3 settimane rispetto al fine della progettazione
- Integrazione e test dell'intero sistema entro 2 settimane rispetto alla fine dello sviluppo

Piano del Collaudo

```
public class TestPaziente{

    Paziente paziente;
    Scheda scheda;

    @Before
    private void inizializza(){

        String nome= "Luigi";
        String cognome= "Mario";
        LocalDate dataNascita=LocalDate.of(year: 2001, month: 9, dayOfMonth: 11);
        String telefono="333 33 33 333";
        String mail= "mario.mario@gmail.com";
        List<Scheda> storico = new ArrayList<Scheda>();

        float peso=70;
        float circonferenzaVita=102;
        float circonferenzaFianchi=105;
        float plicaTricipitale=4;
        float plicaSottoscapolare=6;
        float plicaSovrailliaca=5;
        float plicaAddominale=9;
        float plicaBicipitale=7;
        float plicaQuadricipitale=8;

        scheda=new Scheda(schedaId: 0, peso, circonferenzaVita, circonferenzaFianchi, plicaTricipitale,
            plicaSottoscapolare, plicaSovrailliaca, plicaAddominale, plicaBicipitale,
            plicaQuadricipitale, new HashMap<String, Path>(), new Dieta());

        storico.add(scheda);
        paziente=new Paziente(pazienteId: 0, nome, cognome, dataNascita, telefono, mail, storico);

    }

}
```

```
@Test
private void testGettersPaziente(){
    assertEquals(paziente.getNome(), "Luigi");
    assertEquals(paziente.getCognome(), "Mario");
    assertEquals(paziente.getDataNascita(), LocalDate.of(year: 2001, month: 9, dayOfMonth: 11));
    assertEquals(paziente.getTelefono(), "333 33 33 333");
    assertEquals(paziente.getMail(), "mario.mario@gmail.com");
}

@Test
private void testSettersPaziente(){
    paziente.setName(nome: "Mario");
    paziente.setCognome(cognome: "Luigi");
    paziente.setMail(mail: "luigi.mario@gmail.com");
    paziente.setTelefono(telefono: "4206942069");
    paziente.setDataNascita(LocalDate.now());
    assertEquals(paziente.getCognome(), "Luigi");
    assertEquals(paziente.getNome(), "Mario");
    assertEquals(paziente.getMail(), "luigi.mario@gmail.com");
    assertEquals(paziente.getTelefono(), "4206942069");
    assertEquals(paziente.getDataNascita(), LocalDate.now());
}

}
```

```

@Test
private void testGettersSettersScheda(){
    Scheda scheda = paziente.getStorico().getScheda(schedaId: 0);

    assertEquals(scheda.getPeso(), 70);
    assertEquals(scheda.getCirconferenzaVita(), 102);
    assertEquals(scheda.getCirconferenzaFianchi(), 105);
    assertEquals(scheda.getPlicaTricipitale(), 4);
    assertEquals(scheda.getPlicaSottoscapolare(), 6);
    assertEquals(scheda.getPlicaSovrailliacca(), 5);
    assertEquals(scheda.getPlicaAddominale(), 9);
    assertEquals(scheda.getPlicaBicipitale(), 7);
    assertEquals(scheda.getPlicaQuadricipitale(), 8);

    float peso=100;
    float circonferenzaVita=150;
    float circonferenzaFianchi=180;
    float plicaTricipitale=10;
    float plicaSottoscapolare=7;
    float plicaSovrailliacca=9;
    float plicaAddominale=2;
    float plicaBicipitale=1;
    float plicaQuadricipitale=20;

    scheda.setPeso(peso);
    scheda.setCirconferenzaFianchi(circonferenzaFianchi);
    scheda.setCirconferenzaVita(circonferenzaVita);
    scheda.setPlicaTricipitale(plicaTricipitale);
    scheda.setPlicaSottoscapolare(plicaSottoscapolare);
    scheda.setPlicaSovrailliacca(plicaSovrailliacca);
    scheda.setPlicaBicipitale(plicaBicipitale);
    scheda.setPlicaAddominale(plicaAddominale);
    scheda.setPlicaQuadricipitale(plicaQuadricipitale);

    assertEquals(scheda.getPeso(), 100);
    assertEquals(scheda.getCirconferenzaVita(), 150);
    assertEquals(scheda.getCirconferenzaFianchi(), 180);
    assertEquals(scheda.getPlicaTricipitale(), 10);
    assertEquals(scheda.getPlicaSottoscapolare(), 7);
    assertEquals(scheda.getPlicaSovrailliacca(), 9);
    assertEquals(scheda.getPlicaAddominale(), 2);
    assertEquals(scheda.getPlicaBicipitale(), 1);
    assertEquals(scheda.getPlicaQuadricipitale(), 20);
}

```

```

    peso=70;
    circonferenzaVita=102;
    circonferenzaFianchi=105;
    plicaTricipitale=4;
    plicaSottoscapolare=6;
    plicaSovrailliacca=5;
    plicaAddominale=9;
    plicaBicipitale=7;
    plicaQuadricipitale=8;
    Scheda secondaScheda = new Scheda(schedaId: 1, peso, circonferenzaVita, circonferenzaFianchi, plicaTricipitale,
    plicaSottoscapolare, plicaSovrailliacca, plicaAddominale, plicaBicipitale,
    plicaQuadricipitale, new HashMap<String, Path>(), new Dieta());

    paziente.getStorico().aggiungiScheda(secondaScheda);
    Scheda scheda2 = paziente.getStorico().getScheda(schedaId: 1);

    assertEquals(scheda2.getPeso(), 70);
    assertEquals(scheda2.getCirconferenzaVita(), 102);
    assertEquals(scheda2.getCirconferenzaFianchi(), 105);
    assertEquals(scheda2.getPlicaTricipitale(), 4);
    assertEquals(scheda2.getPlicaSottoscapolare(), 6);
    assertEquals(scheda2.getPlicaSovrailliacca(), 5);
    assertEquals(scheda2.getPlicaAddominale(), 9);
    assertEquals(scheda2.getPlicaBicipitale(), 7);
    assertEquals(scheda2.getPlicaQuadricipitale(), 8);
}

```

Progettazione

Requisiti non funzionali

Nell'Analisi del Problema (Tabella Vincoli) sono emersi i seguenti requisiti non funzionali che impongono dei vincoli al sistema:

1. L'applicazione deve poter essere eseguita in ambiente locale sul PC del cliente che usa Windows 10 come sistema operativo.
2. L'applicazione deve essere rapida e facilmente navigabile.
3. L'applicazione deve garantire la protezione dei dati sensibili dei pazienti per mezzo di meccanismi di cifratura.
4. L'applicazione deve controllare gli accessi ai suddetti dati.

Uno degli aspetti chiave dell'applicazione è l'usabilità: la piattaforma deve essere intuitiva e semplice da utilizzare per soddisfare i requisiti delineati dal cliente. Allo stesso tempo si vuole che i dati dei pazienti vengano tutelati, per cui è fondamentale trovare il giusto compromesso tra semplicità, rapidità del sistema e sicurezza dei dati.

La cifratura e il controllo degli accessi permette al cliente di monitorare la piattaforma al fine di garantire la sicurezza dei dati. Abbiamo deciso di evitare rallentamenti inutili dell'applicazione e complicazioni cifrando solo i dati personali dei pazienti e scrivendo i log solo delle operazioni di lettura e scrittura che li riguardano.

La natura concentrata dell'applicazione ci tutela sia dal punto di vista della sicurezza che da quello dell'efficienza, evitando rallentamenti dovuti a problemi di connettività.

Scelta dell'architettura

Dal punto di vista architetturale, essendo l'applicazione non distribuita, la scelta più adatta risulta essere il pattern MVP per separare i componenti dell'applicazione.

L1 - Applicazione

L'applicazione avrà modo di interagire con l'utente (il nutrizionista), e di interfacciarsi con il DBMS in locale al fine di recuperare i dati necessari e/o aggiungere nuovi dati o modificare quelli già esistenti. Essendo l'applicazione concentrata su un singolo host non sono previsti meccanismi di sicurezza raffinati durante l'interazione tra le due entità, in quanto nessun dato sensibile lascerà mai il calcolatore.

L2 - Database

Al fine di mantenere i dati persistenti si è scelto l'utilizzo di un opportuno DBMS che gestirà la base dati della piattaforma (il cui schema di dati si può trovare nelle pagine successive). La metodologia Object Relational Mapping (ORM) e una sua diretta implementazione, ovvero il framework Hibernate, permetterà l'interfacciamento con il DBMS.

Per quanto riguarda i log, questi saranno salvati in un file apposito nel calcolatore del cliente.

Diagramma package

Nella figura sottostante è riportata l'Architettura del nostro Sistema organizzata in un diagramma dei package.

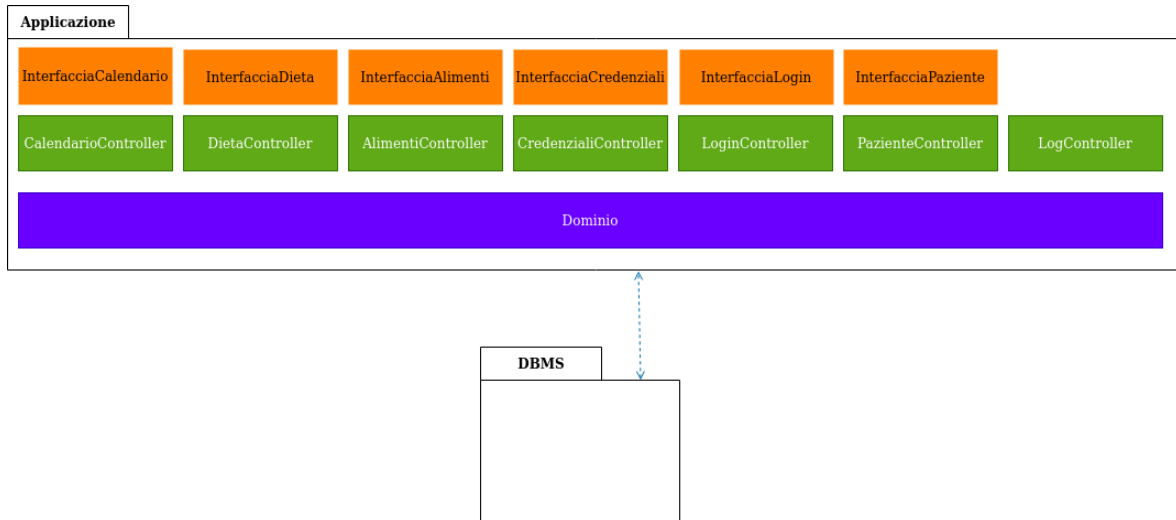
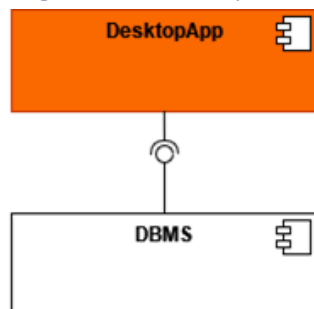


Diagramma componenti



Abbiamo optato per il diagramma di cui sopra perché essendo la natura della nostra applicazione monolitica, non abbiamo individuato altri componenti che non fossero l'applicazione stessa e il DBMS. L'applicazione sarà distribuita su un singolo archivio .jar. Il DBMS sarà preventivamente installato sulla macchina del cliente.

Scelte tecnologiche

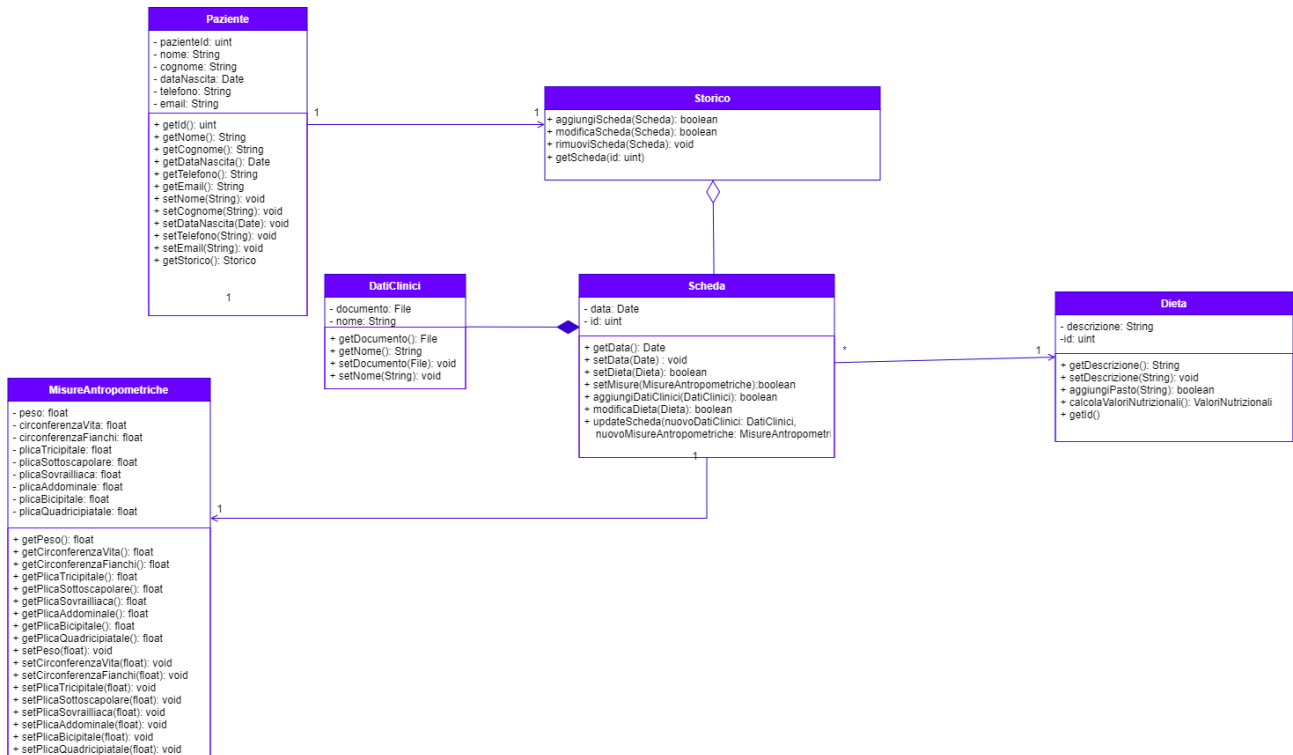
Per le interfacce grafiche abbiamo deciso di utilizzare javaFX con il tool SceneBuilder, il quale genera file fxml. Per aumentare il disaccoppiamento fra view e presenter i file .fxml saranno organizzati in un package (view) diverso dalle classi java che controllano le view (presentation). In questo modo aumentiamo il grado di disaccoppiamento del sistema. Per l'interfacciamento con il Database utilizzeremo il framework Hibernate. Per il deployment utilizzeremo l'archivio con estensione .jar. L'abbiamo ritenuta la scelta migliore in quanto la JVM permette omogeneità tra sistemi diversi, in questo modo non dobbiamo preoccuparci troppo dell'installazione sulla macchina del cliente.

Progettazione di dettaglio

Nel seguito si riportano i diagrammi di dettaglio delle varie parti del Sistema.

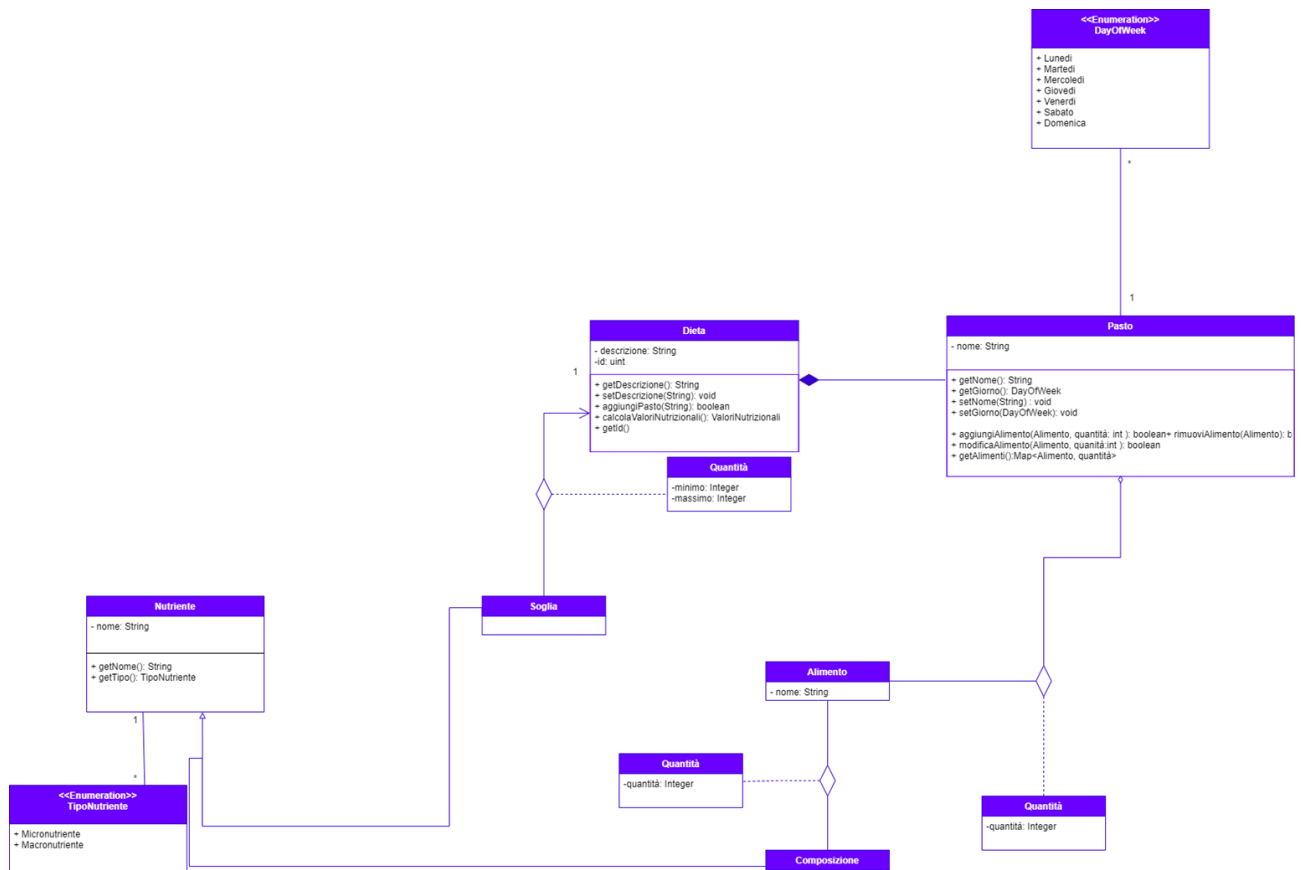
Struttura

Diagramma di Dettaglio: Dominio – Paziente



Scheda: l'oggetto scheda ci permette di recuperare i dati clinici e biometrici di un paziente al momento della visita, con eventuale assegnazione di una nuova dieta. Questa è un'informazione critica per il nostro cliente, e per questo salviamo tutte le schede in uno storico.

Diagramma di Dettaglio: Dominio – Dieta



Nutriente: abbiamo deciso di sfruttare l’ereditarietà per esprimere la differenza fra soglia e composizione: le soglie hanno un valore di massimo e uno di minimo, e le utilizziamo all’aggiunta di una dieta per assicurarci che i valori nutrizionali rispettino i vincoli scelti dal cliente, così si riduce la possibilità di errore; la composizione rappresenta la quantità presente in un alimento.

Diagramma di Dettaglio: Dominio – Agenda

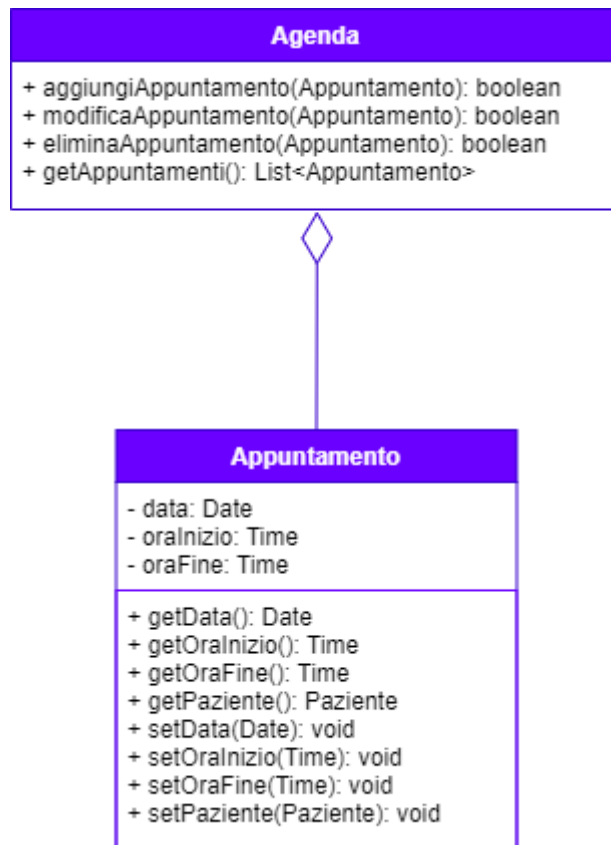


Diagramma di Dettaglio: Dominio – Completo

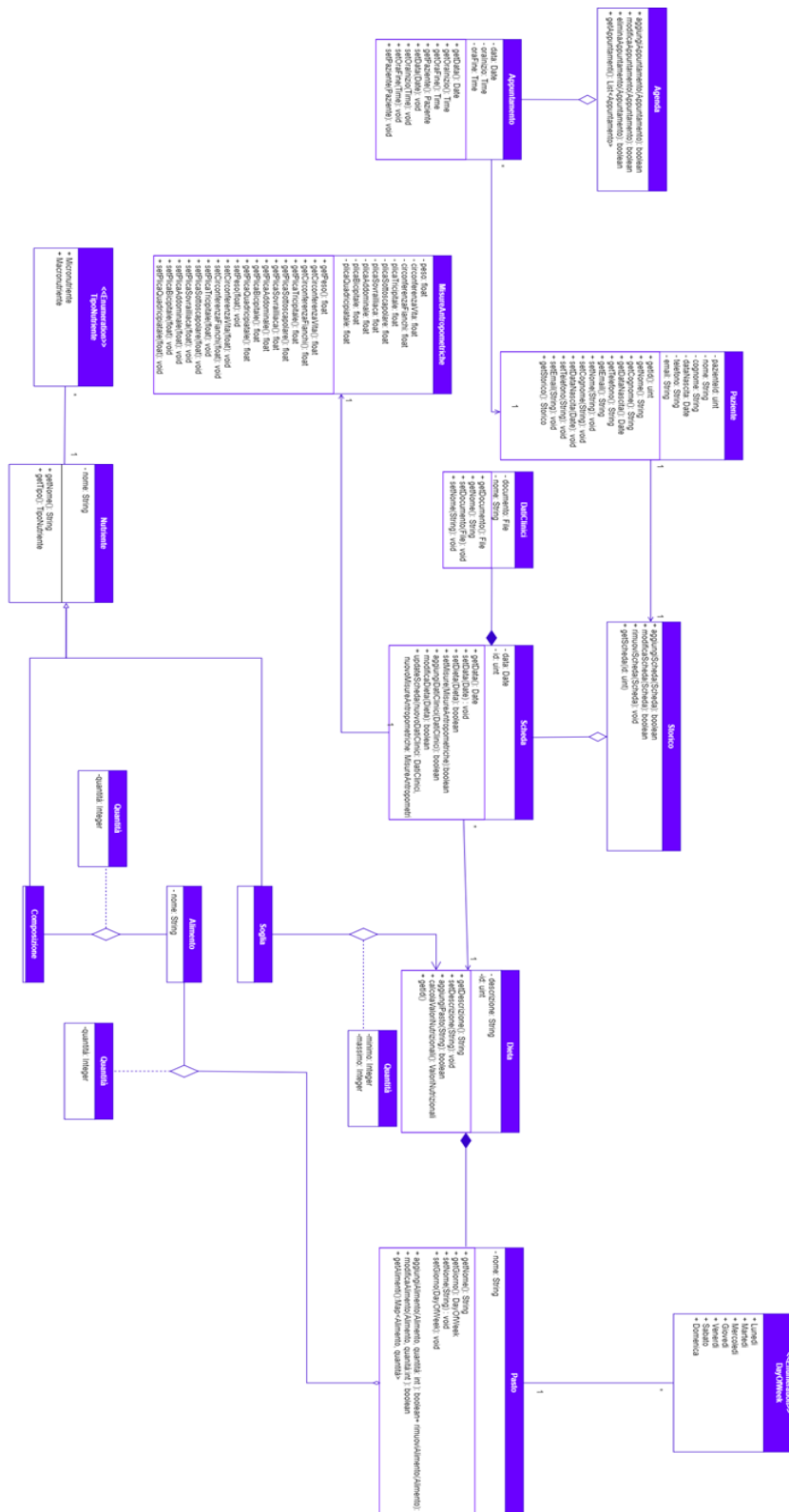


Diagramma di Dettaglio: Interfacce

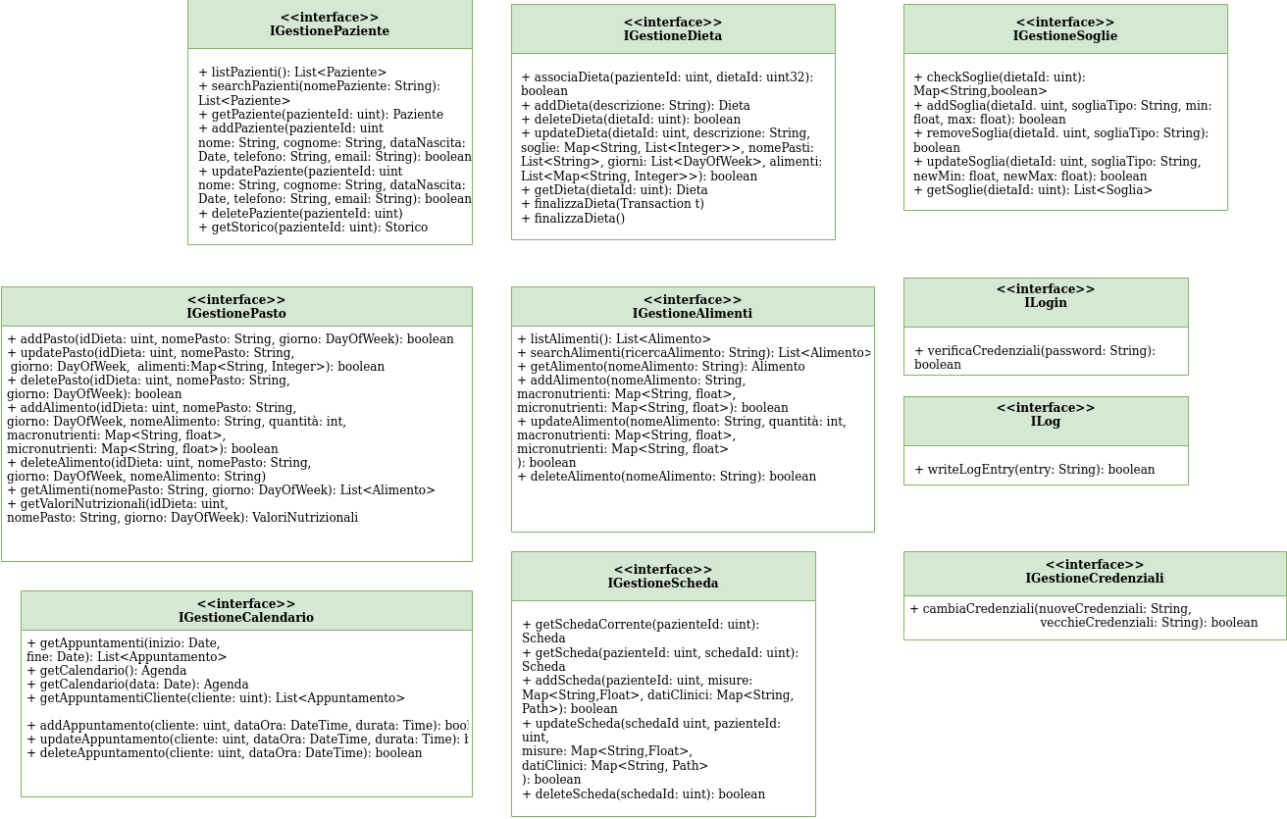


Diagramma di Dettaglio: Calendario Controller

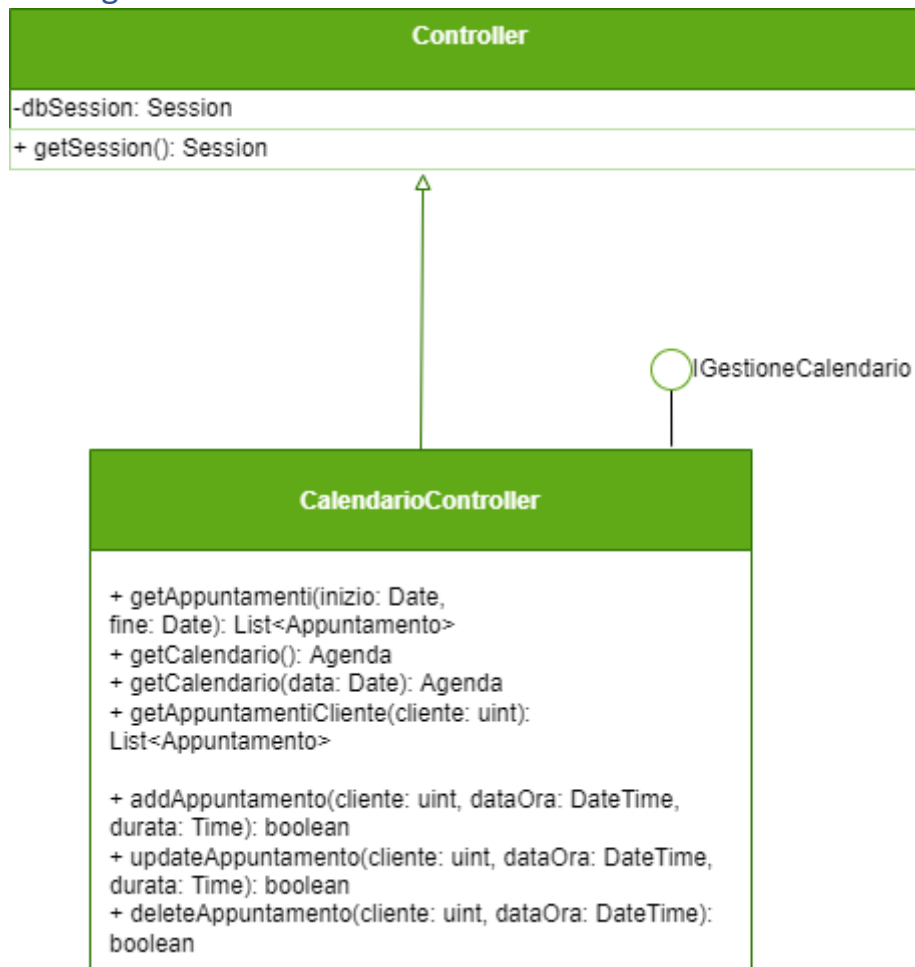


Diagramma di Dettaglio: Credenziali Controller

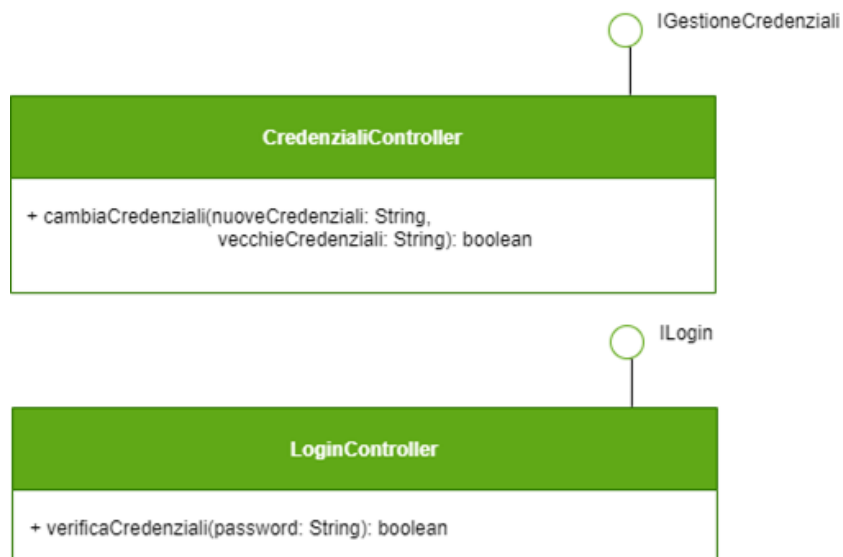


Diagramma di Dettaglio: Dieta Controller

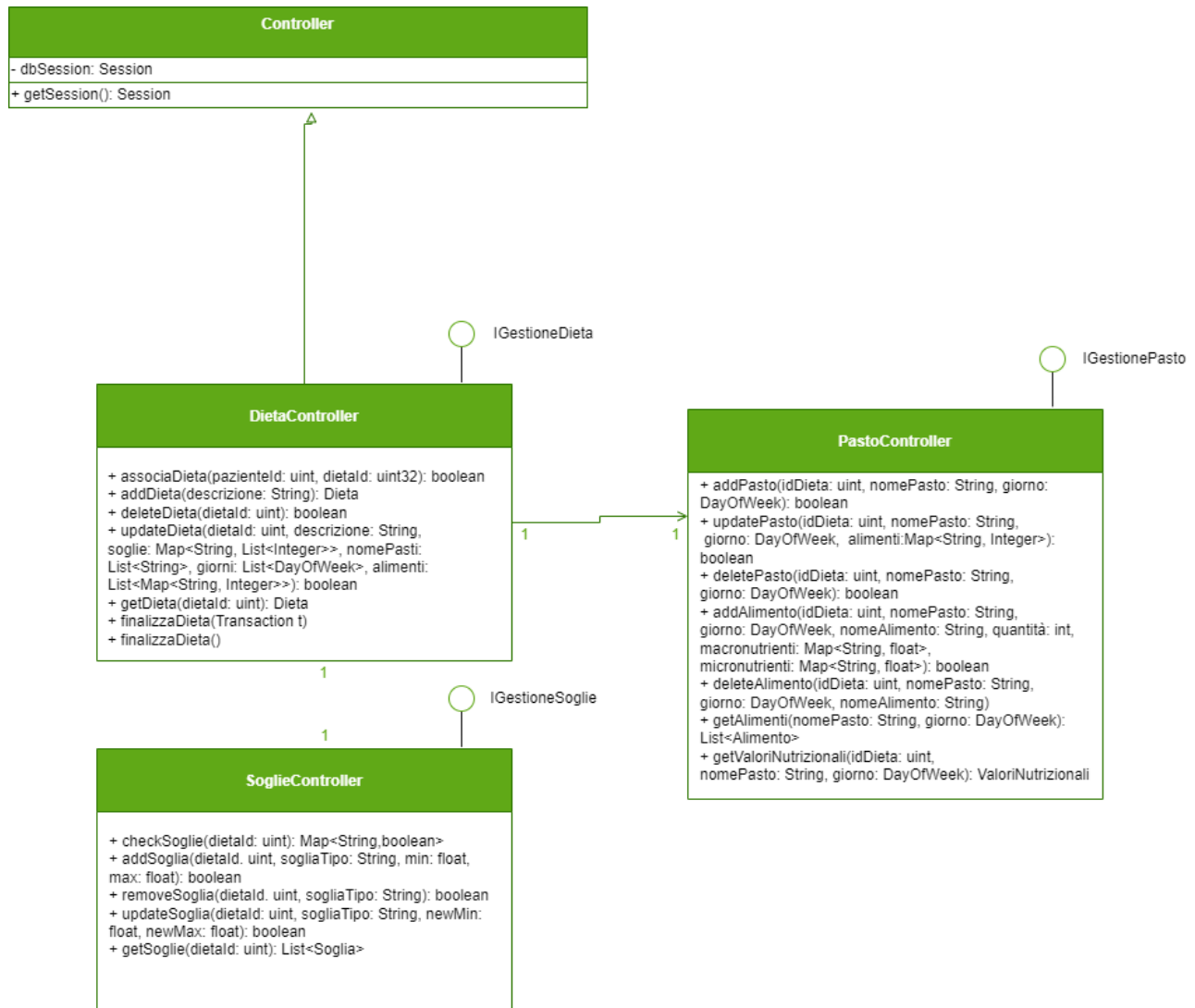


Diagramma di Dettaglio: Paziente Controller

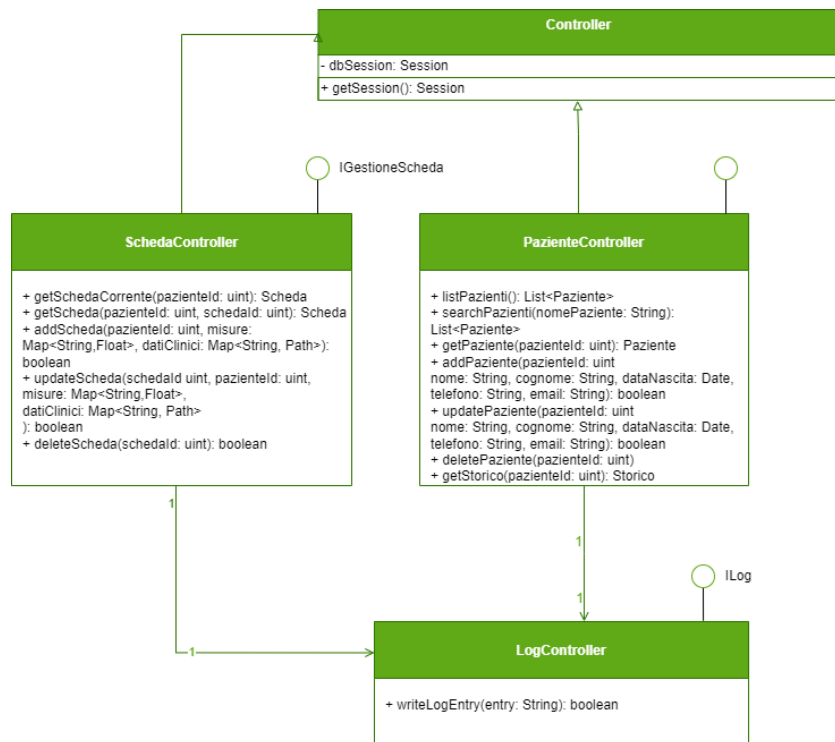


Diagramma di Dettaglio: Controller Completo

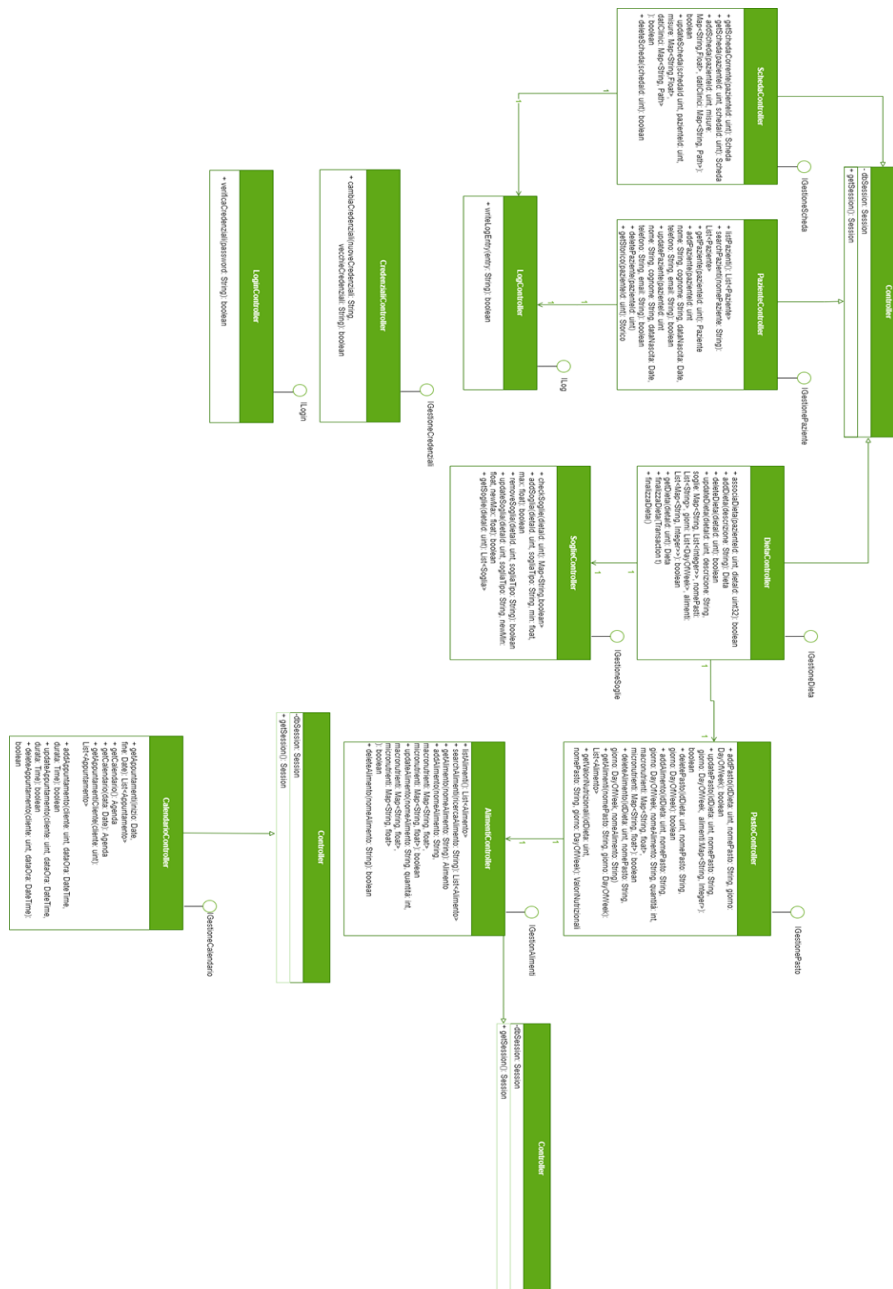
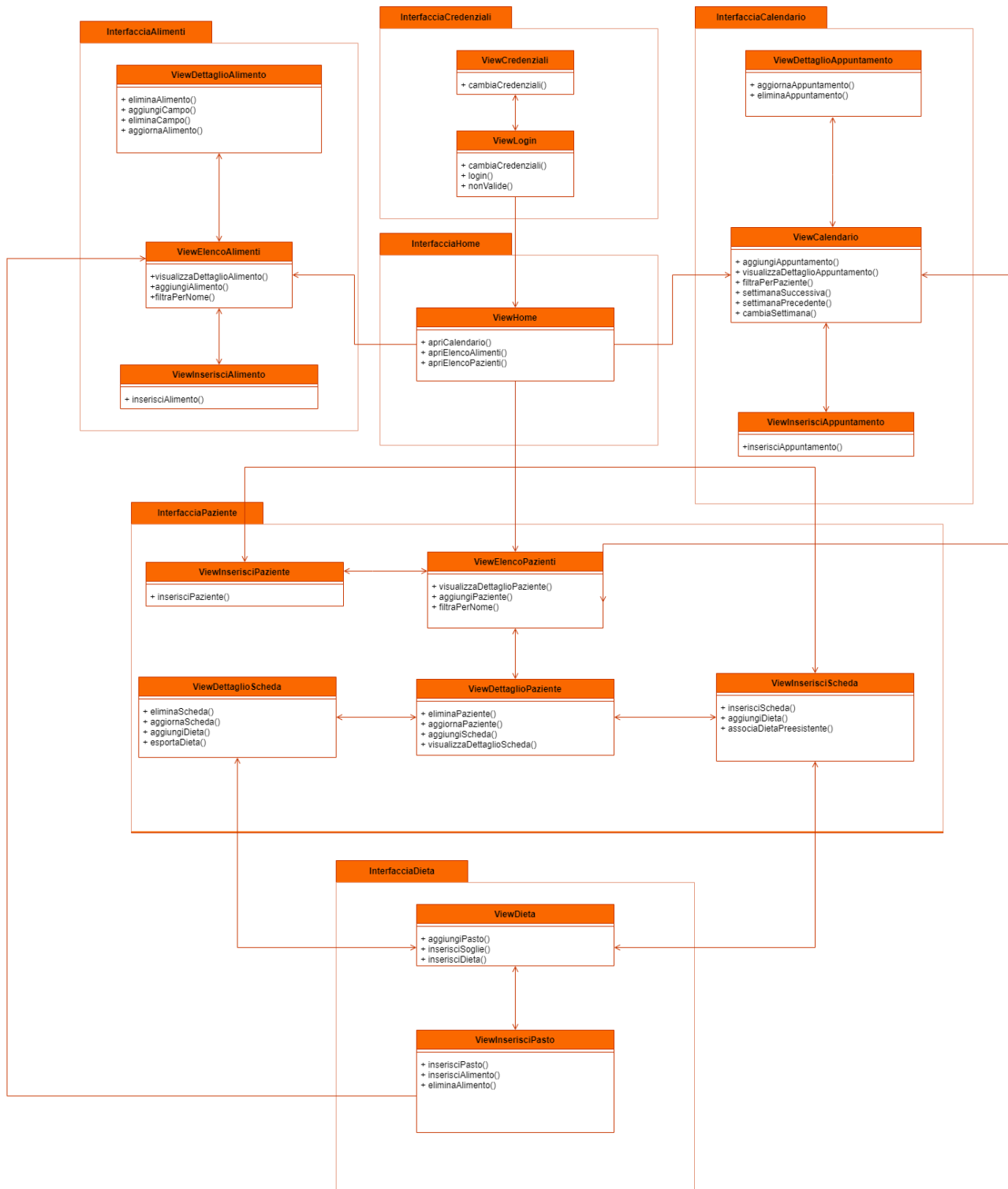
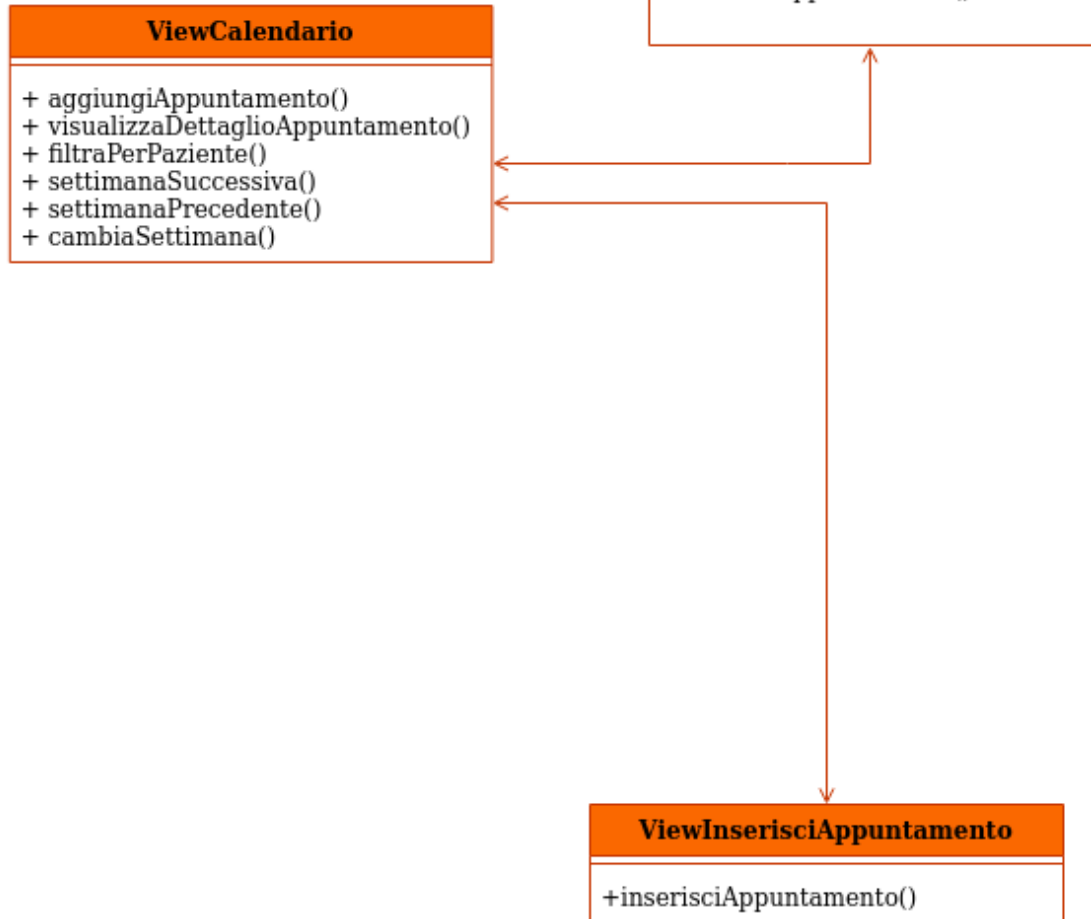


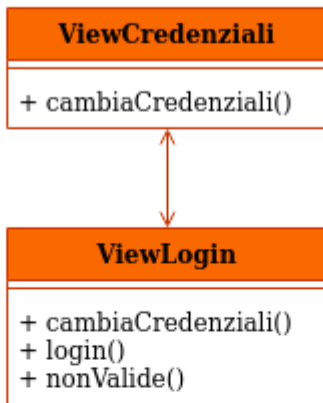
Diagramma di dettaglio: view rese disponibili ai client



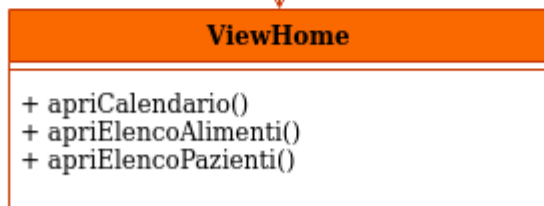
InterfacciaCalendario



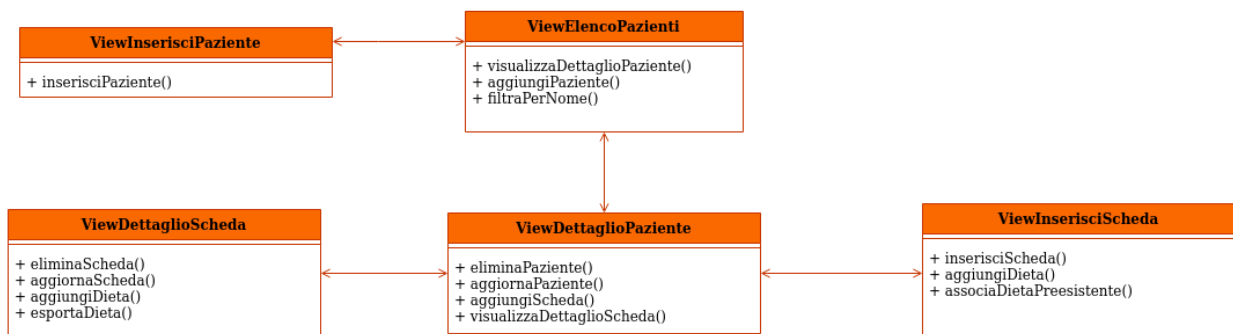
InterfacciaCredenziali

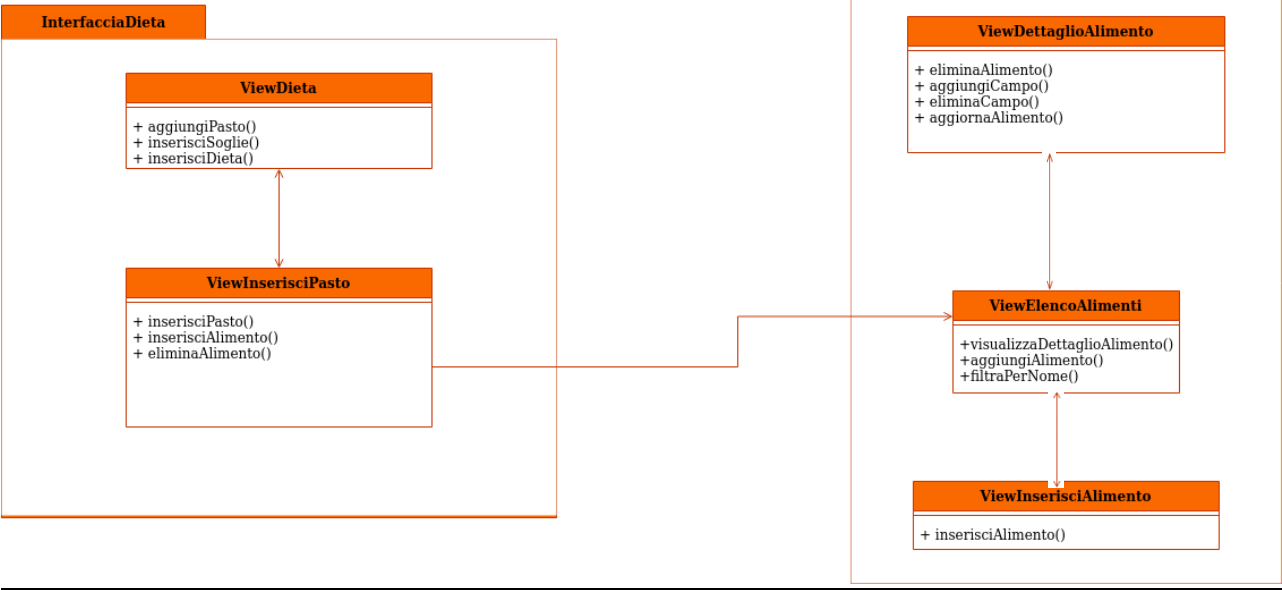


InterfacciaHome



InterfacciaPaziente





Alcune Interfacce grafiche



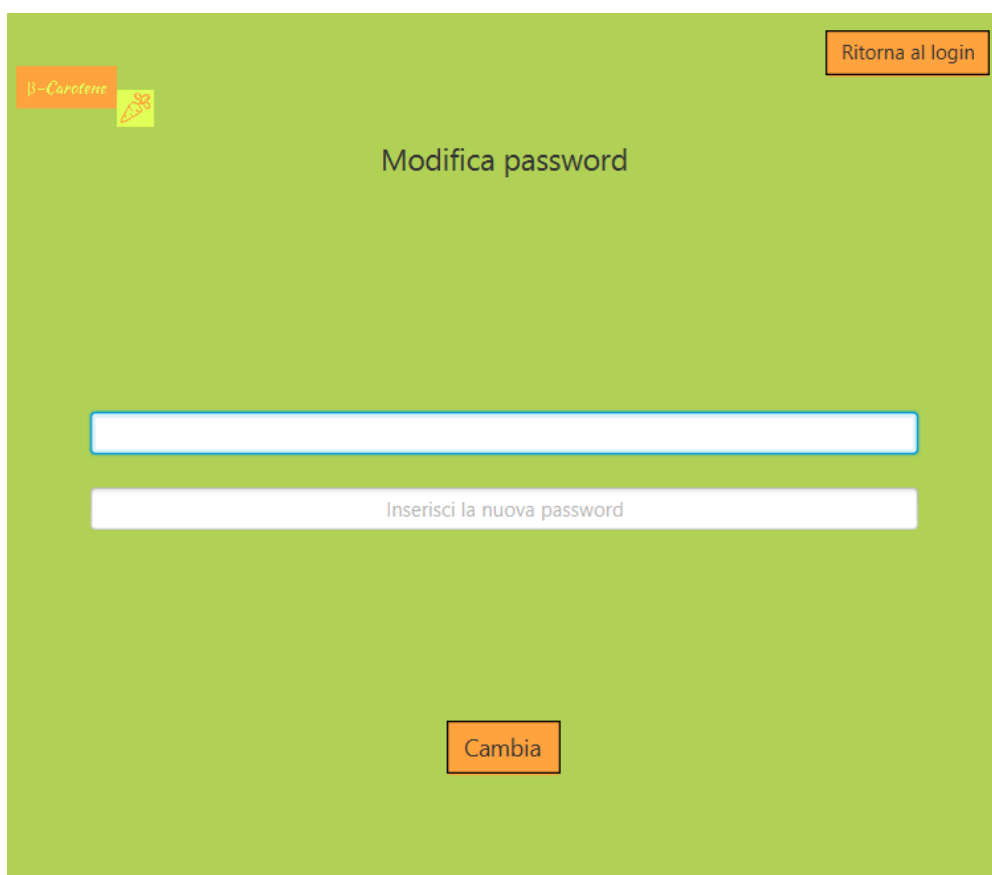
The login interface features a green background. At the top left, there is a logo with the text β -Carotene in yellow on an orange rectangle, followed by a yellow square containing a red outline of a carrot. Below the logo, there is a white rectangular input field with the placeholder text "Inserisci la password". Underneath the input field is an orange rectangular button with the text "Login". At the bottom center, the text "Cambia Password" is displayed in a smaller font.

β -Carotene

Inserisci la password

Login

Cambia Password



The password modification interface has a green background. In the top left corner, there is a small version of the β -Carotene logo. In the top right corner, there is an orange rectangular button with the text "Ritorna al login". The main heading "Modifica password" is centered. Below the heading, there is a white rectangular input field with a blue border. Underneath this field is another white rectangular input field with the placeholder text "Inserisci la nuova password". At the bottom center, there is an orange rectangular button with the text "Cambia".

β -Carotene

Ritorna al login

Modifica password

Inserisci la nuova password

Cambia

Agenda

Pazienti

Alimenti

Cerca...	Aggiungi paziente	
NOME	COGNOME	DATA DI NASCITA

Nome	<input type="text"/>
Cognome	<input type="text"/>
Data di nascita	<input type="text"/>
Telefono	<input type="text"/>
E-mail	<input type="text"/>

Ok

Cancel

Dettaglio paziente

Nome	Mario
Cognome	Rossi
Data di nascita	01/01/2000
Telefono	3333333333
E-mail	mariorossi@gmail.com

Aggiungi Scheda

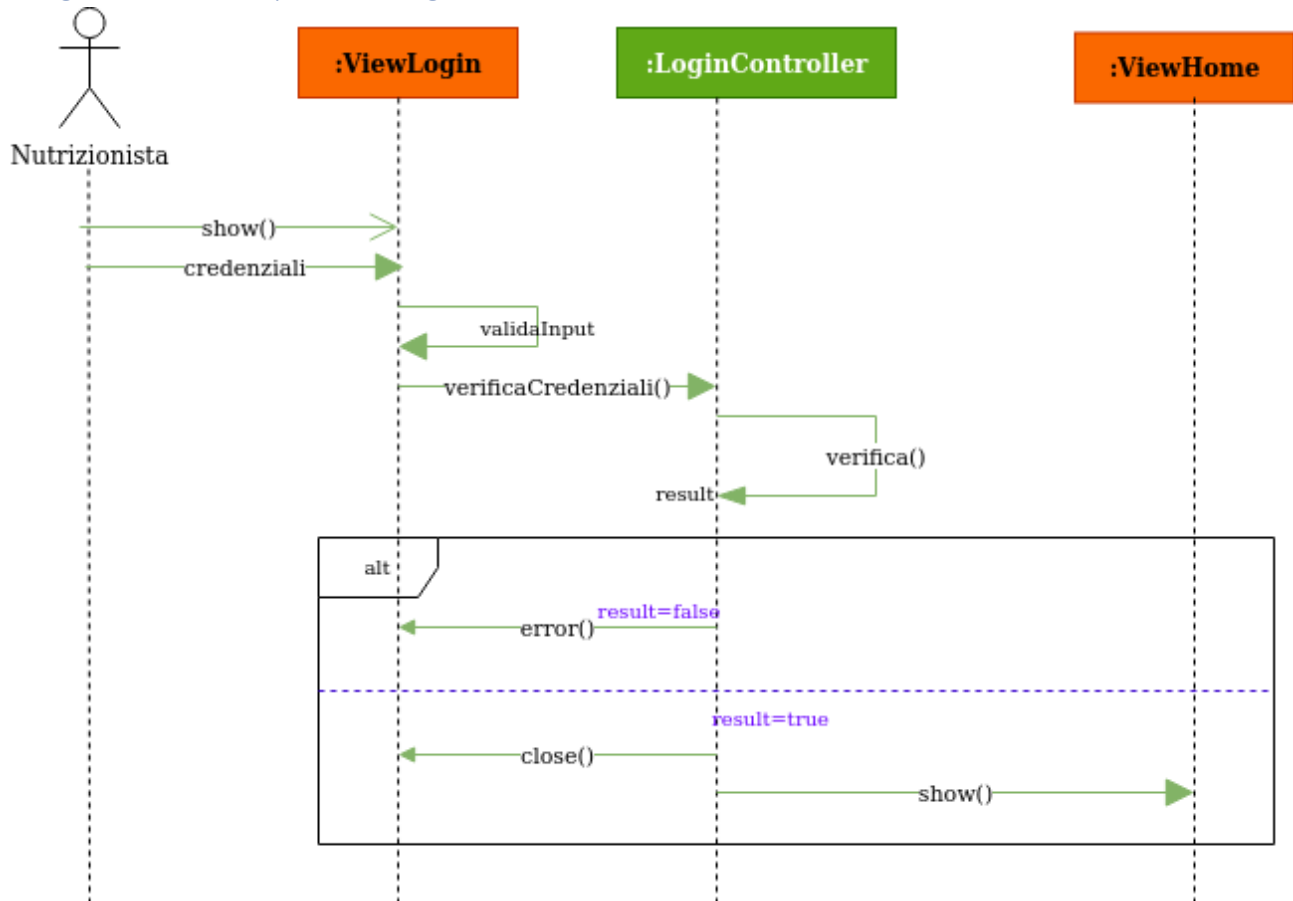
Visualizza Scheda

Modifica

Elimina

Diagrammi di Interazione

Diagramma di Sequenza: Login



L'hash della password viene calcolato con SHA256 e memorizzato su file.

L'applicazione, essendo pensata per un sistema concentrato, verifica la password calcolandone l'hash e confrontandolo col file.

Diagramma di Sequenza: Aggiungi appuntamento

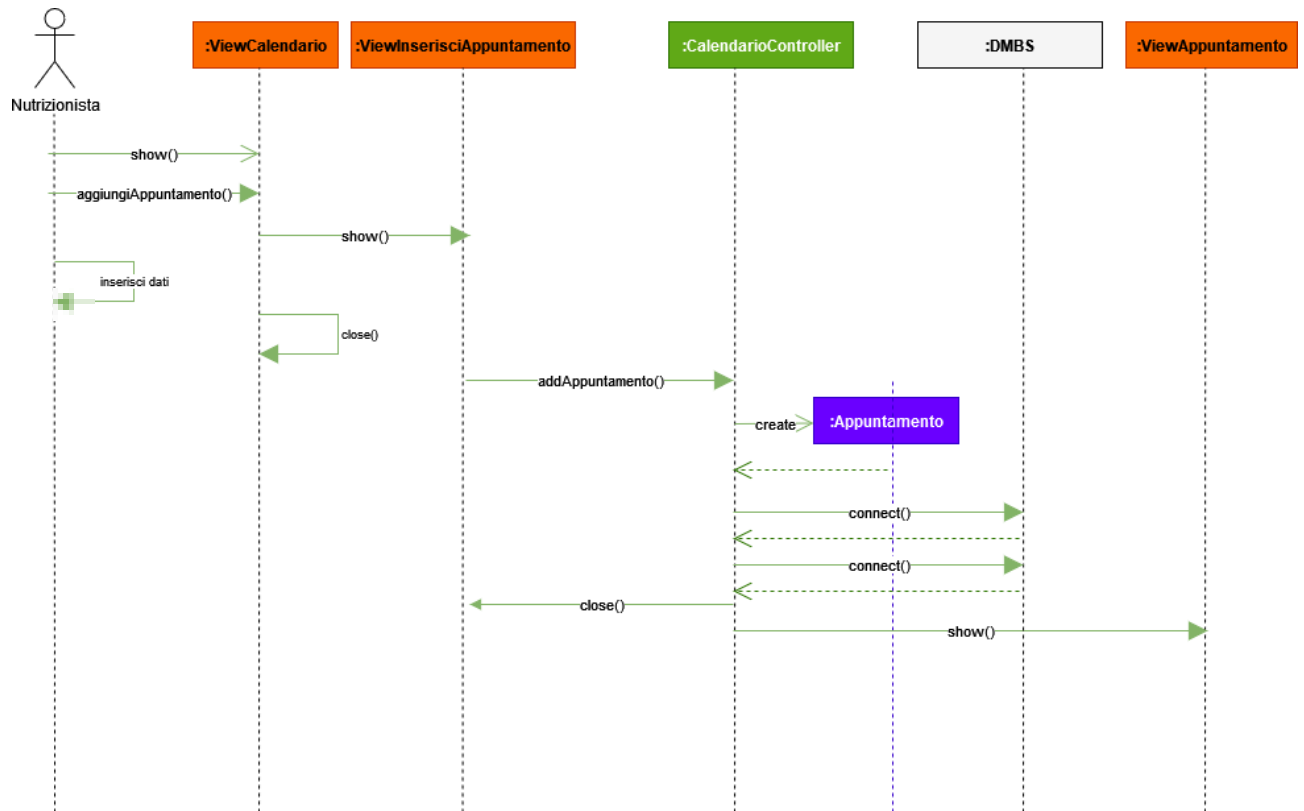


Diagramma di Sequenza: Crea Alimento

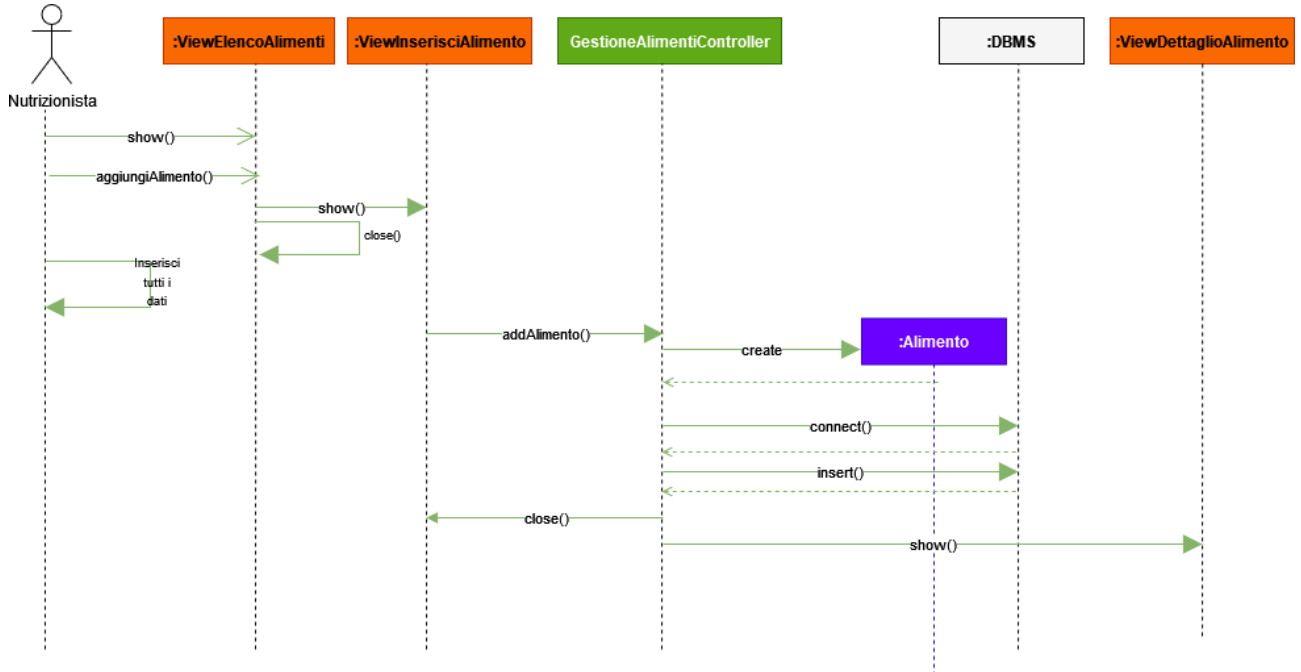


Diagramma di Sequenza: Aggiungi Scheda

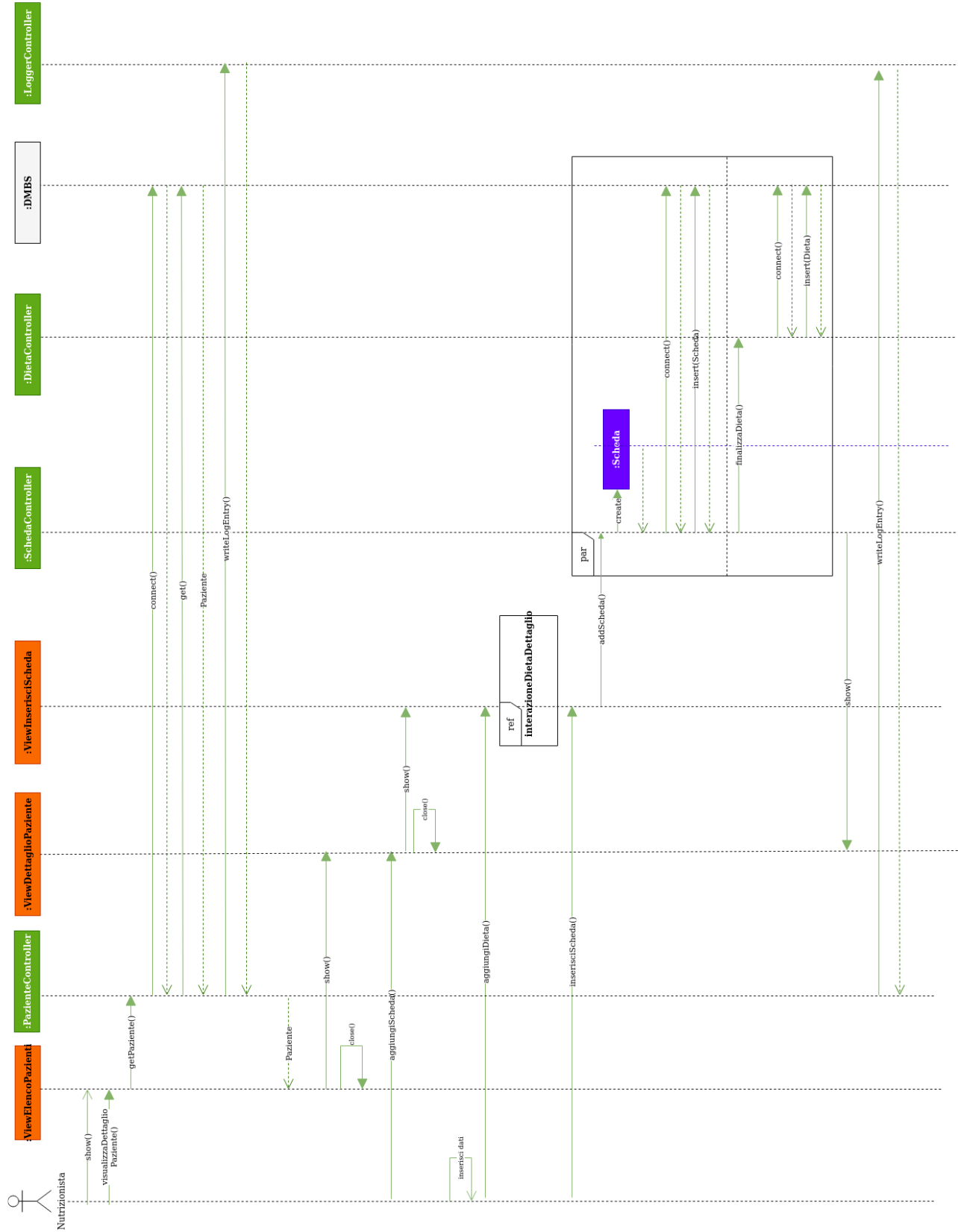


Diagramma di Sequenza: Aggiungi Dieta

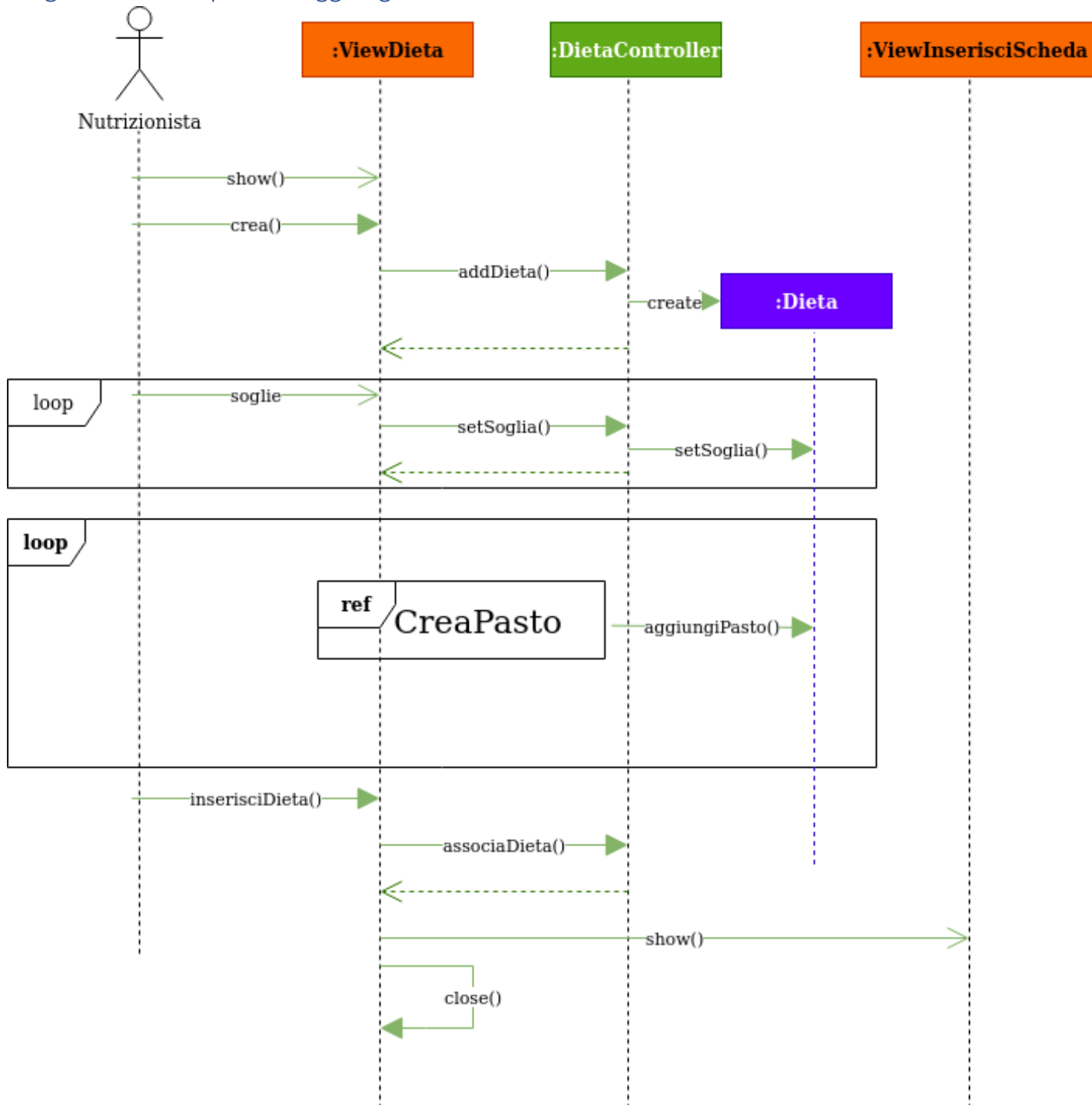
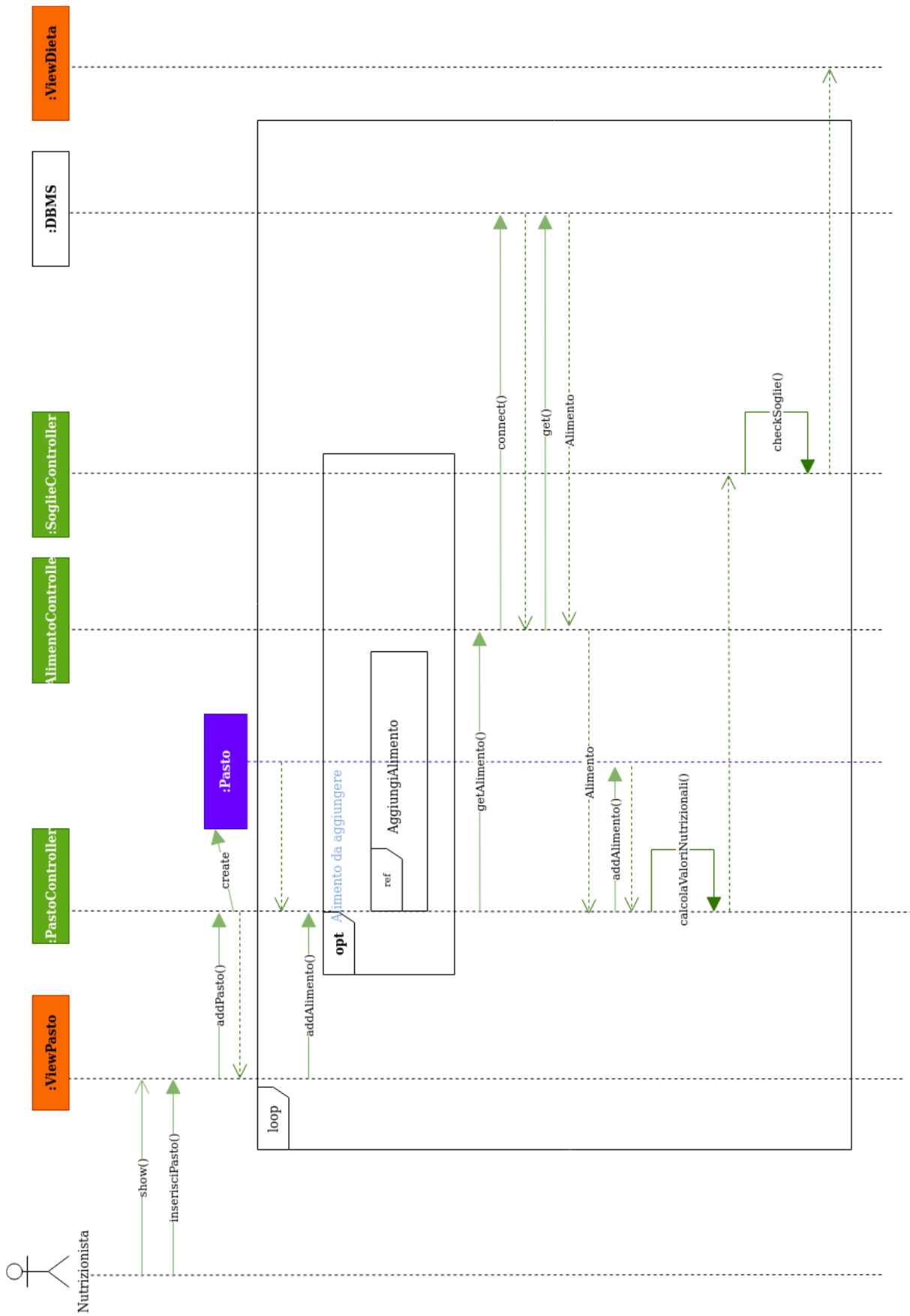
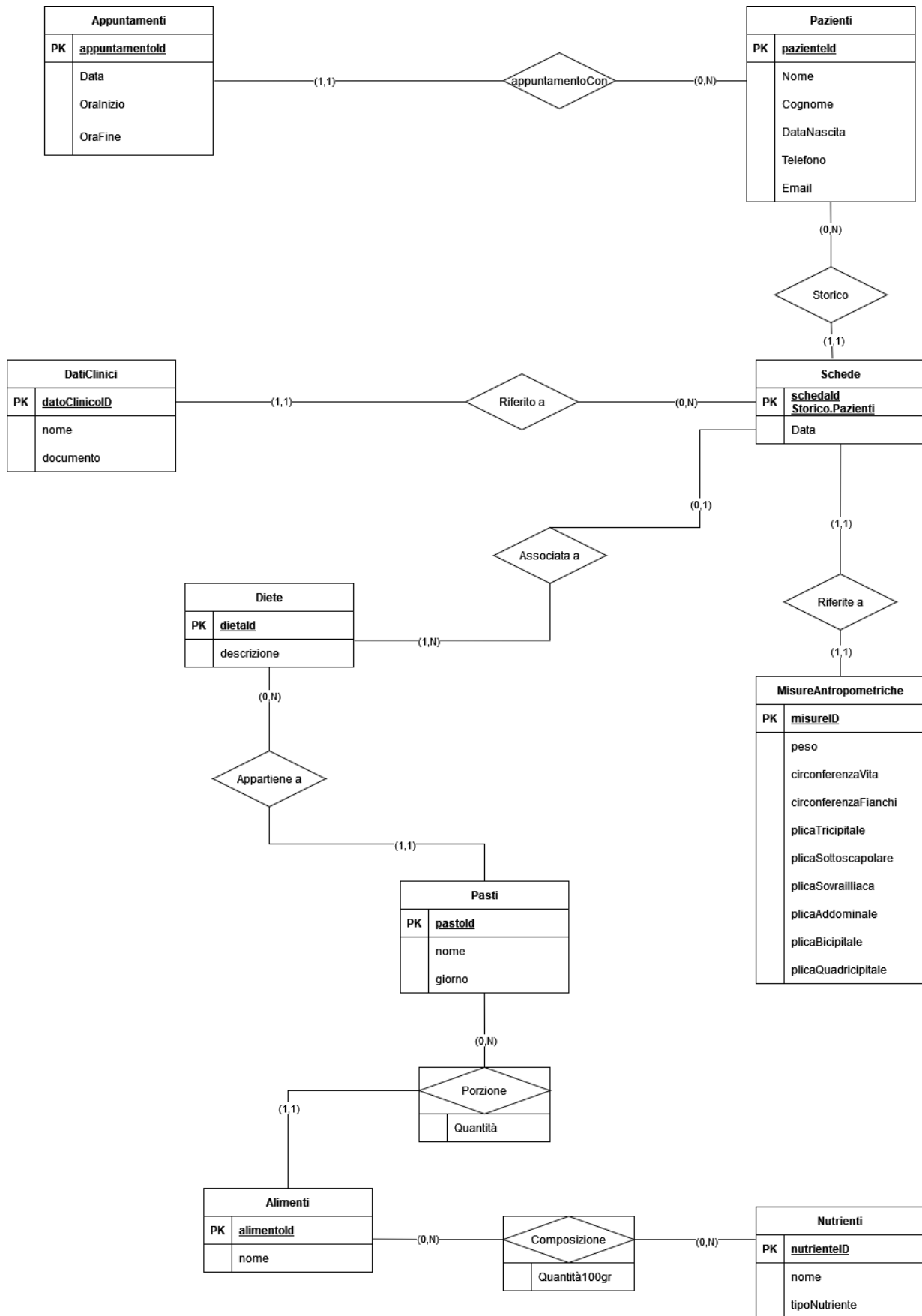


Diagramma di Sequenza: Crea Pasto



Progettazione della Persistenza



Progettazione del Collaudo

```
public class TestAlimenti{

    Paziente paziente;
    Scheda scheda;
    DataBaseMock db;
    @Before
    private void inizializza(){

        db = new DataBaseMock();
        Nutriente proteine = new Nutriente(nome: "Proteine", TipoNutriente.MACRONUTRIENTI);
        Nutriente lipidi = new Nutriente(nome: "lipidi", TipoNutriente.MACRONUTRIENTI);
        Nutriente glucidi = new Nutriente(nome: "glucidi", TipoNutriente.MACRONUTRIENTI);

        Nutriente vitaminaB1 = new Nutriente(nome: "vitaminaB1", TipoNutriente.MICRONUTRIENTI);
        Nutriente vitaminaC = new Nutriente(nome: "vitaminaC", TipoNutriente.MICRONUTRIENTI);
        Nutriente ferro = new Nutriente(nome: "ferro", TipoNutriente.MICRONUTRIENTI);
        Nutriente calcio = new Nutriente(nome: "calcio", TipoNutriente.MICRONUTRIENTI);
        Nutriente zinco = new Nutriente(nome: "zinco", TipoNutriente.MICRONUTRIENTI);
        Nutriente colesterolo = new Nutriente(nome: "colesterolo", TipoNutriente.MICRONUTRIENTI);
        Nutriente betacarotene = new Nutriente(nome: "betacarotene", TipoNutriente.MICRONUTRIENTI);

        String nome = "PastaConTonnoh";

        HashMap<Nutriente, Float> valoriNutrizionali = new HashMap<Nutriente, Float>();
        valoriNutrizionali.put(vitaminaC, value: 1000f);
        valoriNutrizionali.put(ferro, value: 1000f);
        valoriNutrizionali.put(calcio, value: 1000f);
        valoriNutrizionali.put(zinco, value: 1000f);
        valoriNutrizionali.put(colesterolo, value: 1000f);
        valoriNutrizionali.put(betacarotene, value: 1000f);
        valoriNutrizionali.put(glucidi, value: 1000f);
        valoriNutrizionali.put(vitaminaB1, value: 1000f);
        valoriNutrizionali.put(lipidi, value: 1000f);
        valoriNutrizionali.put(proteine, value: 1000f);

        Alimento alimento1 = new Alimento(nome, valoriNutrizionali);

        String nomeZuppetta = "Zuppetta";

        HashMap<Nutriente, Float> valoriNutrizionaliZuppetta = new HashMap<Nutriente, Float>();

        valoriNutrizionaliZuppetta.put(colesterolo, value: 1000000000f);
        valoriNutrizionaliZuppetta.put(lipidi, value: 1000000000f);
        valoriNutrizionaliZuppetta.put(betacarotene, value: 1000000000f);

        Alimento alimento2 = new Alimento(nomeZuppetta, valoriNutrizionaliZuppetta);

        db.addAlimento(alimento1);
        db.addAlimento(alimento2);
    }
}
```

@Test

private void testGettersPaziente()

```
    AlimentiController controller = new AlimentiController();
    Nutriente proteine = new Nutriente(nome: "Proteine", TipoNutriente.MACRONUTRIENTI);
    Nutriente lipidi = new Nutriente(nome: "lipidi", TipoNutriente.MACRONUTRIENTI);
    Nutriente glucidi = new Nutriente(nome: "glucidi", TipoNutriente.MACRONUTRIENTI);
    Nutriente vitaminaB1 = new Nutriente(nome: "vitaminaB1", TipoNutriente.MICRONUTRIENTI);
    Nutriente vitaminaC = new Nutriente(nome: "vitaminaC", TipoNutriente.MICRONUTRIENTI);
    Nutriente ferro = new Nutriente(nome: "ferro", TipoNutriente.MICRONUTRIENTI);
    Nutriente calcio = new Nutriente(nome: "calcio", TipoNutriente.MICRONUTRIENTI);
    Nutriente zinco = new Nutriente(nome: "zinco", TipoNutriente.MICRONUTRIENTI);
    Nutriente colesterolo = new Nutriente(nome: "colesterolo", TipoNutriente.MICRONUTRIENTI);
    Nutriente betacarotene = new Nutriente(nome: "betacarotene", TipoNutriente.MICRONUTRIENTI);
    HashMap<Nutriente, Float> valoriNutrizionali = new HashMap<Nutriente, Float>();
    valoriNutrizionali.put(vitaminaC, value: 1000f);
    valoriNutrizionali.put(ferro, value: 1000f);
    valoriNutrizionali.put(calcio, value: 1000f);
    valoriNutrizionali.put(zinco, value: 1000f);
    valoriNutrizionali.put(colesterolo, value: 1000f);
    valoriNutrizionali.put(betacarotene, value: 1000f);
    valoriNutrizionali.put(glucidi, value: 1000f);
    valoriNutrizionali.put(vitaminaB1, value: 1000f);
    valoriNutrizionali.put(lipidi, value: 1000f);
    valoriNutrizionali.put(proteine, value: 1000f);

    Alimento alimento1 = controller.getAlimento(nomeAlimento: "PastaConTonno");
    assertEquals("PastaConTonno", alimento1.getNome());
    assertEquals(valoriNutrizionali, alimento1.getValoriNutrizionali());
```

}

@Test

private void testAddAlimento()

```
    AlimentiController controller = new AlimentiController();

    Nutriente lipidi = new Nutriente(nome: "lipidi", TipoNutriente.MACRONUTRIENTI);
    Nutriente glucidi = new Nutriente(nome: "glucidi", TipoNutriente.MACRONUTRIENTI);

    Nutriente ferro = new Nutriente(nome: "ferro", TipoNutriente.MICRONUTRIENTI);
    Nutriente calcio = new Nutriente(nome: "calcio", TipoNutriente.MICRONUTRIENTI);
    Nutriente colesterolo = new Nutriente(nome: "colesterolo", TipoNutriente.MICRONUTRIENTI);
    Nutriente betacarotene = new Nutriente(nome: "betacarotene", TipoNutriente.MICRONUTRIENTI);
    String nome = "CrudoCottoSalsiccia";
    HashMap<Nutriente, Float> valoriNutrizionali = new HashMap<Nutriente, Float>();
    valoriNutrizionali.put(ferro, value: 5000f);
    valoriNutrizionali.put(calcio, value: 100000f);
    valoriNutrizionali.put(colesterolo, value: 1f);
    valoriNutrizionali.put(betacarotene, value: 2000f);
    valoriNutrizionali.put(glucidi, value: 4000f);
    valoriNutrizionali.put(lipidi, value: 3000f);

    HashMap<String, Float> micronutrienti = new HashMap<String, Float>();
    HashMap<String, Float> macronutrienti = new HashMap<String, Float>();

    micronutrienti.put(key: "lipidi", value: 3000f);
    micronutrienti.put(key: "glucidi", value: 4000f);
    macronutrienti.put(key: "ferro", value: 5000f);
    macronutrienti.put(key: "calcio", value: 100000f);
    macronutrienti.put(key: "colesterolo", value: 1f);
    macronutrienti.put(key: "betacarotene", value: 2000f);
    Alimento alimento = new Alimento(nome, valoriNutrizionali);

    controller.addAlimento(nome, macronutrienti, micronutrienti);

    Alimento lastAlimento = controller.getAlimento(nome);

    assertEquals(alimento.getNome(), lastAlimento.getNome());
    assertEquals(alimento.getValoriNutrizionali(), lastAlimento.getValoriNutrizionali());
}
```



```

@Test
public void testAggiunta(){
    int pazienteId = 2;
    String nome = "Giulio";
    String cognome = "Giuliani";
    LocalDate dataNascita=LocalDate.of(year: 1969, month: 4, dayOfMonth: 20);
    String telefono = "222 22 22 222";
    String mail = "giulio.giuliani@yahoo.com";
    List<Scheda> storico = new ArrayList<Scheda>();
    Paziente giulio = new Paziente(pazienteId, nome, cognome, dataNascita, telefono, mail, storico);
    PazientiController controller= new PazientiController();
    controller.addPaziente(pazienteId, nome, cognome, dataNascita, telefono, mail, storico);
    List<Paziente> pazienti=controller.searchPazienti(nome);

    boolean found = false;
    for(Paziente paz : pazienti){
        if(paz.equals(giulio)){
            found = true;
        }
    }
    assertEquals(found, true);
}

@Test
public void testModifica(){
    PazientiController controller= new PazientiController();
    String nome = "Giulio";
    String cognome = "Giuliano";
    LocalDate dataNascita=LocalDate.of(year: 1969, month: 4, dayOfMonth: 20);
    String telefono = "222 22 22 456";
    String mail = "giulio.giuliano@yahoo.com";
    List<Scheda> storico = new ArrayList<Scheda>();
    int giulioId = controller.getPaziente(nome, cognome, dataNascita).getPazienteId();
    Paziente newGiulio = new Paziente(giulioId, nome, cognome, dataNascita, telefono, mail, storico);
    controller.updatePaziente(giulioId, nome, cognome, dataNascita, telefono, mail, storico);
    List<Paziente> pazienti=controller.searchPazienti(nome);

    boolean found = false;
    for(Paziente paz : pazienti){
        if(paz.equals(newGiulio)){
            found = true;
        }
    }
    assertEquals(found, true);
}

```

```

public class TestSchedaDb {

    DataBaseMock db;

    @Before
    public void inizializza(){
        db=new DataBaseMock();
        int pazienteId = 1;
        String nome= "Luigi";
        String cognome= "Mario";
        LocalDate dataNascita=LocalDate.of(year: 2001, month: 9, dayOfMonth: 11);
        String telefono="333 33 33 333";
        String mail= "mario.mario@gmail.com";
        List<Scheda> storico = new ArrayList<Scheda>();
        float peso=70;
        float circonferenzaVita=102;
        float circonferenzaFianchi=105;
        float plicaTricipitale=4;
        float plicaSottoscapolare=6;
        float plicaSovrailliaca=5;
        float plicaAddominale=9;
        float plicaBicipitale=7;
        float plicaQuadricipitale=8;
        Scheda scheda1=new Scheda(schedaId: 0, peso, circonferenzaVita, circonferenzaFianchi,
        plicaTricipitale, plicaSottoscapolare,
        plicaSovrailliaca, plicaAddominale, plicaBicipitale,
        plicaQuadricipitale, new HashMap<String, Path>(), new Dieta());

        peso=65;
        circonferenzaVita=98;
        circonferenzaFianchi=100;
        plicaTricipitale=3;
        plicaSottoscapolare=4;
        plicaSovrailliaca=6;
        plicaAddominale=6;
        plicaBicipitale=2;
        plicaQuadricipitale=4;
        Scheda scheda2=new Scheda(schedaId: 1, peso, circonferenzaVita, circonferenzaFianchi, plicaTricipitale,
        plicaSovrailliaca, plicaAddominale, plicaBicipitale, plicaQuadricipitale,
        new HashMap<String, Path>(), new Dieta());

        storico.add(scheda1);
        storico.add(scheda2);
        Paziente paziente=new Paziente(pazienteId, nome, cognome, dataNascita, telefono, mail, storico);
        db.addPaziente(paziente);
    }
}

```



```

@Test
public void testModifica(){
    int pazienteId = 1;
    SchedaController controller= new SchedaController();
    float peso=70;
    float circonferenzaVita=102;
    float circonferenzaFianchi=105;
    float plicaTricipitale=4;
    float plicaSottoscapolare=6;
    float plicaSovrailliaca=5;
    float plicaAddominale=9;
    float plicaBicipitale=7;
    float plicaQuadricipitale=8;
    HashMap<String, Float> misure = new HashMap<String, Float>();
    misure.put(key: "circonferenzaVita", circonferenzaVita);
    misure.put(key: "circonferenzaFianchi", circonferenzaFianchi);
    misure.put(key: "plicaTricipitale", plicaTricipitale);
    misure.put(key: "peso", peso);
    misure.put(key: "plicaSottoscapolare", plicaSottoscapolare);
    misure.put(key: "plicaSovrailliaca", plicaSovrailliaca);
    misure.put(key: "plicaAddominale", plicaAddominale);
    misure.put(key: "plicaBicipitale", plicaBicipitale);
    misure.put(key: "plicaQuadricipitale", plicaQuadricipitale);

    HashMap<String, Path> datiClinici = new HashMap<String, Path>();
    datiClinici.put(key: "analisi",Path.of(first: "notdummy.txt"));

    Dieta dieta = new Dieta();

    Scheda schedaNew=new Scheda(controller.getSchedaCorrente(pazienteId).getSchedaId(), peso, circonferenzaVita, circonferenzaFianchi, plicaTricipitale, plicaSottoscapolare,
    plicaSovrailliaca, plicaAddominale, plicaBicipitale, plicaQuadricipitale, datiClinici, dieta);
    controller.updateScheda(controller.getSchedaCorrente(pazienteId).getSchedaId(), pazienteId, misure, datiClinici, new Dieta());

    Scheda lastScheda = controller.getSchedaCorrente(pazienteId);
    assertEquals(controller.getSchedaCorrente(pazienteId).schedaNew.getSchedaId(),
    lastScheda.getSchedaId());
    assertEquals(misure, lastScheda.getMisure());
    assertEquals(datiClinici, lastScheda.getDatiClinici());
    assertEquals(dieta, lastScheda.getDieta());
}
}

```

```

@Test
public void testGetSchedaCorrente(){
    int pazienteId = 1;
    SchedaController controllerScheda = new SchedaController();
    Scheda scheda=controllerScheda.getSchedaCorrente(pazienteId);
    assertEquals(scheda.getSchedaId(), 1);
}

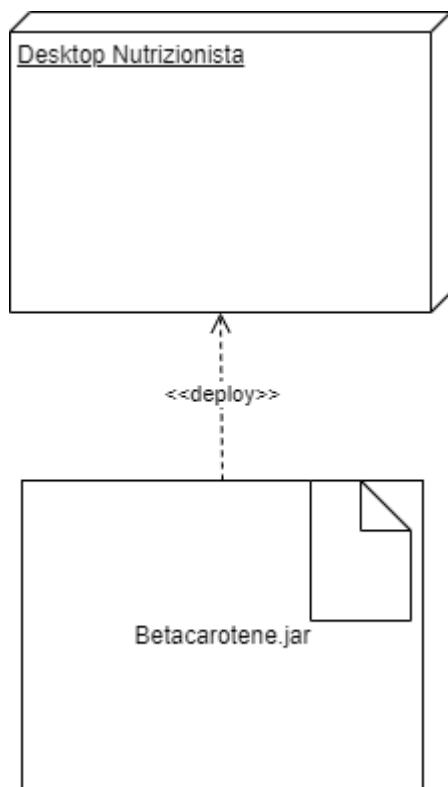
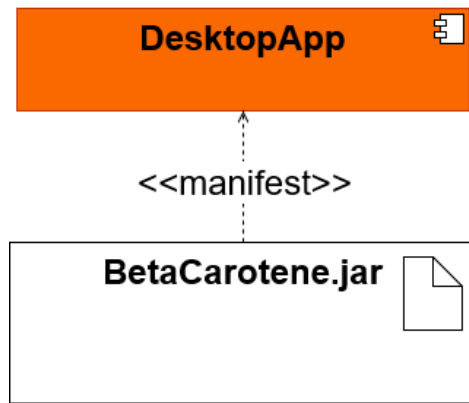
@Test
public void testAggiunta(){
    int pazienteId = 1;
    float peso=70;
    float circonferenzaVita=102;
    float circonferenzaFianchi=105;
    float plicaTricipitale=4;
    float plicaSottoscapolare=6;
    float plicaSovrailliaca=5;
    float plicaAddominale=9;
    float plicaBicipitale=7;
    float plicaQuadricipitale=8;

    HashMap<String, Float> misure = new HashMap<String, Float>();
    misure.put(key: "circonferenzaVita", circonferenzaVita);
    misure.put(key: "circonferenzaFianchi", circonferenzaFianchi);
    misure.put(key: "plicaTricipitale", plicaTricipitale);
    misure.put(key: "peso", peso);
    misure.put(key: "plicaSottoscapolare", plicaSottoscapolare);
    misure.put(key: "plicaSovrailliaca", plicaSovrailliaca);
    misure.put(key: "plicaAddominale", plicaAddominale);
    misure.put(key: "plicaBicipitale", plicaBicipitale);
    misure.put(key: "plicaQuadricipitale", plicaQuadricipitale);
    HashMap<String, Path> datiClinici = new HashMap<String, Path>();
    datiClinici.put(key: "analisiDelSangue",Path.of(first: "dummy.txt"));
    Dieta dieta = new Dieta();
    SchedaController controller= new SchedaController();
    controller.addScheda(pazienteId, misure, datiClinici, dieta);
    Scheda lastScheda=controller.getSchedaCorrente(pazienteId);

    assertEquals(misure, lastScheda.getMisure());
    assertEquals(datiClinici, lastScheda.getDatiClinici());
    assertEquals(dieta, lastScheda.getDieta());
}
}

```

Diagramma di Deployment



Non segnaliamo il nodo del DBMS perché intendiamo installarlo in locale sulla macchina del cliente.