



UNIVERSITA' POLITECNICA DELLE MARCHE

*Facoltà di Ingegneria*

**PROGETTO:**

**REALIZZAZIONE DI UN SOFTWARE GESTIONALE  
PER UNA AUTOSCUOLA**

*Relazione di:*

*De Ritis Riccardo  
Giacconi Alessio  
D'Ambrosio Luca  
Dellisanti Francesco*

*Esame di ingegneria del software*

*Corso di Laurea Triennale Ingegneria Informatica e dell'Automazione*

# Sommario



L'obiettivo di questa relazione è la progettazione di un software gestionale per una ipotetica autoscuola del nostro territorio, “Autoscuola Sprint”, una realtà aziendale modesta ma che necessita anch’essa di modernizzare e rendere più efficienti le sue operazioni quotidiane. Il nostro compito è quello di progettare, documentare e implementare un prodotto software che possa essere un utile strumento di supporto per le attività dei dipendenti dell’“Autoscuola Sprint”, a partire dai segretari fino agli istruttori e passando per i medici oculisti che collaborano con l’autoscuola. Il lavoro richiesto è quindi quello di pianificare ed eseguire un vero e proprio processo software, con tutte le attività ad esso connesse: specifiche del software, il suo sviluppo e parte della convalida del prodotto finale. Non potendo ovviamente confrontarci con un cliente reale la convalida del software è svolta tramite una serie di test volti a determinare il corretto funzionamento del programma. Tutta la documentazione relativa alla parte di progettazione del software che abbiamo provveduto a redigere è un utile strumento sia per il cliente, che potrà verificare da diversi punti di vista se il software sviluppato soddisfa le sue necessità e le sue aspettative, sia per eventuali team di sviluppo che si dovranno occupare in futuro dell’evoluzione del software, nel caso in cui il cliente richieda l’aggiunta di nuovi requisiti oppure di interventi di manutenzione o refactoring.

Per quanto riguarda la fase di implementazione del software, il programma è stato scritto in linguaggio Python 3 tramite l’IDE PyCharm, ambiente di sviluppo integrato che offre utili funzionalità per l’indentazione del codice, il debugging e un rapido collegamento con GitHub, in modo da supportare la condivisione e lo sviluppo del codice tra i vari membri del team di sviluppo.

<b>1. INTERVISTA</b>	<b>3</b>
<b>2. DESCRIZIONE DEL SISTEMA IN LINGUAGGIO NATURALE</b>	<b>8</b>
<b>3. DIAGRAMMA DEI SISTEMI</b>	<b>9</b>
<b>4. GLOSSARIO DEI TERMINI</b>	<b>10</b>
<b>5. ANALISI DEI REQUISITI (REQUISITI FUNZIONALI E NON FUNZIONALI)</b>	<b>11</b>
5.1    REQUISITI FUNZIONALI	12
5.2    REQUISITI NON FUNZIONALI	13
<b>6. CASI D'USO: ATTORI</b>	<b>13</b>
<b>7. DESCRIZIONE DEI CASI D'USO</b>	<b>14</b>
7.1    UC: GESTIONE DIPENDENTI	15
7.2    UC: VISUALIZZA DOCUMENTI	17
7.3    UC: GESTIONE PAGAMENTI	20
7.4    UC: GESTIONE VISITE MEDICHE	22
7.5    UC: GESTIONE CLIENTI	24
7.6    UC: GESTIONE ATTIVITÀ	26
<b>8. DIAGRAMMI DEI CASI D'USO</b>	<b>31</b>
<b>9. MATRICE DI MAPPING REQUISITI - CASI D'USO</b>	<b>32</b>
<b>10. MAPPA DELL'ARCHITETTURA</b>	<b>33</b>
<b>11. DIAGRAMMA DELLE CLASSI</b>	<b>34</b>
11.01    DC HOME	35
11.02    DC CLIENTI	36
11.03    DC DIPENDENTI	37
11.04    DC DOCUMENTI	38
11.05    DC RINNOVO	38
11.06    DC PAGAMENTO	39
11.07    DC ESITO ESAME	39
11.08    DC ORARI PRENOTAZIONI	40
11.09    DC DISDETTA PRENOTAZIONE	40
11.10    DC NUOVA PRENOTAZIONE	41
11.11    DC VISITA MEDICA	41
11.12    DC ESITO VISITA MEDICA	42
11.13    DC DISDICI VISITA MEDICA	42
<b>12. DIAGRAMMI DI SEQUENZA ED ATTIVITÀ</b>	<b>43</b>
12.1    UC: GESTIONE DIPENDENTI	43
12.2    UC: VISUALIZZA DOCUMENTI	45
12.3    UC: GESTIONE PAGAMENTI	48
12.4    UC: GESTIONE VISITE MEDICHE	50
12.5    UC: GESTIONE CLIENTI	52
12.6    UC: GESTIONE ATTIVITÀ	54
<b>13. MOCK-UP</b>	<b>57</b>
<b>14. IMPLEMENTAZIONE</b>	<b>68</b>
<b>15. PYUNIT</b>	<b>71</b>

# 1. Intervista

## • **Introduzione del cliente**

“Sono Gianpiero, CEO (Chief Executive Officer) dell’azienda “Autoscuola Sprint”, la nostra autoscuola nasce in un piccolo paesino dell’entroterra, fondata dal padre di mio padre e gestita in ambito familiare da oltre 60 anni. Negli ultimi anni abbiamo avuto l’esigenza di ampliare la nostra azienda e di aprire altre sedi dislocate nel territorio. L’ultima sede inaugurata è stata aperta ad Ancona nell’ultimo anno. L’azienda conta quattro poli distaccati che coprono tutta la provincia.

In tutte le varie sedi si può usufruire di una vasta gamma di servizi: dai corsi di guida per auto e ciclomotori all’assistenza burocratica per il disbrigo di pratiche auto, dalle immatricolazioni di nuovi veicoli ai rinnovi e duplicati di patenti. Il personale dell’”Autoscuola Sprint” possiede una lunga esperienza nel settore: i corsi di guida sono tenuti da docenti e istruttori qualificati, mentre gli adempimenti amministrativi sono affidati a figure specializzate, in grado di gestire le vostre pratiche auto con serietà e competenza.

L’offerta formativa della nostra “Autoscuola Sprint” è variegata, per venire incontro ad ogni esigenza: dai corsi di guida mirati al conseguimento della Patente di Guida di tipo A, B, C, D ed E, ai corsi per il patentino per ciclomotori e per le patenti professionali, che si tengono in diverse fasce orarie.

Siamo alla ricerca di un software per il controllo interno dell’azienda che ci consenta di uniformare quanto più possibile la gestione delle varie sedi e aumentare l’efficienza del personale diminuendo tempi e costi.

Le quotidiane mansioni che un dipendente esercita nella nostra azienda sono varie. Si occupa di registrare un nuovo studente, di fornire tutte le info necessarie quali costi, orari delle lezioni, etc.., provvedere al registro dei pagamenti di bollettini e quote, prenotare la visita medica. Inoltre, deve tenersi in contatto con la motorizzazione per entrare a conoscenza delle date di esami teorici e pratici e aggiornare la bachecca di questi ultimi. In base alle date scelte dai corsisti i dipendenti provvedono alla prenotazione definitiva.”

## • **“Qual è l’iter e come funziona una nuova iscrizione?”**

“Un nuovo studente che si presenta in autoscuola per conseguire una nuova patente dovrà per prima cosa essere registrato all’interno del sistema, si chiede che tipo di patente vuole sostenere, compilare un modulo con i propri dati anagrafici, successivamente si procederà con il pagamento dei bollettini ed infine si sottoporrà ad una visita medico-oculistica.”

- “Quali sono i requisiti per conseguire una nuova patente?”

“I requisiti per il conseguimento di una patente variano in base al tipo di patente che si intende perseguire.”

(Nella seguente tabella sono riportati i requisiti necessari per ogni categoria di patente)

tipo patente	requisiti
AM	14 anni
A1	16 anni
A2	18 anni
A	20 se si è titolari della patente A2 da 2 anni 21 anni per la guida dei tricicli 24 anni
B1	16 anni
A2, B, B96, BE	18 anni
C1	18 anni con obbligo di patente B
C1E	18 anni con obbligo di patente C1
C	21 anni con l'obbligo di patente B (18 se si consegue la CQC merci)
CE	21 anni con l'obbligo di patente C (18 se si consegue la CQC merci)
D1	21 anni con l'obbligo di patente B
D1E	21 anni con l'obbligo di patente D1
D	24 anni con obbligo della patente B (21 anni se si consegue la CQC persone)
DE	24 anni, con obbligo della patente D (21 se si consegue la CQC persone)
KA	21 anni con obbligo della patente A
KB	21 anni con obbligo della patente B
CQC persone/merci	18 anni

*Tabella 1: Requisiti patenti*

- “**Come avviene la visita medico-oculistica?**”

“È sufficiente che lo studente si presenti in autoscuola e prenoti la visita e sarà poi il medico a procedere con il controllo all'interno della sede, per accettare lo stato di salute e valutare eventualmente uso di occhiali/lenti.

Sarà poi compito dell'oculista notificare all'autoscuola l'esito del consulto medico.”

- “**Quali sono i passi successivi all'iscrizione?**”

“Superato questo primo step lo studente dovrà versare la prima rata, la quale comprende l'iscrizione al primo appello e il corso teorico, che permette allo studente di acquisire conoscenze approfondite riguardo tutti i cartelli stradali, precedenze, distanze di sicurezza, limiti di velocità e qualche conoscenza base riguardo il funzionamento dell'autoveicolo.

Le lezioni si svolgono, per singolo studente, 3 volte a settimana in diversi orari stabiliti che possono essere scelti a piacere”

- “**In che cosa consiste l'esame teorico?**”

“L'esame teorico consiste nella soluzione di un quiz (vero o falso) a crocette composto da 40 domande. Gli argomenti del test verteranno su tutto il programma studiato durante le lezioni teoriche (quindi segnaletica stradale, verticale, orizzontale, spie, incroci, rotatorie etc...).

Il candidato potrà totalizzare massimo quattro errori per superare l'esame con esito positivo. Nel caso in cui gli errori siano superiori a quattro la prova verrà considerata negativa e sarà quindi necessario prenotarsi nuovamente e ripetere la prova teorica.”

- “**Come prosegue il percorso dello studente dopo il superamento dell'esame teorico?**”

Superato l'esame teorico si entra in possesso del foglio rosa che dà la possibilità allo studente di iniziare la parte pratica, ovvero le guide del mezzo. Sia in modo autonomo (per la patente B dovrà essere accompagnato da una persona in possesso della stessa da almeno 10 anni e non superare i 65 anni di età) che con l'insegnamento dei nostri istruttori.

In base al tipo di patente lo studente dovrà sostenere come minimo 6 ore di guida con gli istruttori dell'autoscuola per potersi iscrivere all'esame pratico.

Una volta superate le ore minime lo studente potrà decidere se continuare ulteriormente ad esercitarsi pagando un sovrapprezzo per ogni guida.

Passati almeno 30 giorni dal conseguimento del foglio rosa (validità di un anno) il candidato potrà prenotare e sostenere l'esame pratico. Superato l'esame allo studente verrà rilasciata la patente di guida. Nel caso in cui lo studente venisse bocciato la prima volta dovrà ripetere quest'ultimo. Se anche la seconda volta non sarà promosso il candidato dovrà ripercorrere tutto l'iter.

Tutte le date degli esami sono comunicate all'autoscuola dalla motorizzazione della provincia.

- **“Invece in che cosa consiste l'esame pratico?”**

“L'esame pratico consiste in una prova di guida su strada in cui l'esaminando, accompagnato da un istruttore della nostra scuola guida e da un esaminatore della motorizzazione, affronterà un percorso urbano. A questa prova l'esaminatore può decidere, a sua discrezione, se porre allo studente anche qualche domanda riguardante la parte teorica del corso. Sulla base del comportamento e delle competenze che l'alunno mostrerà in sede d'esame, l'esaminatore deciderà se promuovere o meno lo studente.”

- **“Una volta sostenuti due esami e non avendoli passati che succede?”**

“La nostra autoscuola consente agli studenti di sostenere l'esame due volte. Uno studente che non riesce a superare la prima volta la prova (teorica o pratica) potrà dare nuovamente l'esame pagando solo un sovrapprezzo. Nell'eventualità in cui lo studente non riuscisse a passare l'esame (teorico o pratico) una seconda volta allora si vedrà costretto a iscriversi nuovamente al corso, sostenere ancora una volta la visita medica e, nel caso lo ritenesse necessario, continuare a frequentare i corsi teorici o a prenotare altre guide con uno dei nostri istruttori. Dopo tutto ciò gli sarà consentito prenotarsi per uno nuovo esame.”

- “**Come funziona il rinnovo?**”

“Il rinnovo della patente è un obbligo imposto dal Codice della strada. Si tratta di una verifica periodica dei requisiti psicofisici dell’automobilista, per garantire la sicurezza sua e quella pubblica.

La patente ha una data di scadenza ed è variabile in base all’età e al tipo. Da qualche anno, inoltre, la data di scadenza coincide con la data di nascita.

Occorre quindi fare richiesta di rinnovo alla motorizzazione, sottoporsi a esame medico-oculistico, in caso positivo consentirà il rinnovo altrimenti verrà negato.”

- “**Quali sono le tariffe proposte agli studenti?**”

“I costi sono i seguenti:

-Tre bollettini iniziali, due da 16€ e uno da 24€

-Visita medica 40€

-Iscrizione 250€

-Guide 15€ l’una

-Ripetizione (prima volta) esame teorico/pratico 50€

-Ripetizione (dopo la seconda volta) esame teorico/pratico 250€

-Rinnovo patente (bollettini e visita medica) 80€”

## 2. Descrizione del sistema in linguaggio naturale

Il progetto proposto consiste nella realizzazione di un sistema informativo per la gestione di un'Autoscuola. Il software progettato e realizzato dovrà gestire il pagamento dei corsi teorici/pratici, rinnovo delle patenti di guida, prenotazione visite oculistiche, prenotazione esercitazione con istruttore e prenotazione data esame teorico/pratico. L'autoscuola è aperta cinque giorni la settimana, dal lunedì al venerdì, tutto l'anno, ad eccezione delle festività.

Ogni dipendente ha una sua mansione specifica.

- Il segretario si occupa di inserire nel sistema il nuovo candidato.

Per ogni studente i dati raccolti sono: nome, cognome, data di nascita e luogo, codice fiscale, e-mail, telefono. Rilascia le info necessarie per l'immatricolazione, quindi, bollettini da pagare, prima quota e calendario lezioni.

Una volta che lo studente avrà portato la ricevuta del bollettino e versato la prima quota gli verrà rilasciato un ID personale utile nel prenotarsi alle lezioni.

Per ognuna di esse si dovrà memorizzare la data e l'orario di inizio e di fine.

Inoltre, inserisce a sistema le prenotazioni degli esami sia teorici che pratici, comunicando alla motorizzazione i dati dello studente.

Prima di poter frequentare le lezioni teoriche, però, il candidato deve sottoporsi a visita medica, verrà fissato un appuntamento e in caso di esito positivo, il medico manderà avanti la pratica abilitando l'utilizzo dell'ID.

Quando il candidato si vuole prenotare all'esame teorico consulta il calendario con le date disponibile e chiede al segretario di registrarlo per la data scelta, in caso ci fosse posto viene aggiunto alla lista degli iscritti. Altrimenti viene iscritto automaticamente all'appello seguente.

- L'istruttore ha il compito di far acquisire a tutti gli studenti le conoscenze approfondite riguardo i cartelli stradali, precedenze, distanze di sicurezza, limiti di velocità e qualche conoscenza base riguardo il funzionamento dell'autoveicolo, ed inoltre aggiorna il calendario delle lezioni comunicando eventuali variazioni.

- L'istruttore di guida ha l'incarico di assistere lo studente alla guida dell'autoveicolo, facendogli mettere in pratica le conoscenze acquisite durante i corsi teorici implementando conoscenze di base sulla meccanica del mezzo e infine prepararlo per l'esame pratico.

Passati 30 giorni dal conseguimento del foglio rosa e effettuate le 6 ore minime di guida, si potrà procedere con la prenotazione dell'esame pratico. Nuovamente l'alunno sceglie la data del primo appello disponibile. Se l'esito risulterà essere positivo

e verrà quindi rilasciata la patente di guida termina il percorso dello studente. Se l'esito sarà negativo dovranno passare altri 30 giorni e pagare per rii-iscriversi all'esame.

Nel caso in cui lo studente venga bocciato due volte allo stesso esame ritorna alla situazione di partenza essendo però già registrato nel sistema mantenendo quindi dati e ID. Ovviamente non avrà i permessi di usufruire dei servizi dell'autoscuola previa completamente primo step.

Per quanto concerne il rinnovo della patente il cliente che richiede la pratica (che si trova nella parte di documentazione) deve compilare i documenti, pagare le tasse e i vari bollettini, e infine sostenere una visita medica.

- Il medico oculista si occupa di prenotare le visite mediche per i clienti dell'autoscuola, visitarli e verificare la loro idoneità alla guida. L'esito viene comunicato al segretario che procede ad aggiornare le informazioni del cliente interessato.

### 3. Diagramma dei sistemi

Il seguente diagramma dei sistemi, detto anche diagramma di contesto, illustra le relazioni tra il sistema di gestione dell'autoscuola e altri sistemi con i quali richiede dati e servizi.

Il sistema “Gestore Pagamenti” si occupa di registrare e certificare i pagamenti effettuati tramite carta di credito e PayPal.

Il “Sistema Notifiche Push” si occupa di inviare notifiche al cliente senza che debba effettuare operazioni di scaricamento.

Il sistema “Motorizzazione” comunica al sistema “Gestore Autoscuola Sprint” le date degli esami e i risultati degli esami svolti. Inoltre, riceve dall'autoscuola informazioni relative al numero di studenti che svolgeranno gli esami e i loro dati.

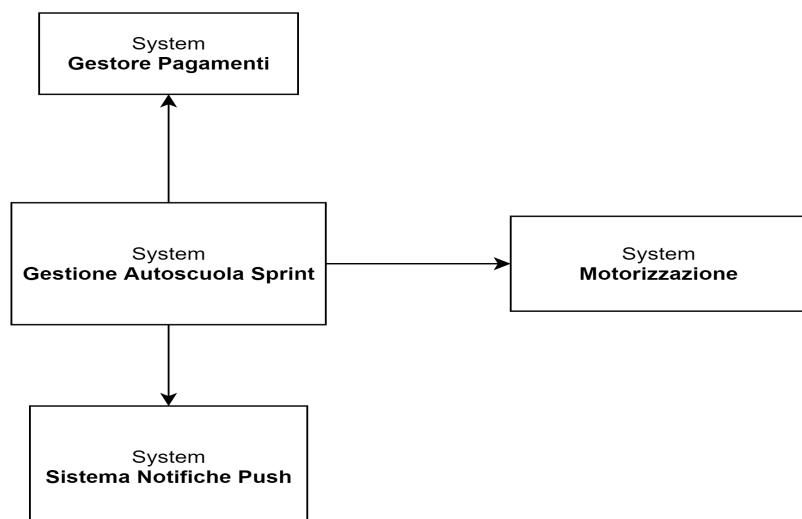


Figura 1: diagramma dei sistemi

## 4. Glossario dei termini

È buona norma corredare la descrizione in linguaggio naturale di un glossario dei termini. Il glossario dei termini contiene tutte le parole che possono generare ambiguità con la relativa spiegazione.

Nella tabella di seguito è contenuto il glossario dei termini della descrizione del gestionale spiegato in precedenza.

Termine	Descrizione	Tipo	Sinonimi
<b>Autoscuola</b>	Istituto dedicato all'educazione stradale, l'istruzione e la formazione dei conducenti.	Business	Scuola guida
<b>Corso teorico</b>	Lezioni svolte nella sede dell'autoscuola con lo scopo di fornire conoscenze teoriche agli studenti riguardo le norme di guida.	Business	Nessuno
<b>Corso pratico</b>	Lezioni di guida pratica svolta dallo studente in compagnia dell'istruttore con uno dei mezzi messi a disposizione dall'autoscuola.	Business	Nessuno
<b>Bollettino</b>	Modulo composto di tre parti che serve per eseguire presso un ufficio postale versamenti.	Business	Nessuno
<b>Servizi</b>	Complesso di attività messe a disposizione dei clienti: corsi teorici/pratici, esami teorici/pratici, visite oculistiche.	Business	Nessuno
<b>Patente di guida</b>	Autorizzazione necessaria per condurre su strade pubbliche veicoli a motore.	Business	Licenza di guida
<b>Rinnovo della patente</b>	Procedura periodica mediante la quale un cittadino già provvisto di patente rinnova la sua validità.	Business	Nessuno
<b>Prenotazione</b>	Atto mediante il quale vengono riservati servizi da parte del cliente.	Business	Assegnamento
<b>Visita oculistica</b>	Consulta medico atto a verificare la necessità del paziente di utilizzare lenti/occhiali alla guida	Business	Nessuno
<b>Studente</b>	Cliente della scuola guida	Business	Alunno
<b>ID personale</b>	Sequenza di numeri assegnata ad uno studente per identificarlo	Tecnico	Codice identificativo
<b>Esame teorico</b>	Prova a quiz che, se superata, consente allo studente di accedere alla parte pratica del corso	Business	Nessuno
<b>Esame pratico</b>	Prova di guida su strada al termine della quale, qualora l'esaminatore lo ritenesse opportuno, lo studente riceverà la patente di guida	Business	Nessuno
<b>Foglio rosa</b>	Licenza consegnata al candidato una volta superato l'esame di teoria per consentirgli di esercitarsi su strada.	Business	Nessuno
<b>Motorizzazione</b>	Ente pubblico che ha in carico il rispetto delle normative tecniche sul trasporto civile in un dato paese	Business	Nessuno
<b>Istruttore</b>	Dipendente della scuola guida che si occupa della preparazione teorica e pratica degli studenti	Business	Insegnante

Tabella 2: Glossario dei termini

## 5. Analisi dei requisiti (requisiti funzionali e non funzionali)

Dividiamo i nostri requisiti in funzionali e non funzionali.

I requisiti funzionali sono definizioni di servizi che il sistema deve fornire. Indicano come il sistema dovrebbe reagire a particolari input e come dovrebbe comportarsi in particolari situazioni. In alcuni casi possono stabilire esplicitamente che cosa il sistema non dovrebbe fare.

I requisiti non funzionali sono vincoli sulle funzioni o sui servizi offerti dal sistema. Includono, vincoli temporali e sul processo di sviluppo, e vincoli imposti dagli standard. Di solito si applicano al sistema completo, non a singole funzioni o servizi.

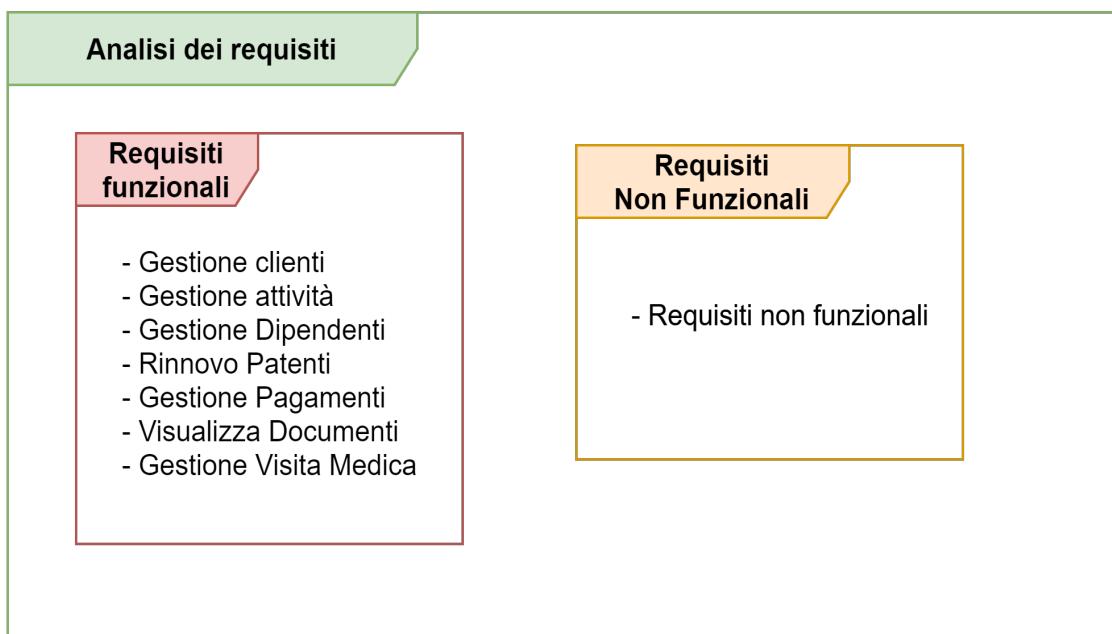


Figura 2: Analisi dei Requisiti

## 5.1 Requisiti funzionali

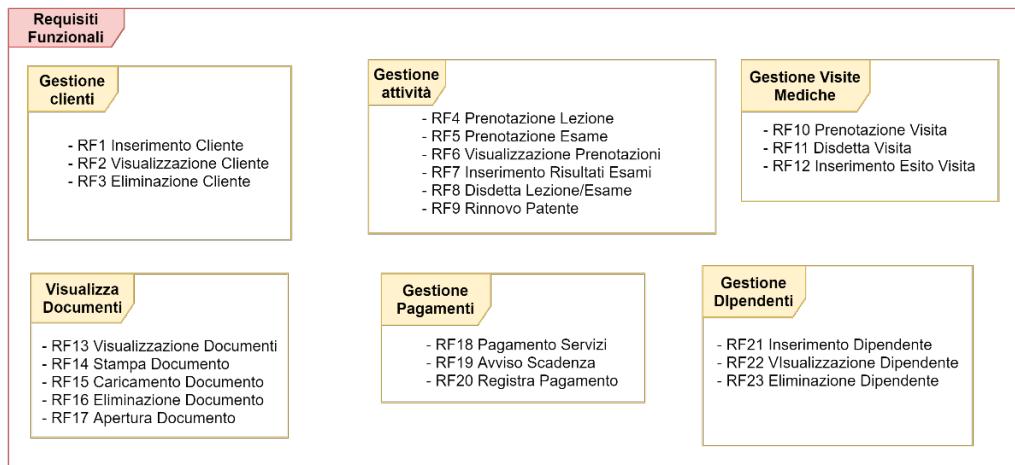


Figura 3: Requisiti Funzionali

Requisito	Descrizione
RF1 Inserimento Cliente	Il sistema dovrà gestire l'inserimento delle informazioni relative a un nuovo cliente.
RF2 Visualizzazione Cliente	Il sistema dovrà gestire la visualizzazione delle informazioni relative a un nuovo cliente.
RF3 Eliminazione Cliente	Il sistema dovrà gestire l'eliminazione delle informazioni relative a un cliente.
RF4 Prenotazione Lezione	Il sistema dovrà gestire l'inserimento di prenotazioni per lezioni.
RF5 Prenotazione Esame	Il sistema dovrà gestire l'inserimento di prenotazioni per esami.
RF6 Visualizzazione Prenotazioni	Il sistema dovrà gestire la visualizzazione delle prenotazioni effettuate.
RF7 Inserimento Risultati Esami	Il sistema dovrà gestire l'inserimento dei risultati degli esami.
RF8 Disdetta Lezioni/Esami	Il sistema dovrà gestire le cancellazioni di prenotazione per lezioni ed esami.
RF9 Rinnovo Patente	Il sistema dovrà gestire la procedura di rinnovo della patente.
RF10 Prenotazione Visita	Il sistema dovrà gestire l'inserimento di prenotazioni per visite mediche.
RF11 Disdetta Visita	Il sistema dovrà gestire la cancellazione di prenotazioni per le visite mediche.
RF12 Inserimento Esito Visita	Il sistema dovrà comunicare l'esito delle visite mediche dei clienti.
RF13 Visualizzazione Documenti	Il sistema dovrà visualizzare i documenti relativi alle attività dell'autoscuola.
RF14 Stampa Documento	Il sistema dovrà provvedere a mandare in stampa il documento selezionato.
RF15 Caricamento Documento	Il sistema dovrà gestire il caricamento di un nuovo documento all'interno dello stesso.
RF16 Eliminazione Documento	Il sistema dovrà gestire l'eliminazione del documento.
RF17 Apertura Documento	Il sistema dovrà gestire l'apertura del documento.
RF18 Pagamento Servizi	Il sistema dovrà gestire il pagamento dei servizi dell'autoscuola.
RF19 Avviso Scadenza	Il sistema dovrà comunicare, entro la data limite del pagamento stabilito, i clienti che devono ancora effettuare i loro pagamenti.
RF20 Registra Pagamento	Il sistema dovrà gestire la visualizzazione dei pagamenti effettuati e quelli che devono ancora essere saldati.
RF21 Inserimento Dipendente	Il sistema dovrà gestire l'inserimento delle informazioni di un nuovo dipendente.
RF22 Visualizzazione Dipendente	Il sistema dovrà visualizzare le informazioni relative a un dipendente.
RF23 Eliminazione Dipendente	Il sistema dovrà gestire l'eliminazione delle informazioni relative a un dipendente.

Tabella 3: Requisiti funzionali

## 5.2 Requisiti non funzionali

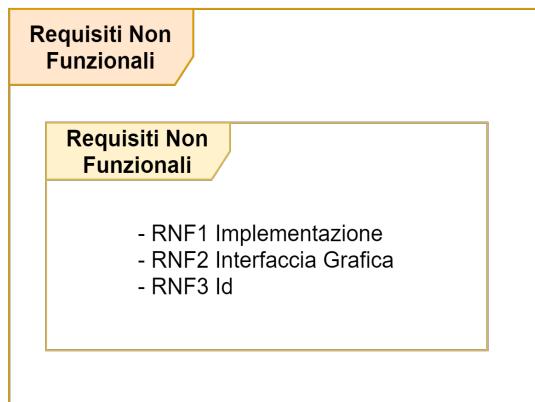


Figura 4: Requisiti Non Funzionali

Requisito	Descrizione
RNF1 Implementazione	Il sistema dovrà essere realizzato in tecnologia Python 3.
RNF2 Interfaccia grafica	Il sistema dovrà essere integrato con una interfaccia grafica
RNF3 Id	Il sistema non dovrà consentire l'impegno di id già in uso

Tabella 4: Requisiti non funzionali

## 6. Casi d'uso: Attori

Nel nostro software abbiamo tre diversi attori che interagiscono con il sistema. Il segretario, l'istruttore e il medico.

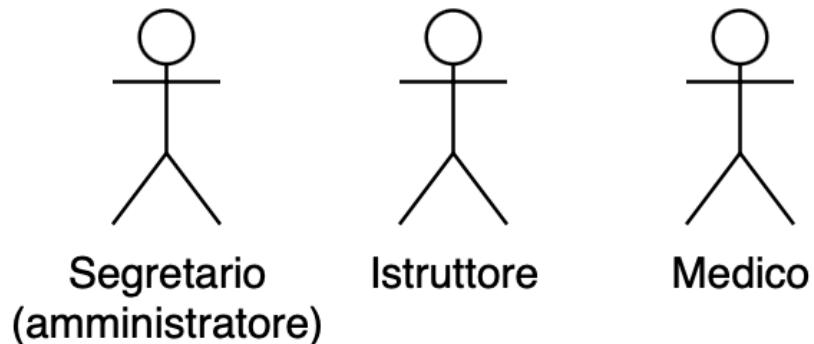


Figura 5: Attori

## 7. Descrizione dei casi d'uso

I casi d'uso sono un modo di descrivere le interazioni tra utenti e sistema utilizzando un modello grafico e un testo strutturato. Nella loro forma più semplice, un caso d'uso identifica gli attori coinvolti in un'interazione e assegna un nome al tipo di interazione. Poi si aggiungono altre informazioni che descrivono l'interazione con il sistema; queste informazioni possono essere descrizioni testuali o modelli grafici, come i diagrammi di sequenza o di stato UML.

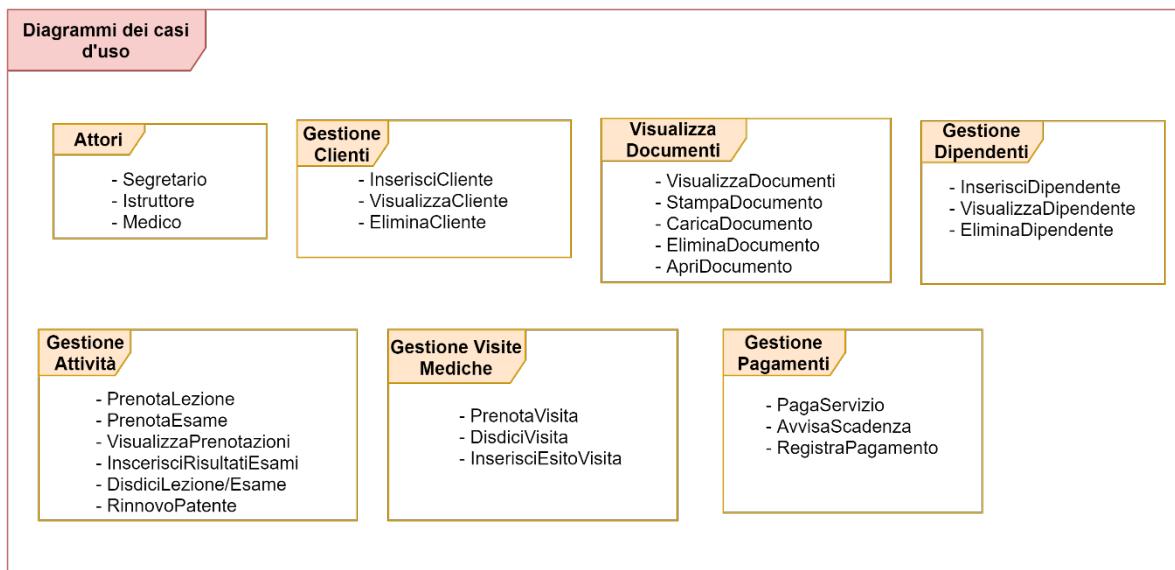


Figura 6: Casi d'Uso

Vediamo ora, per ogni gruppo di scenari che abbiamo definito in precedenza, il corrispettivo diagramma e la descrizione.

Andremo ora a descrivere ogni caso d'uso del nostro sistema, elencheremo i vincoli (precondizioni e post condizioni) e mostreremo le sequenze degli eventi principali ed alternative (se esistono)

## 7.1 UC: Gestione Dipendenti

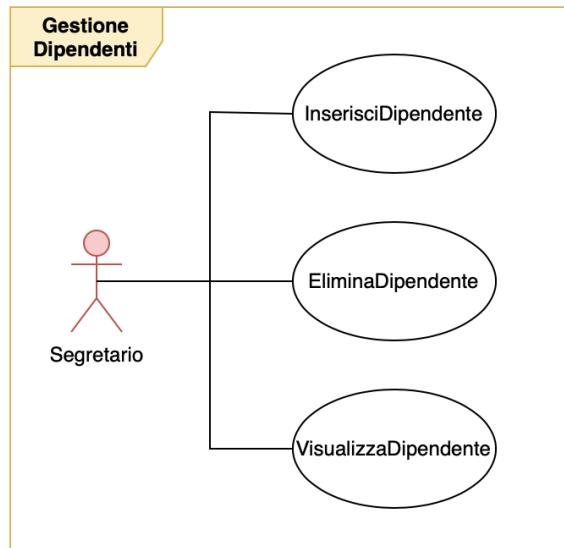


Figura 7: Diagramma dei Casi d'Uso per la Gestione dei Dipendenti

### 1) InserisciDipendente

Questo caso d'uso si verifica nel caso in cui il segretario voglia inserire un nuovo dipendente a sistema.

#### Pre-condizioni

Il dipendente non esiste a sistema.

#### Post-condizioni

Il dipendente esiste a sistema.

#### Sequenza degli eventi principale

1. Il caso d'uso inizia quando il segretario dell'autoscuola vuole inserire un nuovo dipendente.
2. Il sistema visualizza la schermata di inserimento informazioni del nuovo dipendente.
3. Il segretario inserisce tutte le informazioni necessarie.
4. Il segretario avvia la procedura di inserimento del nuovo dipendente.
5. Il sistema inserisce con successo il nuovo dipendente.

#### Sequenza degli eventi alternativa

La sequenza di eventi alternativa inizia dal punto 4.

5. Le informazioni immesse dal segretario sono incomplete.
6. L'inserimento del nuovo dipendente non va a buon fine.

## 2) VisualizzaDipendente

Questo caso d'uso si verifica qualora il segretario voglia visualizzare le informazioni relative a un dipendente e, se desidera, apportare modifiche.

### Pre-condizioni

Il dipendente esiste a sistema.

### Post-condizioni

Nessuna.

### Sequenza degli eventi principale

1. Il caso d'uso ha inizio quando il segretario desidera visualizzare le informazioni di un dipendente.
2. Il sistema stampa a schermo l'elenco dei dipendenti.
3. Il segretario sceglie il dipendente di cui vuole visualizzare le informazioni.
4. Il sistema stampa a schermo le informazioni del dipendente selezionato.

### Sequenza degli eventi alternativa

La sequenza alternativa inizia al punto 4.

5. Il segretario preme il bottone per modificare le informazioni relative al dipendente scelto.
6. Il segretario modifica i campi da aggiornare.
7. Le modifiche apportate vengono salvate nel sistema.

## 3) EliminaDipendente

Questo caso d'uso si verifica qualora il segretario voglia eliminare le informazioni relative a un dipendente dal sistema.

### Pre-condizioni

Il dipendente esiste a sistema.

### Post-condizioni

Il dipendente non esiste a sistema.

### Sequenza degli eventi principale

1. Il caso d'uso inizia quando il segretario vuole eliminare le informazioni di un dipendente.
2. Il sistema visualizza a schermo l'elenco dei dipendenti.

3. Il segretario avvia la procedura di eliminazione del cliente scelto.
4. Il sistema elimina dal sistema le informazioni del dipendente.

#### **Sequenza degli eventi alternativa**

Nessuna.

## 7.2 UC: Visualizza Documenti

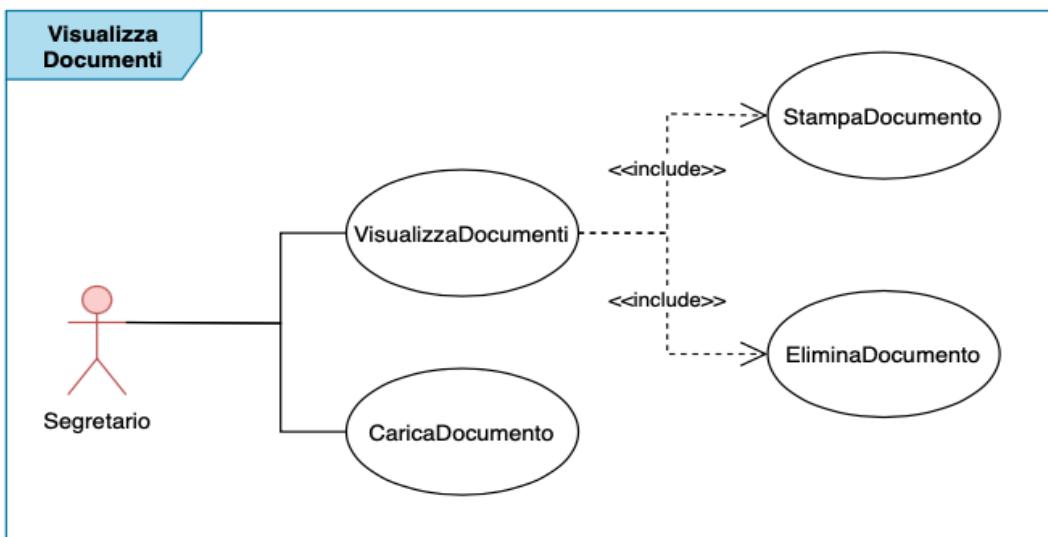


Figura 8: Diagramma dei Casi d'Uso per la Visualizzazione dei Documenti

#### **1) VisualizzaDocumenti**

Questo caso d'uso si verifica qualora il segretario voglia visualizzare l'archivio dei documenti.

#### **Pre-condizioni**

I documenti esistono a sistema.

#### **Post-condizioni**

Nessuna.

#### **Sequenza degli eventi principale**

1. Il caso d'uso ha inizio quando il segretario desidera visualizzare un documento.
2. Il sistema visualizza a schermo la lista di documenti presenti a sistema.

#### **Sequenza degli eventi alternativa**

Nessuna.

## **2) StampaDocumento**

Questo caso d'uso si verifica qualora il segretario voglia stampare uno o più documenti.

### **Pre-condizioni**

Il documento esiste a sistema.

### **Post-condizioni**

Nessuna.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario vuole stampare un documento.
2. Il sistema visualizza a schermo il documento.
3. Il segretario avvia la procedura di stampa.
4. Il sistema avvia la procedura di stampa del documento.

### **Sequenza degli eventi alternativa**

Nessuna.

## **3) EliminaDocumento**

Questo caso d'uso si verifica qualora il segretario voglia eliminare uno o più documenti.

### **Pre-condizioni**

Il documento esiste a sistema.

### **Post-condizioni**

Il documento non esiste a sistema.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario vuole eliminare un documento.
2. Il sistema visualizza a schermo il documento.
3. Il segretario avvia la procedura di eliminazione.
4. Il sistema elimina dal sistema il documento.

### **Sequenza degli eventi alternativa**

Nessuna.

#### **4) CaricaDocumento**

Questo caso d'uso si verifica qualora il segretario voglia caricare dei nuovi documenti.

##### **Pre-condizioni**

Il documento non esiste a sistema.

##### **Post-condizioni**

Il documento esiste a sistema.

##### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario dell'autoscuola vuole caricare a sistema un nuovo documento.
2. Il sistema visualizza la schermata di upload.
3. Il segretario sceglie il file da caricare.
4. Il segretario avvia la procedura di upload.
5. Il sistema inserisce con successo il nuovo documento.

##### **Sequenza di eventi alternativa**

La sequenza di eventi alternativa inizia dal punto 3.

4. Il file da caricare non è di un formato supportato (pdf)
5. La procedura di upload fallisce.

#### **5) ApriDocumento**

Questo caso d'uso si verifica qualora il segretario voglia visualizzare un documento presente all'interno dell'archivio.

##### **Pre-condizioni**

Il documento esiste a sistema.

##### **Post-condizioni**

Nessuna.

##### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario vuole aprire un documento presente all'interno dell'archivio.
2. Il sistema visualizza la lista dei documenti.
3. Il segretario sceglie il documento da visualizzare.

- Il documento viene aperto a finestra intera.

#### **Sequenza degli eventi alternativa**

Nessuna.

### 7.3 UC: Gestione Pagamenti

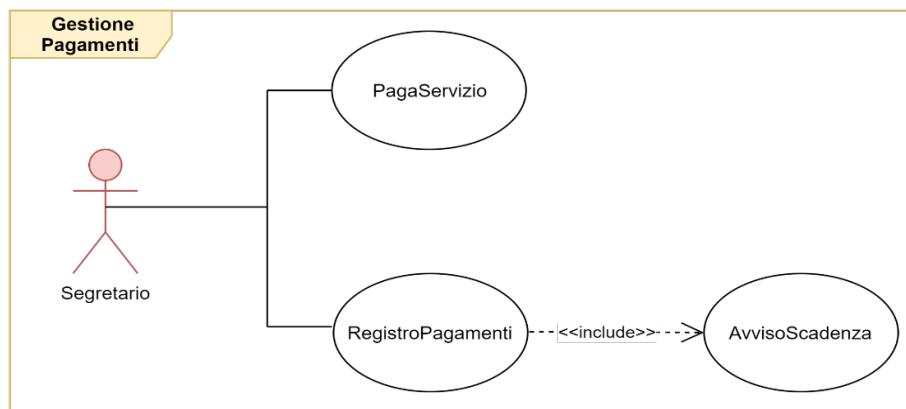


Figura 9: Diagramma dei Casi d'Uso per la Gestione dei Pagamenti

#### **1) PagaServizio**

Questo caso d'uso si verifica qualora un cliente voglia effettuare un pagamento per un servizio da lui sfruttato.

#### **Pre-condizioni**

Il cliente esiste a sistema.

#### **Post-condizioni**

Il pagamento viene registrato e il registro dei pagamenti viene aggiornato.

#### **Sequenza degli eventi principale**

- Il caso d'uso ha inizio quando il cliente vuole pagare uno o più servizi offerti dall'autoscuola e di cui ha usufruito.
- Il segretario allora seleziona il cliente intenzionato a effettuare il pagamento.
- Il segretario seleziona i servizi che il cliente desidera pagare.
- Il cliente sceglie se effettuare il pagamento tramite PayPal oppure in contanti.
- Una volta effettuato il pagamento, il registro pagamenti viene aggiornato.

#### **Sequenza degli eventi alternativa**

Nessuna.

## 2) RegistroPagamenti

Questo caso d'uso si verifica quando un cliente ha usufruito di uno o più servizi offerti dall'autoscuola.

### Pre-condizioni

Il cliente esiste a sistema.

### Post-condizioni

Il pagamento esiste a sistema.

### Sequenza degli eventi principale

1. Il caso d'uso inizia quando il segretario vuole inserire un nuovo pagamento.
2. Il sistema visualizza l'elenco dei pagamenti.
3. Il segretario seleziona il cliente dal menù a tendina.
4. Il segretario seleziona uno o più servizi di cui il cliente ha usufruito.
5. Il sistema registra i servizi selezionati e visualizza il prezzo totale.

### Sequenza degli eventi alternativa

Nessuna.

## 3) AvvisoScadenza

Questo caso d'uso si verifica qualora un servizio offerto dall'autoscuola non sia ancora stato pagato dal cliente due giorni prima della scadenza e una volta oltrepassato il termine ultimo di pagamento.

### Pre-condizione

Il servizio non pagato è presente nel registro dei pagamenti.

### Post-condizione

Il servizio risulta "pagato" nel registro dei pagamenti.

### Sequenza degli eventi principale

1. Il caso d'uso inizia quando il pagamento per un servizio non è stato effettuato due giorni prima della scadenza del pagamento.
2. Il sistema inserisce le informazioni del cliente, l'importo del pagamento da saldare e il termine ultimo nel file relativo agli avvisi di scadenza.

3. Il servizio, dopo che il debito è stato saldato, risulterà come “pagato” nel registro dei pagamenti e verrà cancellato dal file relativo agli avvisi di scadenza.

#### **Sequenza degli eventi alternativa**

Una sequenza degli eventi alternativa inizia dal punto 2.

3. Passati trenta giorni dal termine ultimo di pagamento di uno o più servizi il sistema provvederà a eliminare le informazioni relative al cliente insolvente.

## 7.4 UC: Gestione Visite Mediche

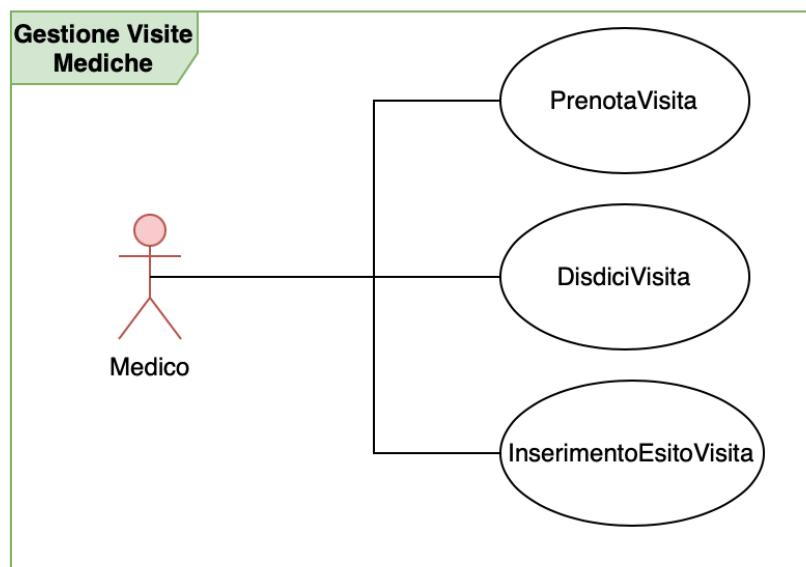


Figura 10: Diagramma dei Casi d'Uso per la Gestione delle Visite Mediche

#### **1) PrenotaVisita**

Questo caso d'uso si verifica nel caso in cui il medico abbia bisogno di prenotare una visita medica per un cliente.

#### **Pre-condizioni**

Il cliente esiste a sistema, non ha prenotato una visita e ha già effettuato i pagamenti necessari.

#### **Post-condizioni**

Il cliente ha una nuova prenotazione per un determinato servizio.

#### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il medico ha bisogno di inserire una nuova prenotazione.
2. Il medico inserisce le informazioni della nuova prenotazione.

3. Il medico avvia la procedura di inserimento nuova prenotazione, selezionando il servizio disponibile e un cliente già iscritto.
4. Il sistema registra con successo la nuova prenotazione per il cliente.

#### **Sequenza degli eventi alternativa**

Nessuna.

#### **2) DisdiciVisita**

Questo caso d'uso si verifica qualora il medico voglia disdire una prenotazione a sistema.

#### **Pre-condizioni**

Esiste a sistema la visita prenotata in precedenza.

#### **Post-condizioni**

La visita non esiste più a sistema.

#### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il medico dell'autoscuola vuole disdire una prenotazione.
2. Il sistema legge le informazioni della prenotazione scelta dal medico.
3. Il sistema visualizza a schermo la prenotazione.
4. Il medico avvia la procedura di eliminazione.
5. Il sistema elimina dal sistema la prenotazione della visita medica.

#### **Sequenza degli eventi alternativa**

Nessuna.

#### **3) InserimentoEsitoVisita**

Questo caso d'uso viene utilizzato nel caso in cui il medico voglia far visualizzare l'esito della visita ai clienti.

#### **Pre-condizioni**

La visita esiste a sistema.

#### **Post-condizioni**

Le informazioni del cliente vengono aggiornate.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il medico vuole inserire l'esito della visita medica.
2. Il sistema inserisce l'esito della visita medica.
3. Il sistema aggiorna le informazioni del cliente.

### **Sequenza degli eventi alternativa**

Nessuna.

## 7.5 UC: Gestione Clienti

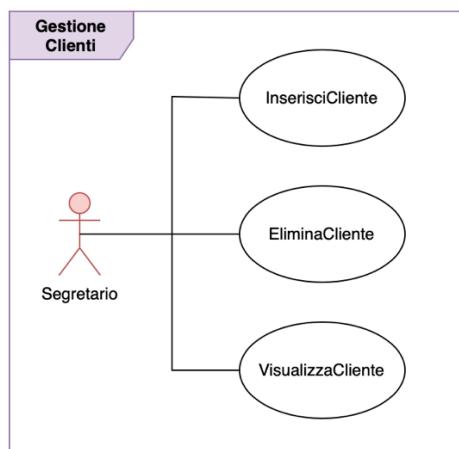


Figura 11: Diagramma dei Casi d'Uso per la Gestione dei Clienti

#### **1) InserisciCliente**

Questo caso d'uso si verifica nel caso in cui il segretario voglia inserire un nuovo cliente a sistema.

#### **Pre-condizioni**

Il cliente non esiste a sistema.

#### **Post-condizioni**

il cliente esiste a sistema.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario dell'autoscuola vuole inserire un nuovo cliente.
2. Il sistema visualizza la schermata di inserimento informazioni del nuovo cliente.
3. Il segretario fornisce tutte le informazioni richieste.
4. Il segretario avvia la procedura di inserimento nuovo cliente.
5. Il sistema aggiunge con successo il nuovo cliente a sistema.

### **Sequenza degli eventi alternativa**

La sequenza di eventi alternativa inizia dal punto 4.

**5.** Le informazioni immesse dal segretario sono incomplete.

**6.** L'inserimento del nuovo cliente viene annullata.

## **2) VisualizzaCliente**

Questo caso d'uso si verifica qualora il segretario voglia visualizzare le informazioni di un cliente.

### **Pre-condizioni**

Il cliente esiste a sistema.

### **Post-condizioni**

Nessuna

### **Sequenza degli eventi principale**

**1.** Il caso d'uso inizia quando il segretario vuole visualizzare i dati di un cliente.

**2.** Il sistema stampa a schermo la lista dei clienti.

**3.** Il segretario seleziona il cliente di cui vuole visualizzare le informazioni.

**4.** Il sistema visualizza a schermo le informazioni del cliente.

### **Sequenza degli eventi alternativa**

La sequenza alternativa inizia al punto 4.

**5.** Il segretario preme il bottone per modificare le informazioni relative al cliente scelto.

**6.** Il segretario modifica i campi da aggiornare.

**7.** Le modifiche apportate vengono salvate nel sistema.

## **3) EliminaCliente**

Questo caso d'uso si verifica qualora il segretario voglia eliminare un cliente dal sistema.

### **Pre-condizioni**

Il cliente esiste a sistema.

### **Post-condizioni**

Il cliente non esiste più a sistema.

### Sequenza degli eventi principale

1. Il caso d'uso inizia quando il segretario vuole eliminare i dati di un cliente.
- 2.. Il sistema visualizza a schermo la lista dei clienti.
- 3.. Il segretario avvia la procedura di eliminazione del cliente scelto.
4. Il sistema elimina le informazioni del cliente.

### Sequenza degli eventi alternativa

Nessuna.

## 7.6 UC: Gestione Attività

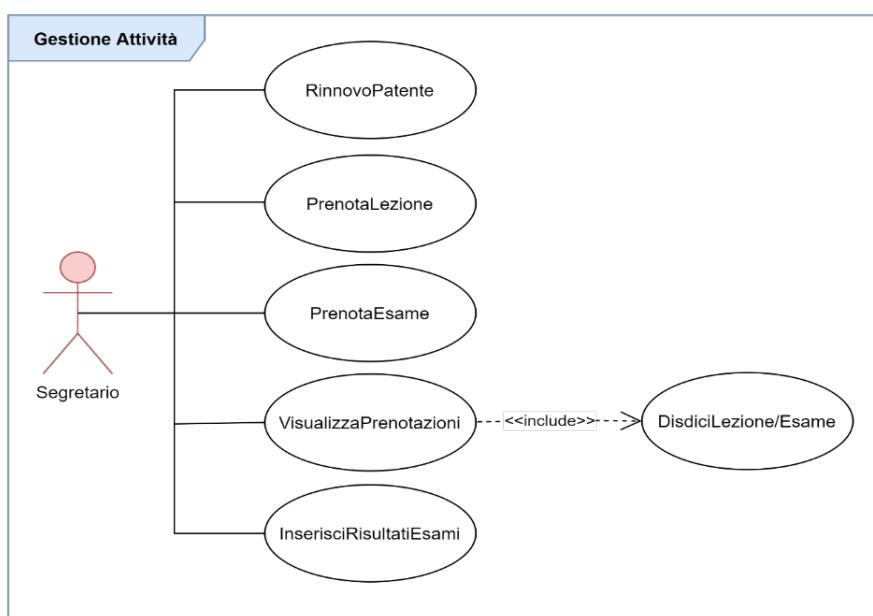


Figura 12: Diagramma dei casi d'uso per la gestione delle attività del segretario

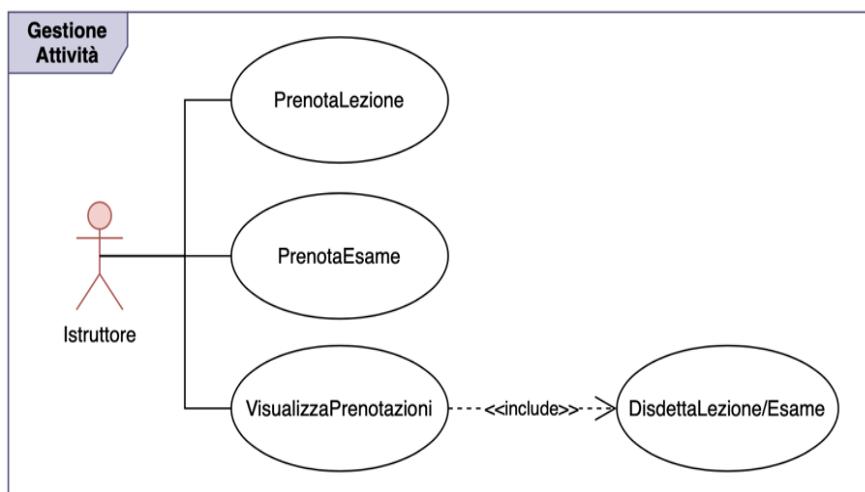


Figura 13: Diagramma dei casi d'uso per la gestione delle attività dell'istruttore

### **1) InserisciRisultatiEsami**

Questo caso d'uso si verifica nel momento in cui l'amministratore voglia inserire i risultati degli esami relativi ai suoi studenti.

#### **Pre-condizioni**

Lo studente deve aver svolto almeno l'esame teorico.

#### **Post-condizioni**

Nessuna

#### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando l'amministratore della scuola guida voglia inserire l'esito dell'esame di uno studente dell'autoscuola.
2. Il dipendente inserisce le informazioni relative all'esame.
3. Il sistema salva le informazioni aggiunte e aggiorna la situazione del cliente.

#### **Sequenza degli eventi alternativa**

Nessuna.

### **1) VisualizzaPrenotazioni**

Questo caso d'uso si verifica nel momento in cui il segretario o l'istruttore voglia visualizzare le prenotazioni di uno o più studenti.

#### **Pre-condizioni**

La prenotazione è presente nel sistema.

#### **Post-condizioni**

Nessuna.

#### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando il segretario o l'istruttore della scuola guida voglia visualizzare le informazioni delle prenotazioni.
2. Il sistema legge le informazioni relative alle prenotazioni.
3. Il sistema visualizza a schermo le informazioni delle prenotazioni.

#### **Sequenza degli eventi alternativa**

Nessuna.

## **2) DisdettaLezione/Esame**

Questo caso d'uso si verifica nel momento in cui il segretario o l'istruttore voglia disdire la prenotazione di un esame o lezione per un suo studente.

### **Pre-condizioni**

La prenotazione è presente nel sistema.

### **Post-condizioni**

La prenotazione viene rimossa dal sistema.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia qualora il segretario o l'istruttore della scuola guida voglia rimuovere una prenotazione per un esame/lezione.
2. Il sistema legge le informazioni relative alla prenotazione.
3. Il sistema visualizza a schermo le informazioni della prenotazione.
4. L'istruttore o il segretario procede con l'eliminazione della prenotazione.
5. Il sistema rimuove la prenotazione dello studente.

### **Sequenza degli eventi alternativa**

Nessuna.

## **3) RinnovoPatente**

Questo caso d'uso si verifica nel momento in cui il segretario voglia fare il rinnovo della patente di guida per un suo cliente.

### **Pre-condizioni**

Il cliente deve essere in possesso di una patente di guida.

### **Post-condizioni**

Il cliente rinnova la sua patente di guida.

### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando l'amministratore della scuola guida voglia fare il rinnovo della patente di guida.
2. L'amministratore inserisce i dati della patente di guida.
3. Il sistema legge i dati della patente di guida.

4. Il cliente provvede alla compilazione dei documenti e al pagamento delle tasse necessarie.
5. Il segretario prenota la visita medica.
6. Il sistema rinnova la patente di guida.

#### **Sequenza degli eventi alternativa**

La sequenza degli eventi alternativa inizia dal punto 5.

6. Qualora il medico riscontri una non idoneità del cliente può decidere di non consentire il rinnovo della patente.
7. Il sistema non rinnova la patente di guida.

#### **4) PrenotaLezione**

Questo caso d'uso si verifica nel momento in cui il segretario o l'istruttore voglia inserire una prenotazione per una lezione.

#### **Pre-condizioni**

Il cliente è presente nel sistema e ha effettuato la visita medica.

#### **Post-condizioni**

Lo studente ha una nuova prenotazione per una lezione.

#### **Sequenza degli eventi principale**

1. Il caso d'uso inizia quando l'amministratore della scuola guida voglia inserire una prenotazione per uno studente ad una lezione.
2. L'amministratore inizia la procedura di inserimento, selezionando la lezione disponibile e uno studente con id verificato.
3. Il sistema registra con successo lo studente alla lezione.

#### **Sequenza degli eventi alternativa**

Nessuna.

## 5) PrenotaEsame

Questo caso d'uso si verifica nel momento in cui l'amministratore voglia inserire una prenotazione per un esame.

### Pre-condizioni

Lo studente è presente nel sistema e la sua età è adeguata alla tipologia di patente per cui vuole sostenere l'esame. Inoltre, lo studente deve aver effettuato almeno 6 ore di guida con un istruttore.

### Post-condizioni

Lo studente ha una nuova prenotazione per un esame.

### Sequenza degli eventi principale

1. Il caso d'uso inizia quando l'amministratore della scuola guida voglia inserire una prenotazione per uno studente ad un esame.
2. L'amministratore inizia la procedura di inserimento, selezionando la data d'esame disponibile e uno studente con id verificato.
3. Il sistema registra con successo lo studente all'esame.

### Sequenza degli eventi alternativa

Nessuna.

## 8. Diagrammi dei casi d'uso

Di seguito riportiamo i diagrammi dei casi d'uso complessivi del nostro software divisi per i vari attori.

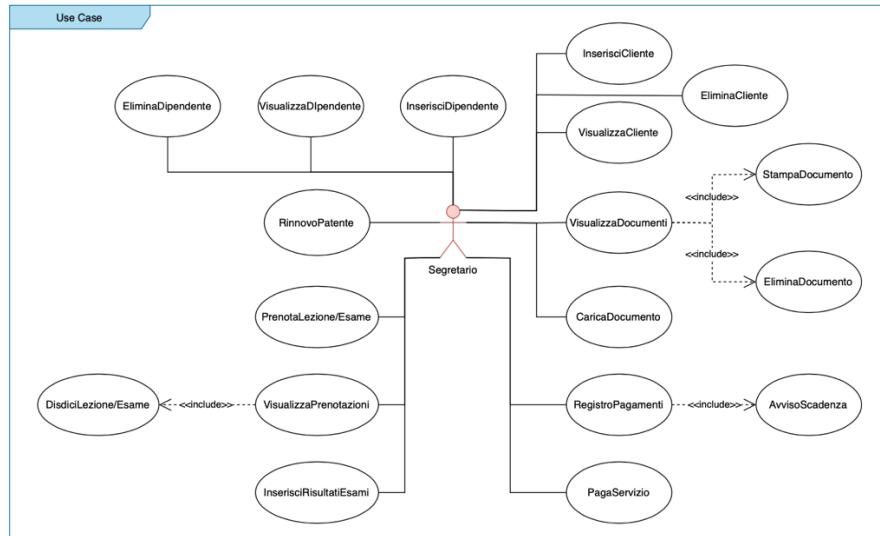


Figura 14: Diagramma dei Casi d'Uso Complessivo per le attività consentire al Segretario

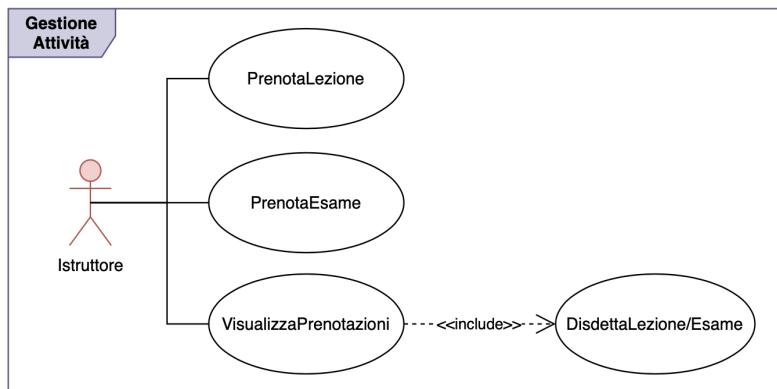


Figura 15: Diagramma dei Casi d'Uso Complessivo per le Attività consentire all'Istruttore

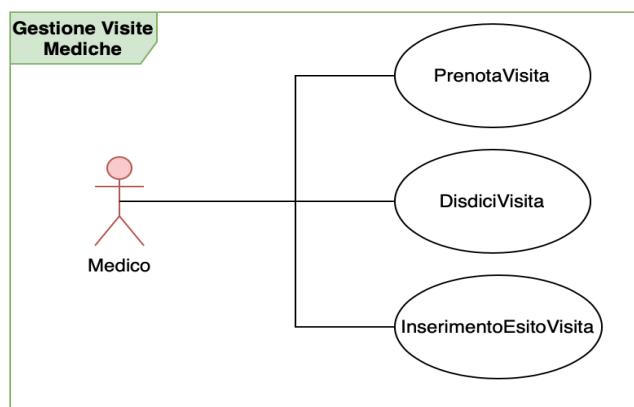


Figura 16: Diagramma dei Casi d'Uso Complessivo per le Attività consentite al Medico

## 9. Matrice di mapping requisiti – casi d'uso

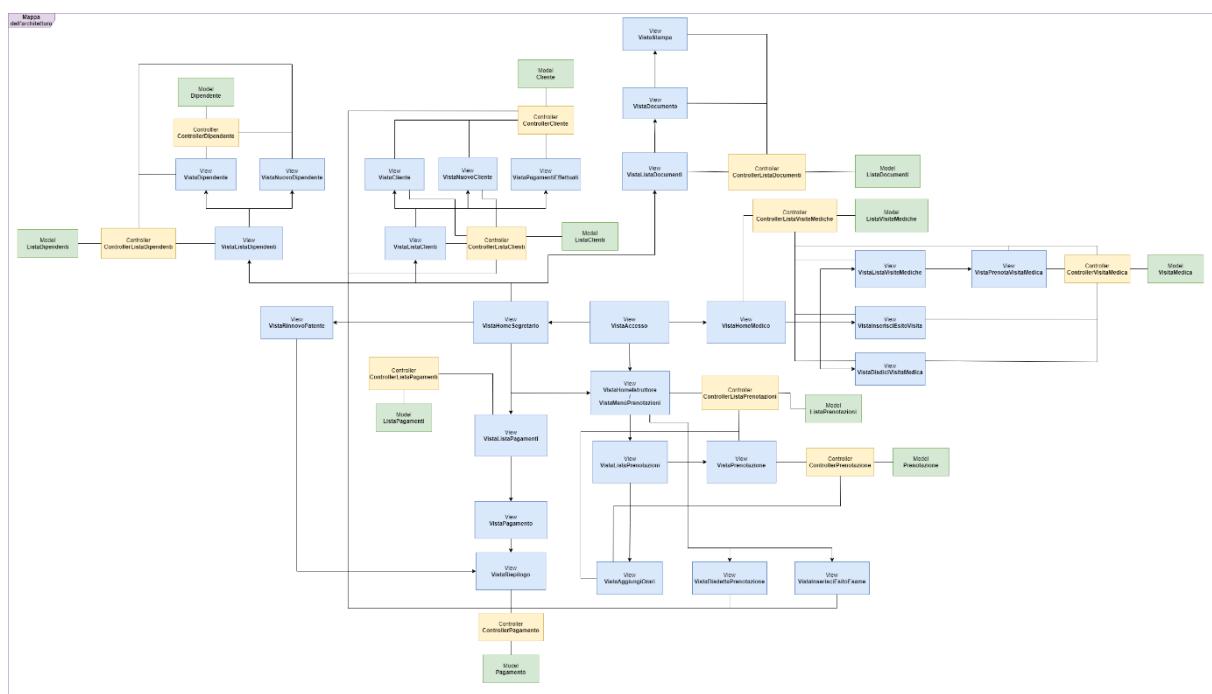
La matrice di mapping è un utile strumento per visualizzare a colpo d'occhio se i casi d'uso progettati soddisfano tutti i requisiti funzionali richiesti dal cliente.

	Rf1 Inserimento Cliente	Rf2 Visualizzazione Cliente	Rf3 Eliminazione Cliente	Rf4 Prenotazione Lettore	Rf5 Prenotazione Esame	Rf6 Visualizzazione Esami	Rf7 Prenotazione Risultati Esami	Rf8 Disetta Letton/Esami	Rf9 Rinnovo Patente	Rf10 Prenotazione Visita	Rf11 Disdetta Visita	Rf12 Inserimento Edto Visita	Rf13 Visualizzazione Documenti	Rf14 Stampa Documento	Rf15 Caricamento Documento	Rf16 Eliminazione Documento	Rf17 Pagamento Servizi	Rf18 Avviso Scadenza	Rf19 Registra Pagamento	Rf20 Inserimento Dipendente	Rf21 Visualizza Dipendente	Rf22 Eliminazione Dipendente	
Inserimento Cliente	x																						
Visualizzazione Cliente		x																					
Eliminazione Cliente		x	x																				
Prenota Visita											x												
Prenota Lettore		x		x																			
Prenota Esame		x			x																		
Visualizza Risultati Esami							x																
Rinnovo Patente								x															
Disdetta Letton/Esami								x		x											x		
Visualizza Prenotazioni									x		x										x		
Disdetta Visita									x		x				x							x	
Visualizzazione Edto Visita										x													
Registra Pagamento											x												
Avviso Scadenza											x	x										x	
Paga Servizio												x											
Elimina Dipendente												x									x		
Visualizza Dipendente												x									x		
Insersici Dipendente													x									x	
Cerca Documenti													x										
Stampa Documento													x	x									

Figura 17: Matrice di Mapping dei Requisiti

## 10. Mappa dell'Architettura

La mappa a seguire rappresenta l'architettura del nostro software, utile a descrivere come è organizzato il sistema. Il modello architettonale che abbiamo applicato per questo specifico prodotto software è un modello MVC (Model-View-Controller), modello utile quando si devono visualizzare dati ed interagire con essi. Questo modello è basato sulla suddivisione dei compiti fra i componenti del software che sono di 3 tipi: Model, gestiscono i dati del sistema e le operazioni a loro associate; View, gestiscono il modo in cui i dati verranno presentati all'utente, e Controller, che gestiscono e attuano i comandi andando a modificare lo stato degli altri due componenti. Abbiamo deciso di utilizzare questo pattern architettonale in quanto consapevoli del fatto che il nostro software avrebbe dovuto lavorare con dati che dovevano essere presentati e in caso modificati dall'utente. Inoltre, questi dati dovevano poter essere modificati senza che la loro rappresentazione venisse modificata. Anche altri vantaggi come la riusabilità del codice e l'indipendenza dei vari componenti in modo da poter suddividere facilmente il lavoro all'interno del gruppo sono stati fattori determinanti nella scelta di questo pattern per il nostro programma.



*Figura 18: Mappa dell'Architettura del Sistema*

# 11. Diagramma delle classi

Il diagramma delle classi è uno schema per sistemi orientati agli oggetti utile a visualizzare le sue classi e come queste sono associate. A seguire si può trovare il diagramma complessivo. All'interno degli insiemi di classi alcuni Controller e Model sono stati ripetuti per rendere più chiare le associazioni tra le classi e su quali dati lavorano.

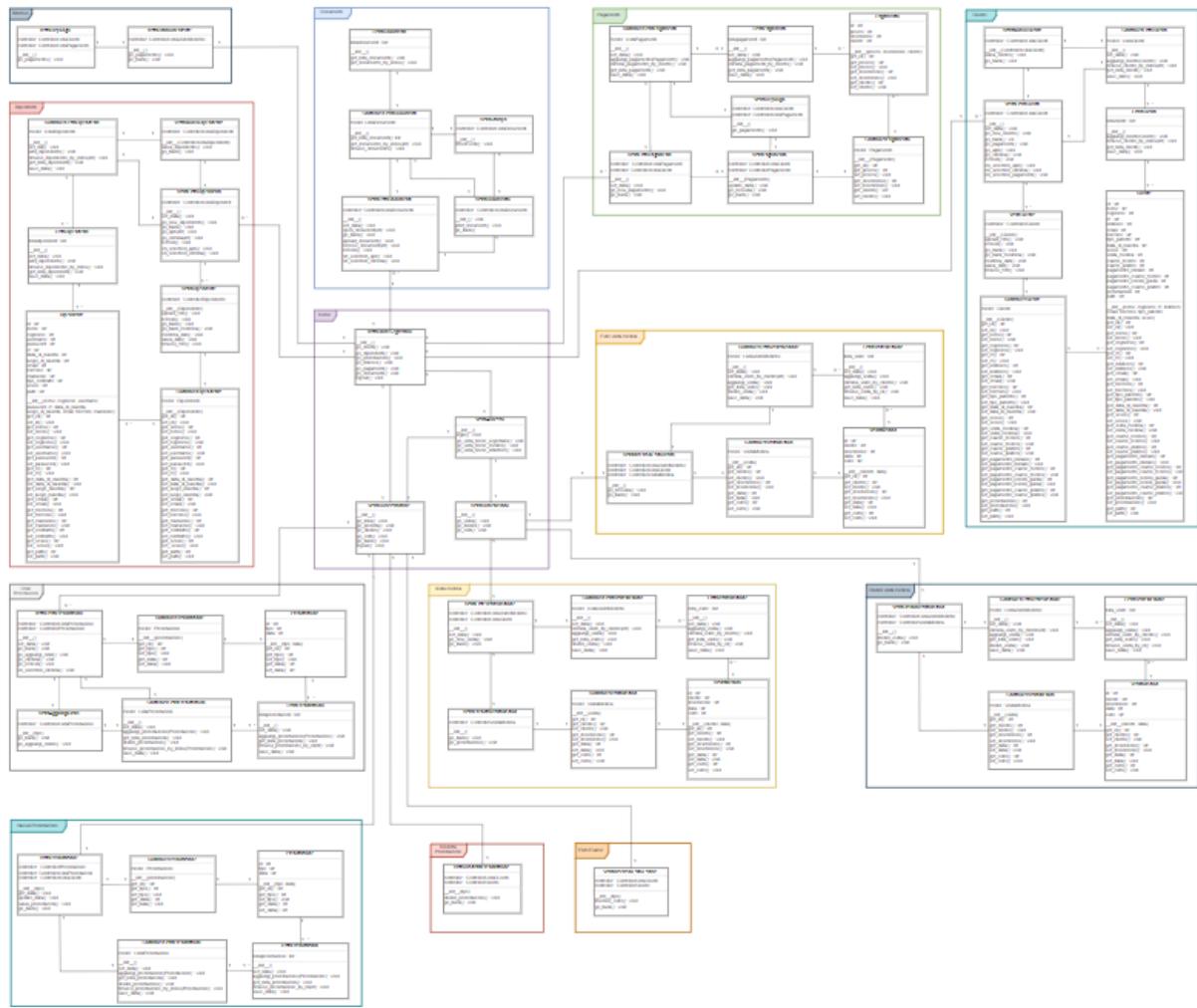


Figura 19: Diagramma delle Classi Complessivo

## 11.01 DC Home

In questo insieme di classi la VistaAccesso è la prima schermata che appare all'utente, consentendogli di inserire le proprie credenziali e, in base ad esse, essere indirizzato a una delle VisteHome: Segretario, Medico oppure Istruttore. Una volta nella Home, l'utente ha a disposizione diverse funzionalità in base al ruolo svolto all'interno dell'autoscuola. Dalla VistaHomeSegretario è possibile accedere alla VistaHomeIstruttore per gestire le operazioni relative alle prenotazioni: la finestra sarà però visualizzata come "Menù Prenotazioni". Questo perché all'istruttore è consentito gestire solo le prenotazioni mentre il segretario deve poter gestire tutte le attività dell'autoscuola, eccezion fatta per le visite mediche che sono gestite unicamente dal medico oculista dell'autoscuola.

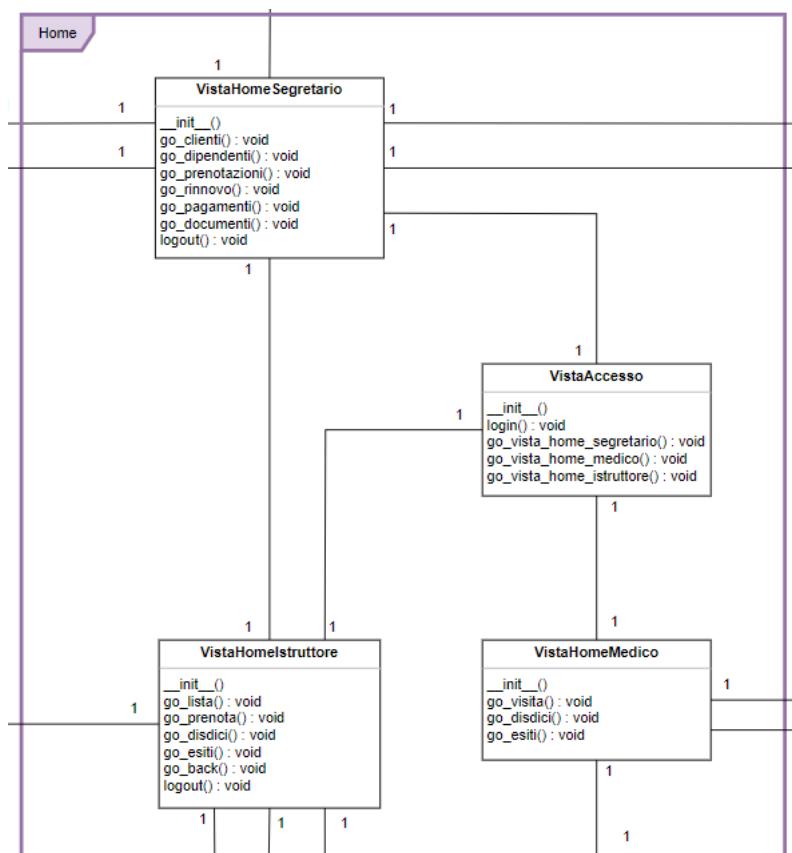


Figura 20: Diagramma delle Classi Home

## 11.02 DC Clienti

Il seguente gruppo di classi gestisce tutte le operazioni relative ai clienti. Innanzitutto, viene visualizzata la VistaListaClienti, che mostra la situazione del cliente a livello di pagamenti ed esami. Tramite un pulsante è possibile aprire la VistaNuovoCliente che consente di registrare nuovi clienti dell'autoscuola inserendo i dati ad essi relativi. Nella VistaListaClienti, ad ogni cliente, sono associati due pulsanti, uno che apre la VistaCliente, utile a visualizzare tutte le informazioni a lui relative, e un pulsante per eliminare il cliente dal sistema. Le informazioni alla VistaListaClienti sono comunicate dal ControllerListaClienti, le informazioni mostrate nella VistaCliente sono invece ottenute attraverso il ControllerCliente.

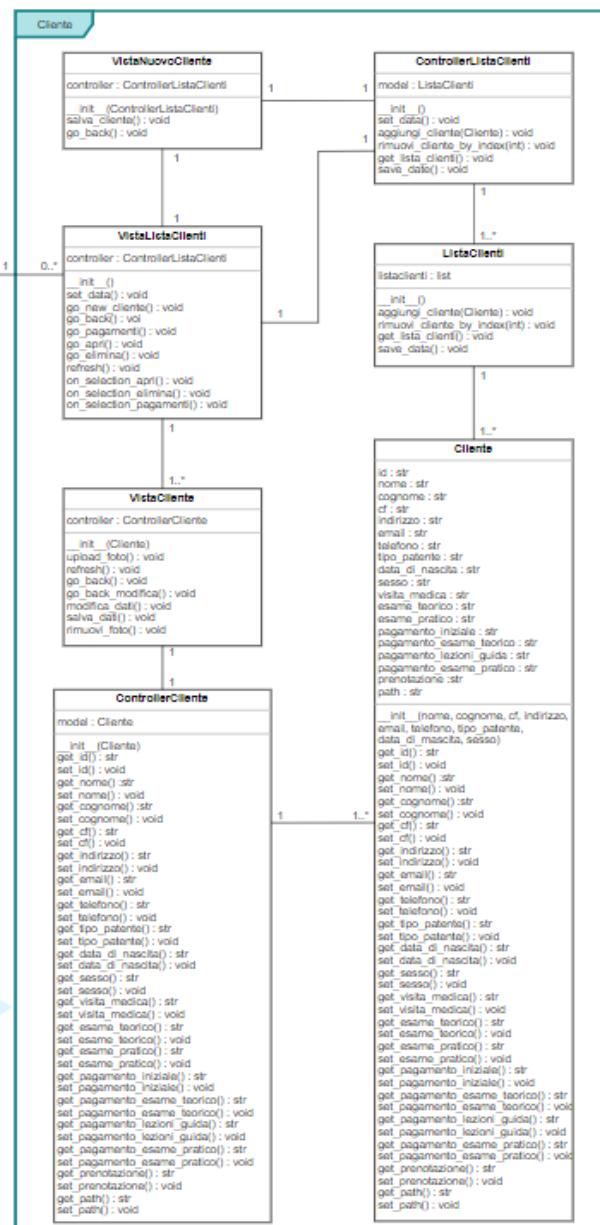


Figura 21: Diagramma delle Classi Clienti

## 11.03 DC Dipendenti

Il gruppo di classi a seguire gestisce le operazioni relative ai dipendenti, in maniera del tutto analoga a come erano gestite quelle dei clienti. Viene visualizzata la VistaListaDipendenti, che mostra le informazioni più importanti relative ad ogni dipendente ricevute dal ControllerListaDipendenti. Ad ogni dipendente è associato un bottone che apre la VistaDipendente, che, grazie al ControllerDipendente, fornisce tutte le informazioni del dipendente scelto. Per ogni dipendente è presente anche il bottone per eliminare le informazioni a lui relative e vi è un pulsante per aprire la VistaNuovoDipendente, utile a registrare nuovi dipendenti nel sistema.

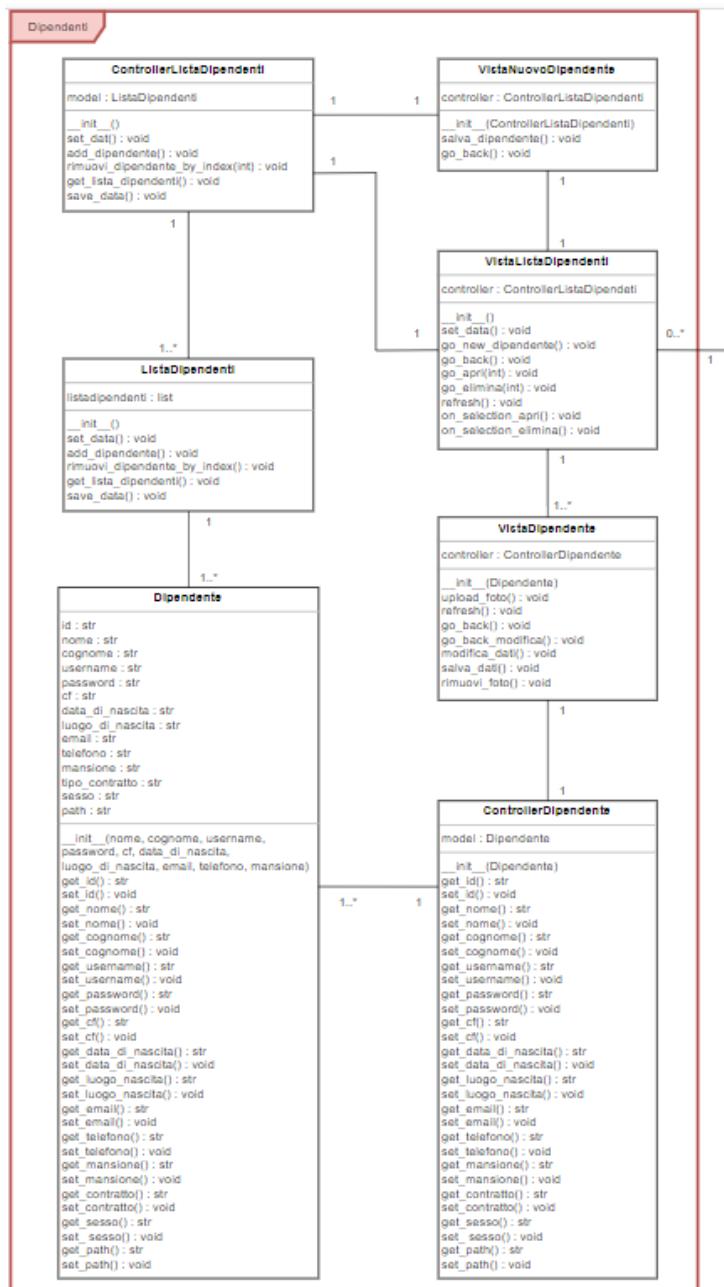


Figura 22: Diagramma delle Classi Dipendenti

## 11.04 DC Documenti

VistaListaDocumenti mostra la lista di tutti i documenti utili alle pratiche dell'autoscuola, ottenuti attraverso il ControllerListaDocumenti. Selezionandone uno è possibile aprirlo, visualizzandolo in VistaDocumento e, tramite un pulsante si può aprire la VistaStampa, avviando così la stampa del documento.

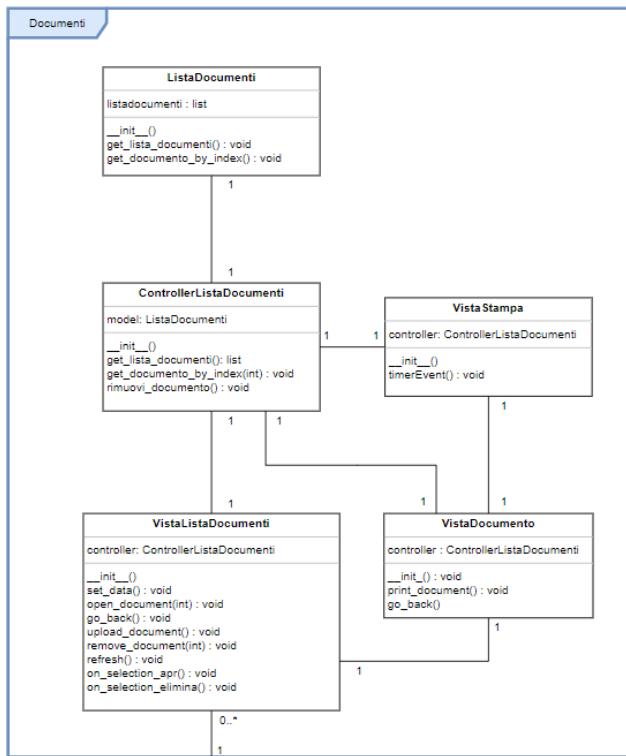


Figura 23: Diagramma delle Classi Documenti

## 11.05 DC Rinnovo

Questo piccolo gruppo di classi gestisce la procedura di rinnovo della patente. All'apertura della VistaRinnovoPatente è possibile inserire le informazioni relative alla patente da rinnovare. Una volta inserite grazie al ControllerListaVisiteMediche è possibile prenotare una visita medica col medico dell'autoscuola e, una volta confermate le informazioni inserite e la prenotazione per la visita, l'utente è rimandato alla VistaRiepilogo che gli permette di effettuare il pagamento. Il pagamento viene poi comunicato alla ListaPagamenti dal ControllerListaPagamenti.

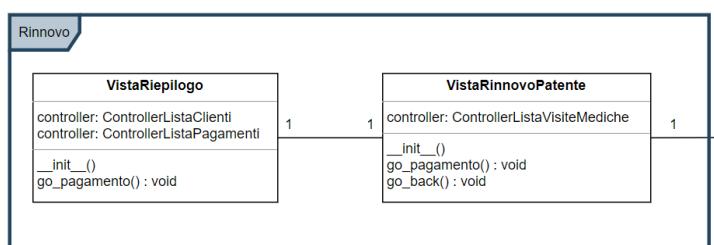


Figura 24: Diagramma delle Classi Rinnovo

## 11.06 DC Pagamento

La VistaListaPagamenti mostra tutti i pagamenti che sono stati effettuati, il loro importo e a quale cliente sono associati. Le informazioni sono comunicate a questa vista dal ControllerListaPagamenti. Per inserire un nuovo pagamento è sufficiente premere il bottone “Nuovo Pagamento” che aprirà la VistaPagamento. Ora si potranno inserire le informazioni relative al pagamento che verranno registrate nel sistema dal ControllerPagamento. Le informazioni sono passate alla VistaRiepilogo che riepiloga i dati del pagamento e gli consente di scegliere il metodo di pagamento. Una volta finalizzato il processo il pagamento è aggiunto alla lista grazie al ControllerListaPagamenti.

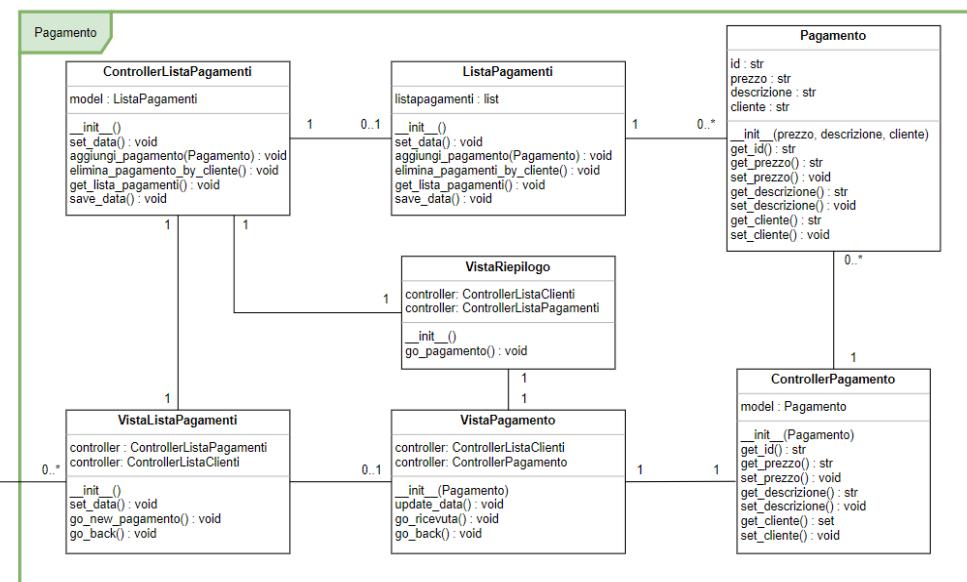


Figura 25: Diagramma delle Classi Pagamenti

## 11.07 DC Esito Esame

All'apertura della VistaInserisciEsitoEsame, grazie al ControllerListaCliente, è possibile selezionare dall'elenco dei clienti che avevano effettuato una prenotazione per un esame. Una volta selezionato si può scegliere l'esito dell'esame, che viene poi registrato tra le informazioni relative al cliente grazie al ControllerCliente.

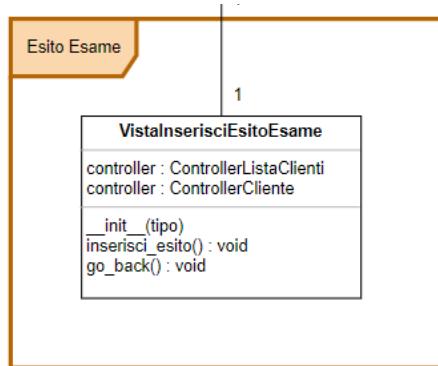


Figura 26: Diagramma delle Classi Esito Esame

## 11.08 DC OrariPrenotazioni

A partire dalla VistaListaPrenotazioni è possibile aggiungere altri orari per cui effettuare prenotazioni. È sufficiente premere il bottone “Aggiungi Orari” l’utente verrà reindirizzato alla VistaAggiungiOrari. In questa vista è possibile scegliere la data per cui si vuole rendere disponibile un nuovo servizio e che tipo di servizio sarà offerto. Tramite il ControllerListaPrenotazioni, se i dati inseriti non corrispondono ad alcun orario già in sistema, il nuovo orario viene aggiunto alla lista degli orari disponibili.

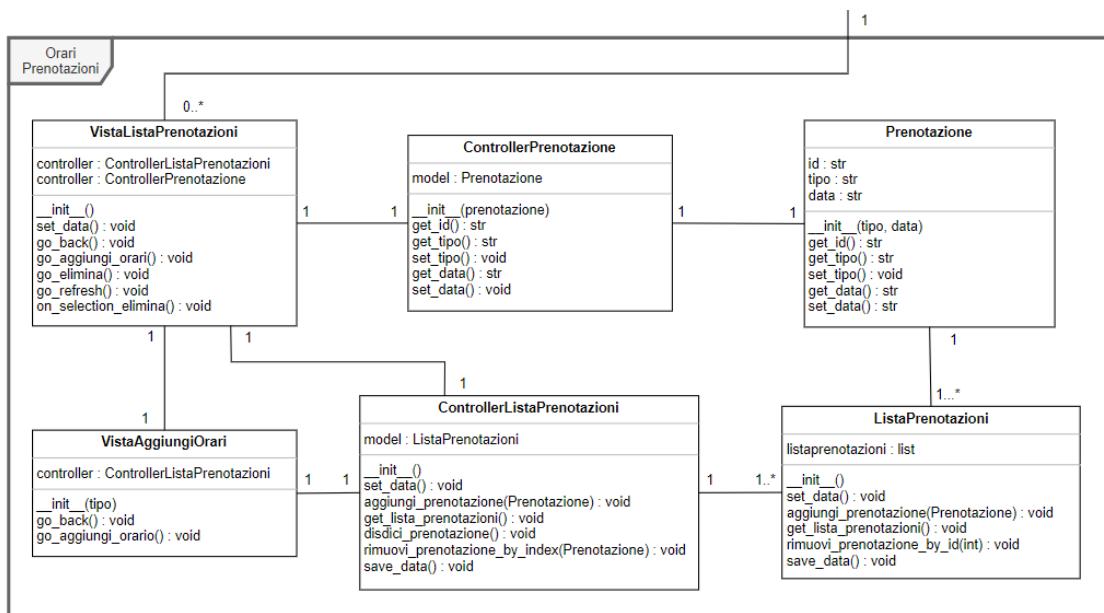


Figura 27: Diagramma delle Classi Orari Prenotazioni

## 11.09 DC DisdettaPrenotazione

La seguente classe ha il compito di gestire la disdetta delle prenotazioni di servizi dell’autoscuola. La VistaDisdettaPrenotazione mostra, grazie al ControllerListaClienti e al ControllerCliente, l’elenco dei clienti con prenotazioni a loro carico. Una volta scelta la prenotazione da cancellare e confermata la decisione, l’eliminazione verrà effettuata e la modifica verrà salvata grazie al ControllerListaClienti.

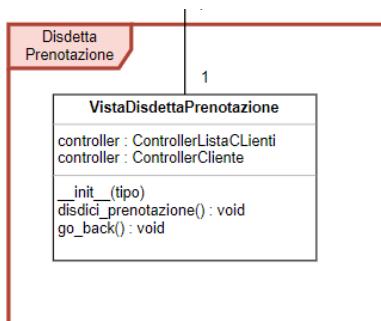


Figura 28: Diagramma delle Classi Disdetta Prenotazione

## 11.10 DC NuovaPrenotazione

All'apertura della VistaPrenotazione è possibile scegliere, grazie al ControllerListaClienti, il cliente per cui effettuare la nuova prenotazione. Dal ControllerListaPrenotazioni si può scegliere la data del servizio per cui il cliente vuole prenotarsi. Una volta confermata la prenotazione essa viene aggiunta alla lista sempre grazie al ControllerListaPrenotazioni.

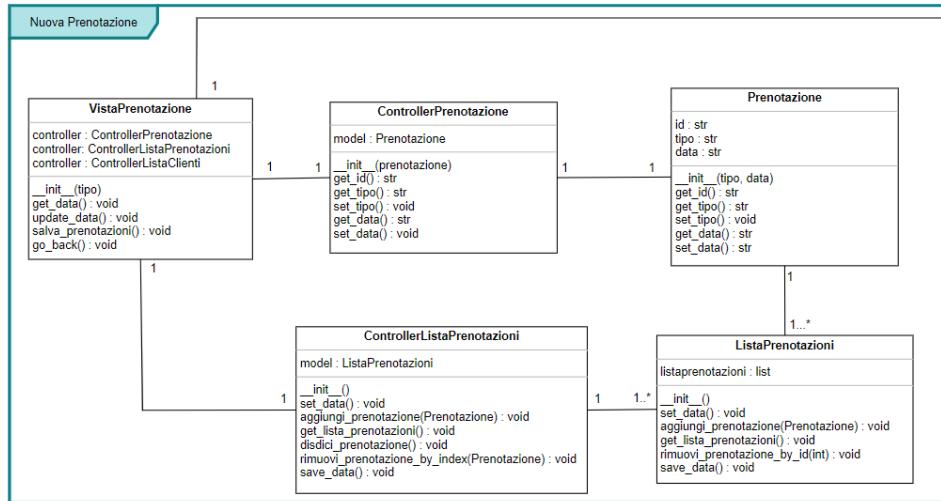


Figura 29: Diagramma delle Classi Nuova Prenotazione

## 11.11 DC VisitaMedica

Questo gruppo di classi permette visualizzare la lista di prenotazioni per le visite mediche e di effettuarne di nuove. VistaListaVisiteMediche mostra appunto la lista delle visite che sono state già prenotate. Tramite un pulsante è possibile aprire la VistaPrenotaVisitaMedica che permette di scegliere un giorno nel quale effettuare una visita medica. Le informazioni relative alla prenotazione sono registrare nel sistema grazie al ControllerVisitaMedica e la prenotazione viene poi aggiunta alla lista grazie al ControllerListaVisiteMediche.

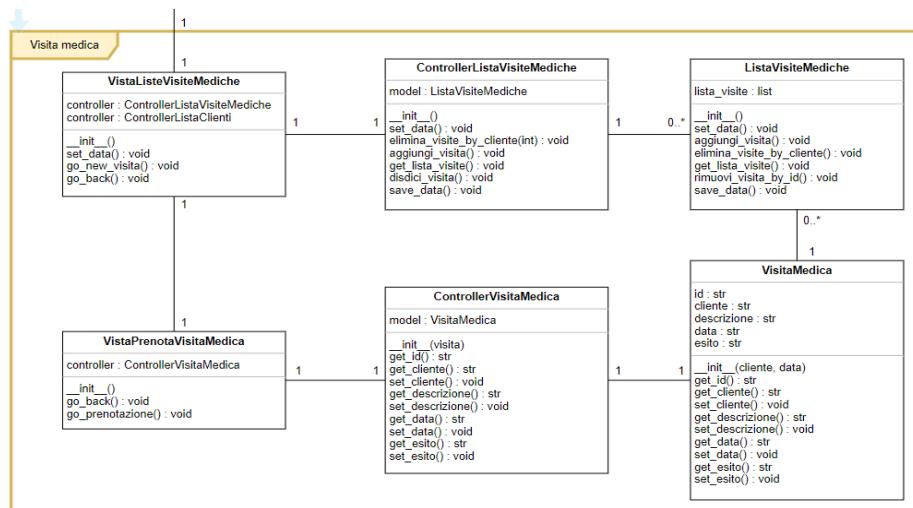


Figura 30: Diagramma delle Classi Visite Mediche

## 11.12 DC EsitoVisitaMedica

Una volta aperta la VistaInserisciEsitoVisita il medico, grazie al ControllerListaVisiteMediche, può scegliere il cliente a cui inserire l'esito. Una volta selezionato viene aggiunto il responso del medico e una breve descrizione, che viene registrato tra i dati del rispettivo cliente grazie al ControllerListaClienti.

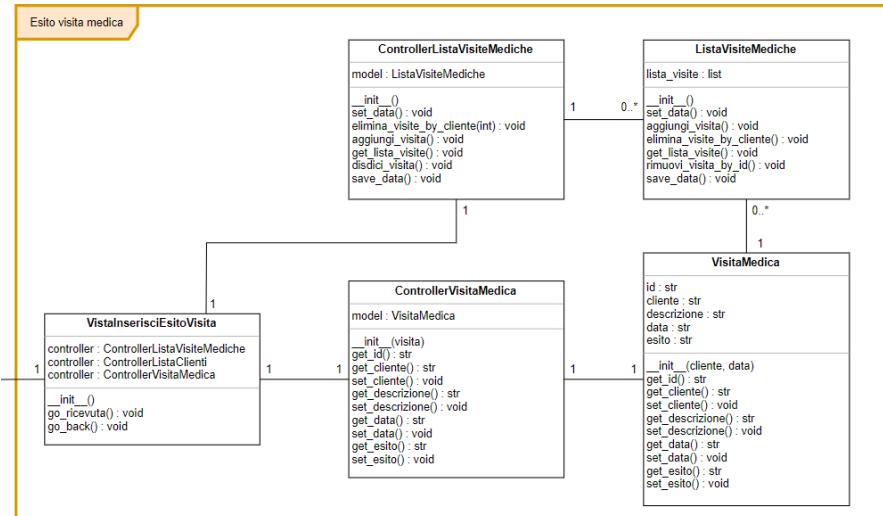


Figura 31: Diagramma delle Classi Esito Visita Medica

## 11.13 DC DisdiciVisitaMedica

Queste classi consentono effettuare la procedura di disdetta di una visita medica precedentemente fissata. Grazie al ControllerListaVisiteMediche, nella VistaDisdiciVisitaMedica è possibile visualizzare l'elenco delle visite mediche che è possibile disdire. Una volta selezionata la visita da disdire la visita viene cancellata e, grazie al ControllerListaVisiteMediche, le modifiche vengono salvate.

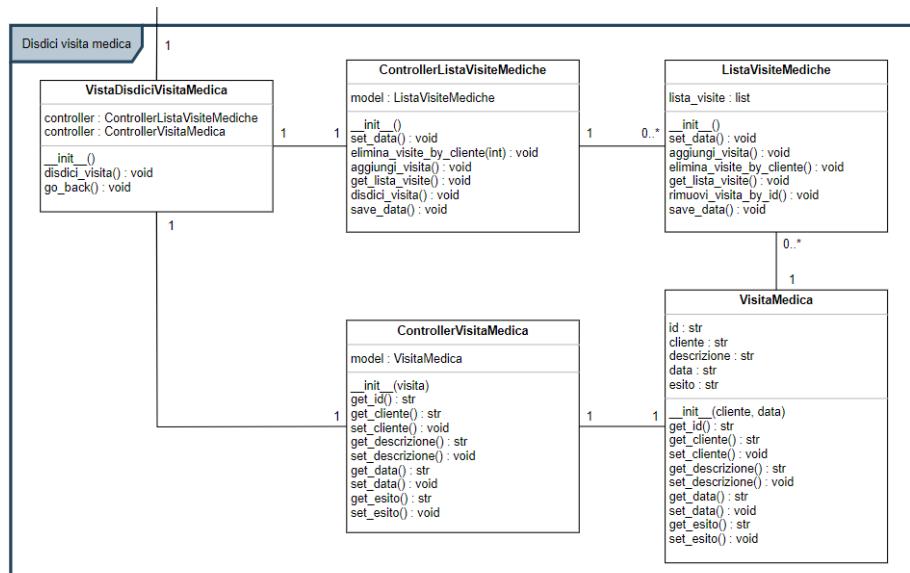


Figura 32: Diagramma delle Classi Disdici Visita Medica

## 12. Diagrammi di Sequenza ed Attività

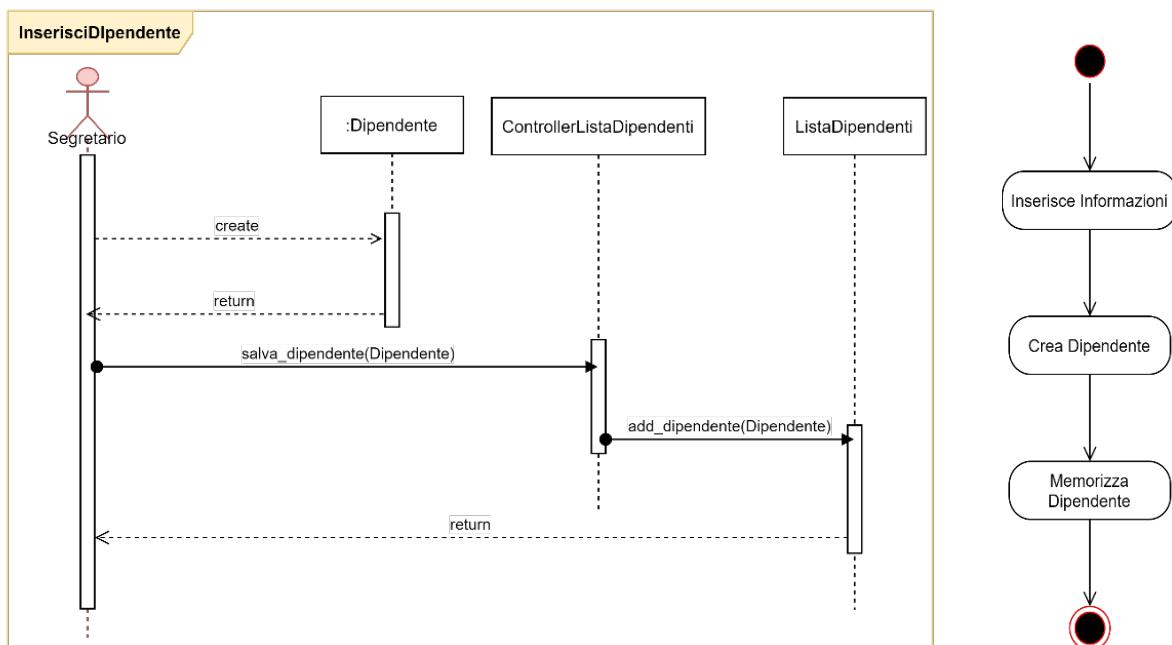
I diagrammi seguenti sono i diagrammi di sequenza e di attività associati ad ogni caso d'uso previsto per il nostro software.

I diagrammi di sequenza sono utili a mostrare le interazioni tra gli attori, il sistema e i componenti del sistema stesso. Sono chiamati in questo modo perché mostrano appunto la sequenza con la quale i vari componenti interagiscono tra loro, andando dall'alto verso il basso.

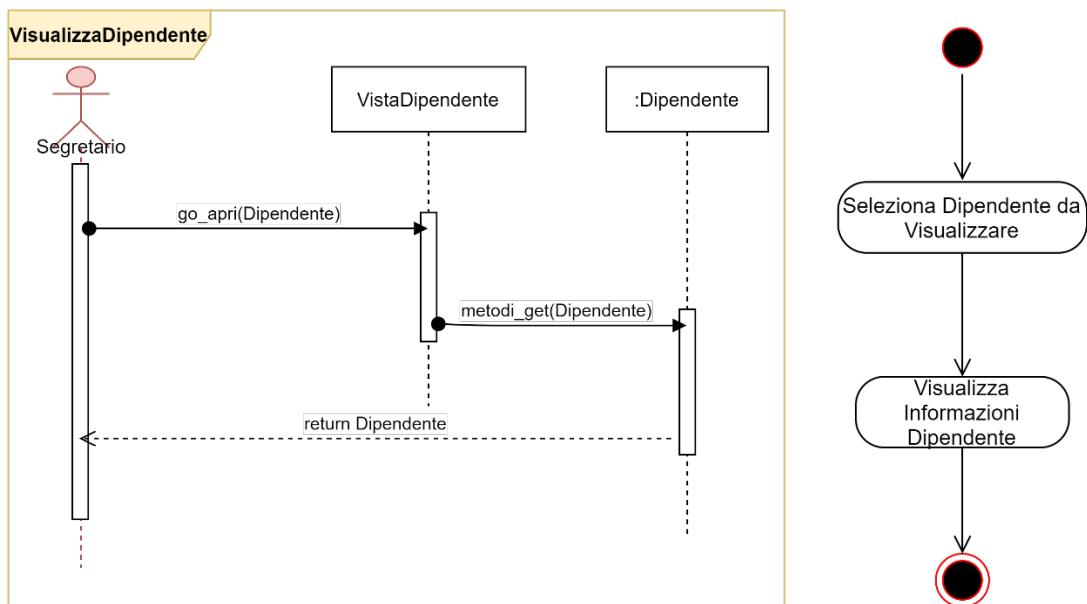
I diagrammi di attività hanno lo scopo di mostrare le attività coinvolte nel processo e i processi aziendali in cui il sistema viene usato.

### 12.1 UC: Gestione Dipendenti

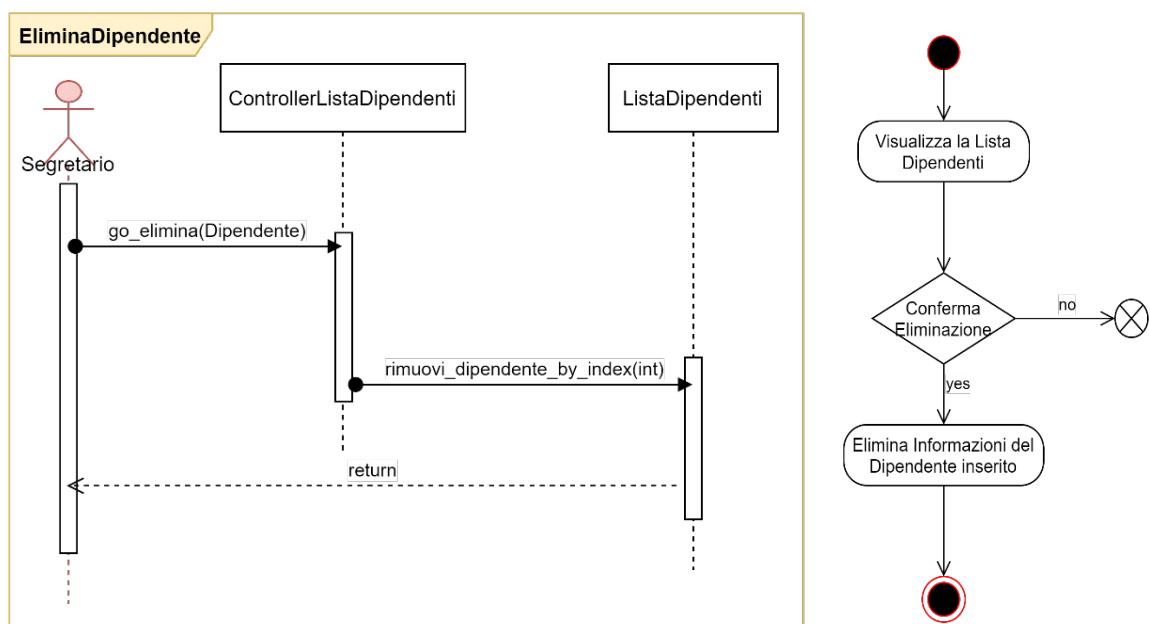
#### 1) InserisciDipendente



## 2) VisualizzaDipendente

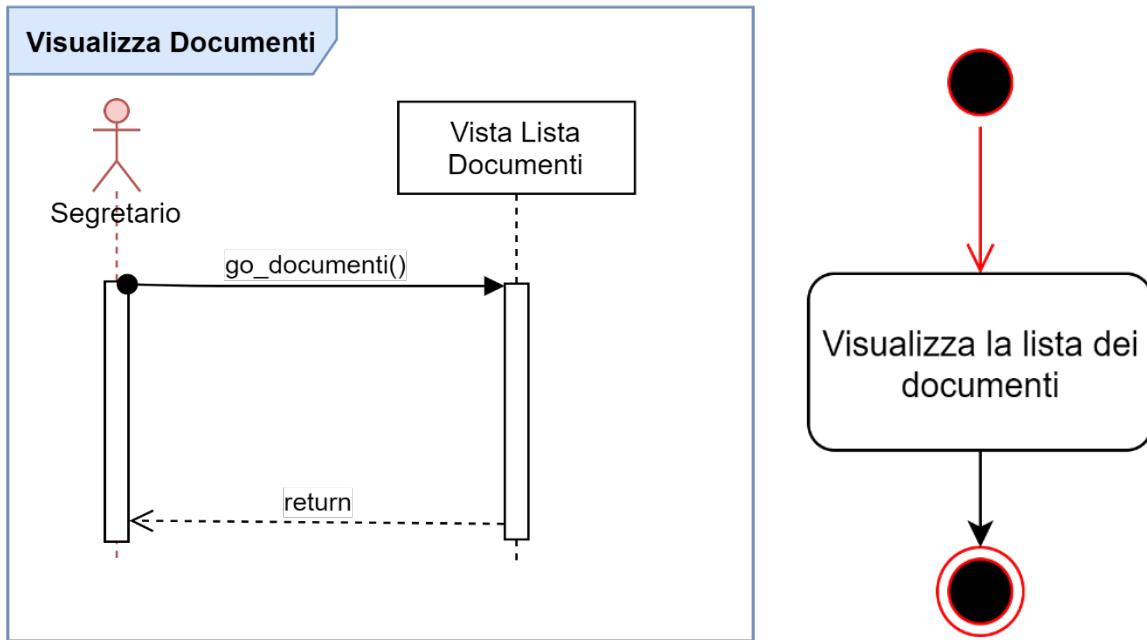


## 3) EliminaDipendente

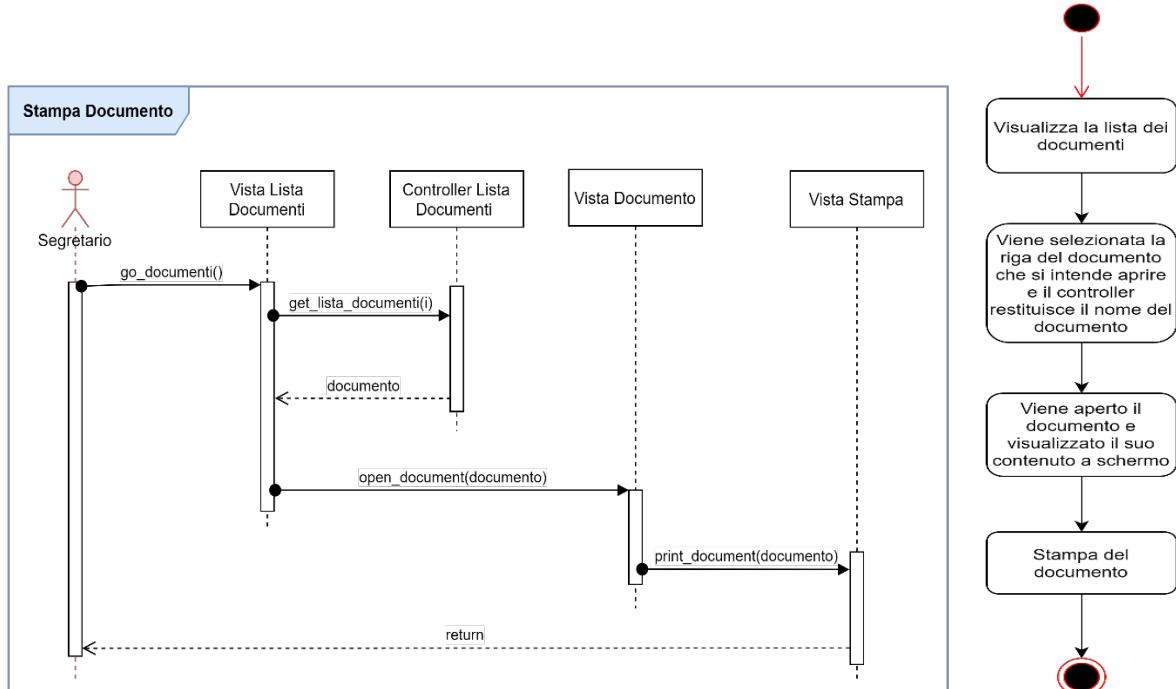


## 12.2UC: Visualizza Documenti

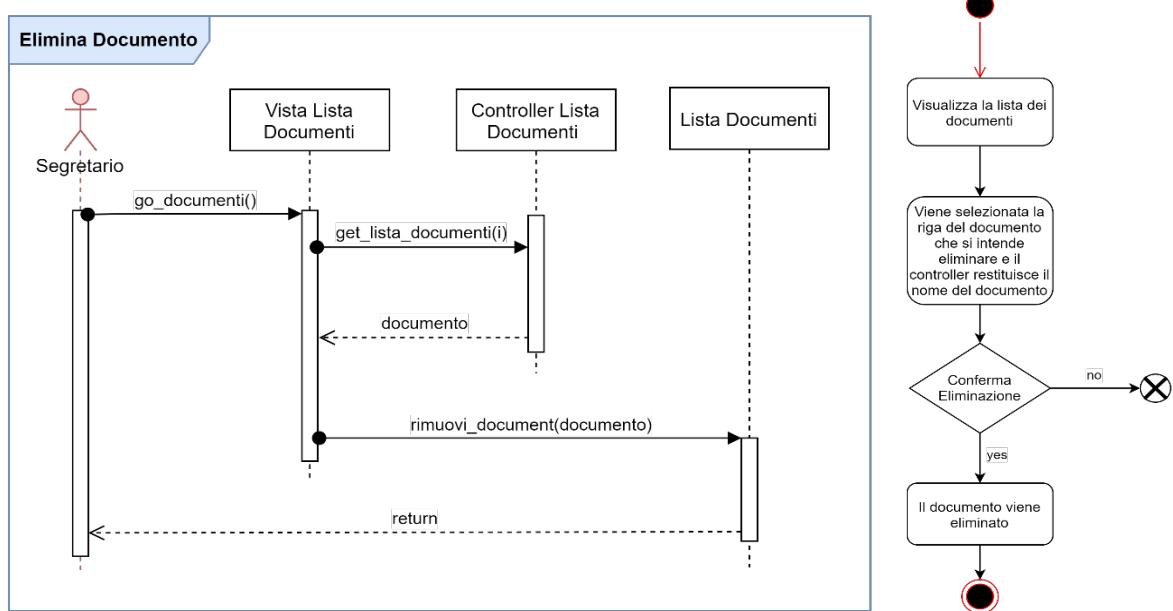
### 1) VisualizzaDocumenti



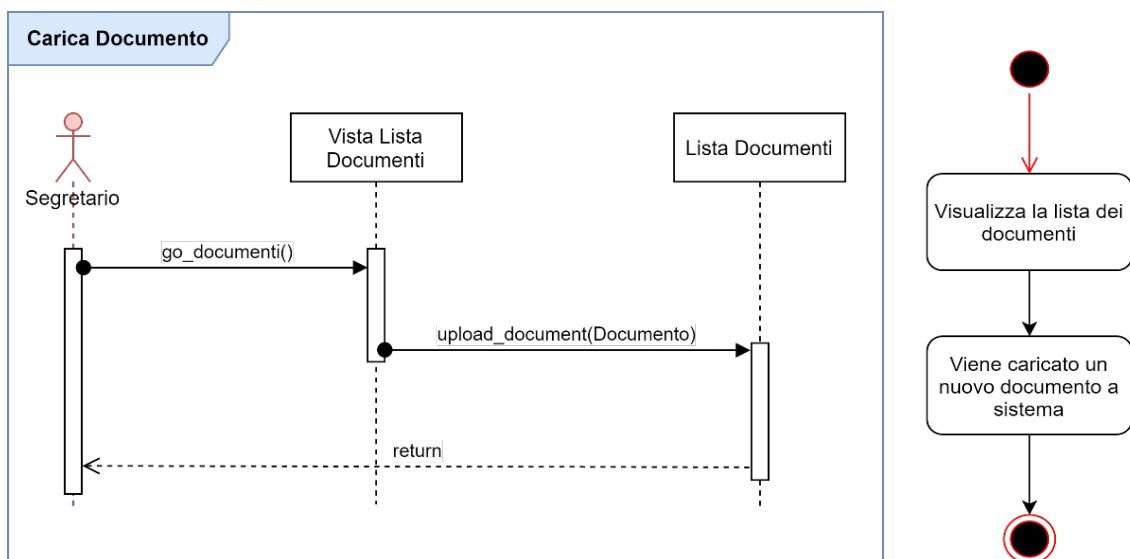
### 2) StampaDocumento



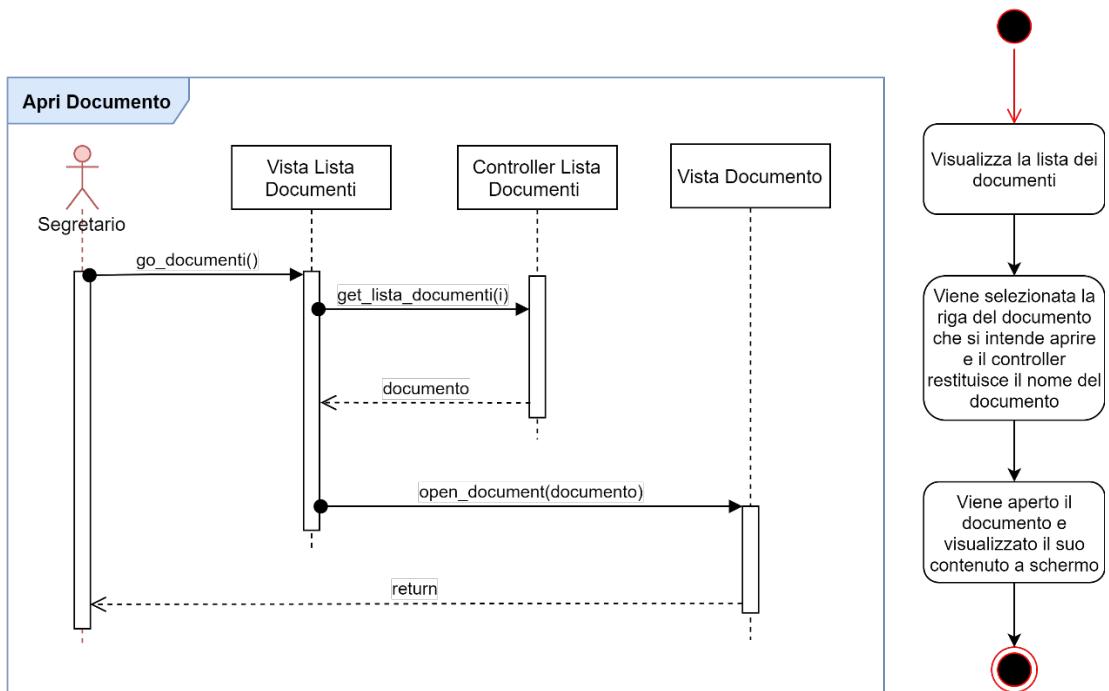
### 3) EliminaDocumento



### 4) CaricaDocumento

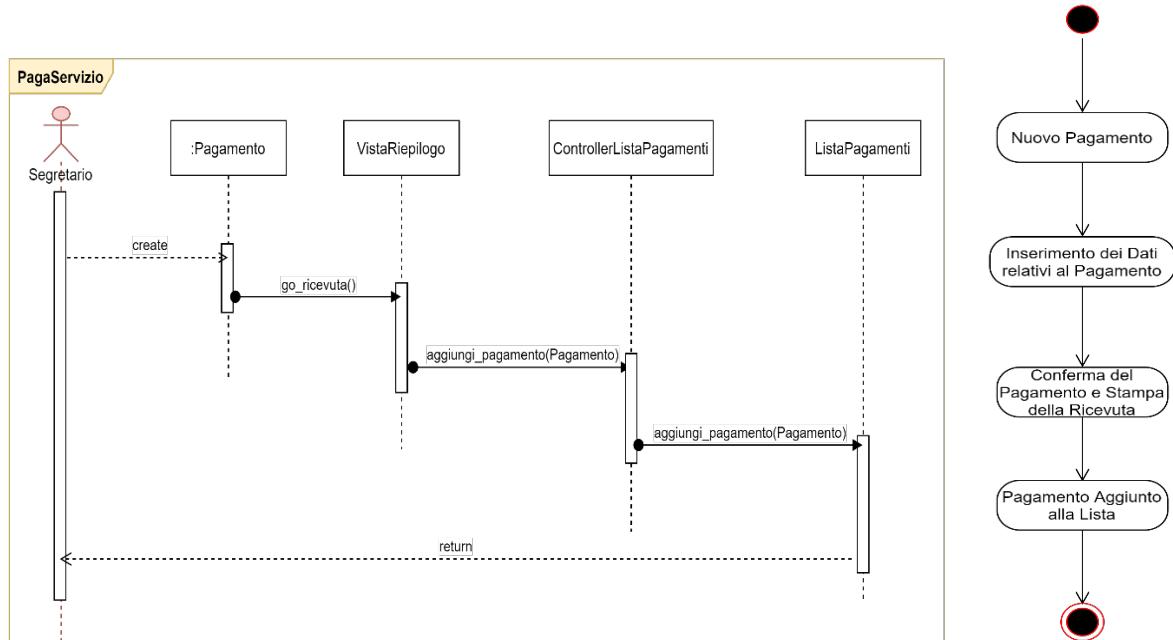


## 5) ApriDocumento

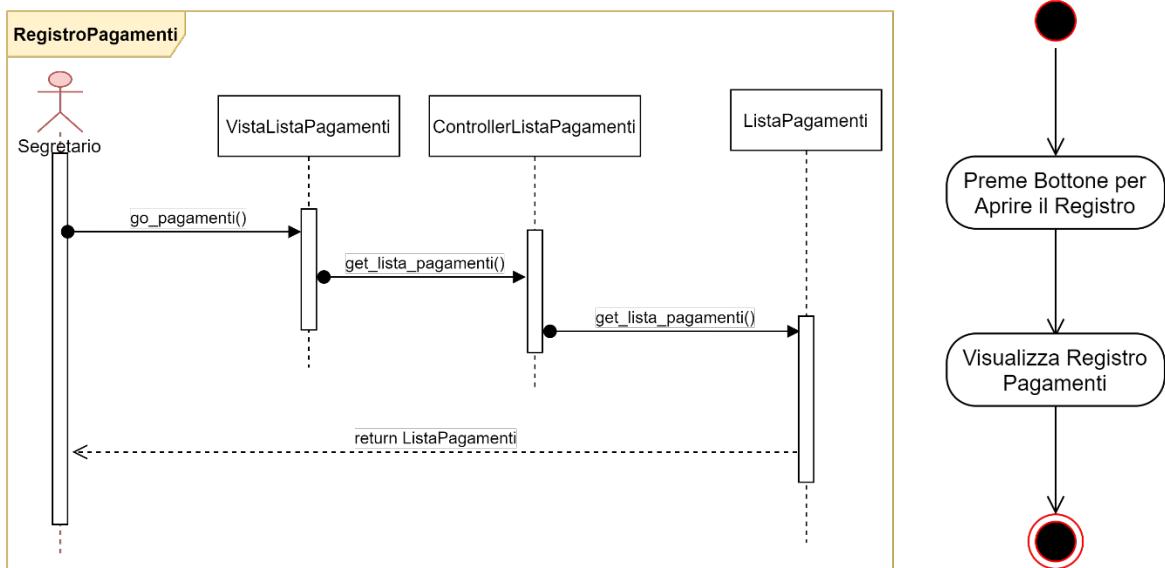


## 12.3 UC: Gestione Pagamenti

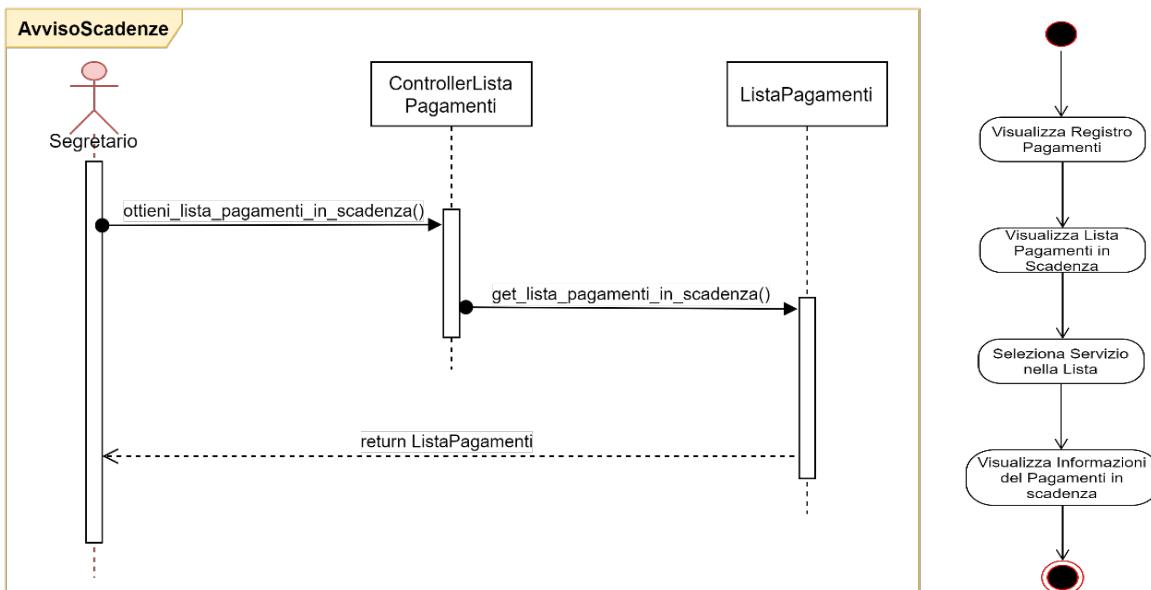
### 1) PagaServizio



### 2) RegistroPagamenti

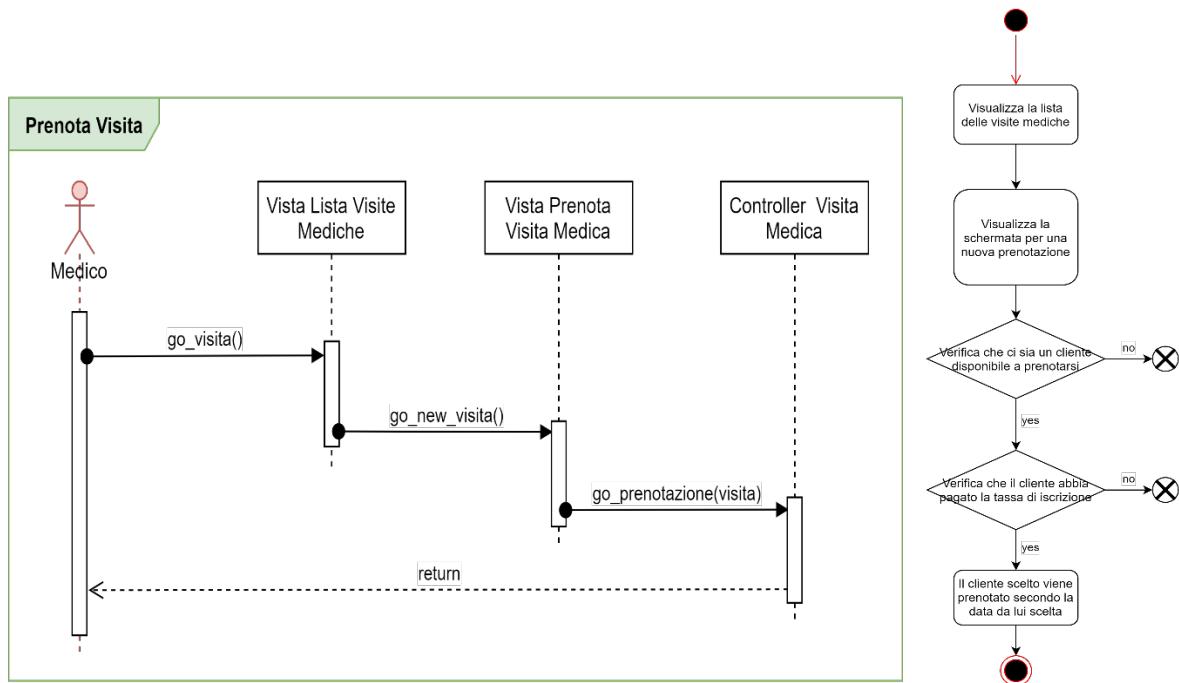


### 3) AvvisoScadenze

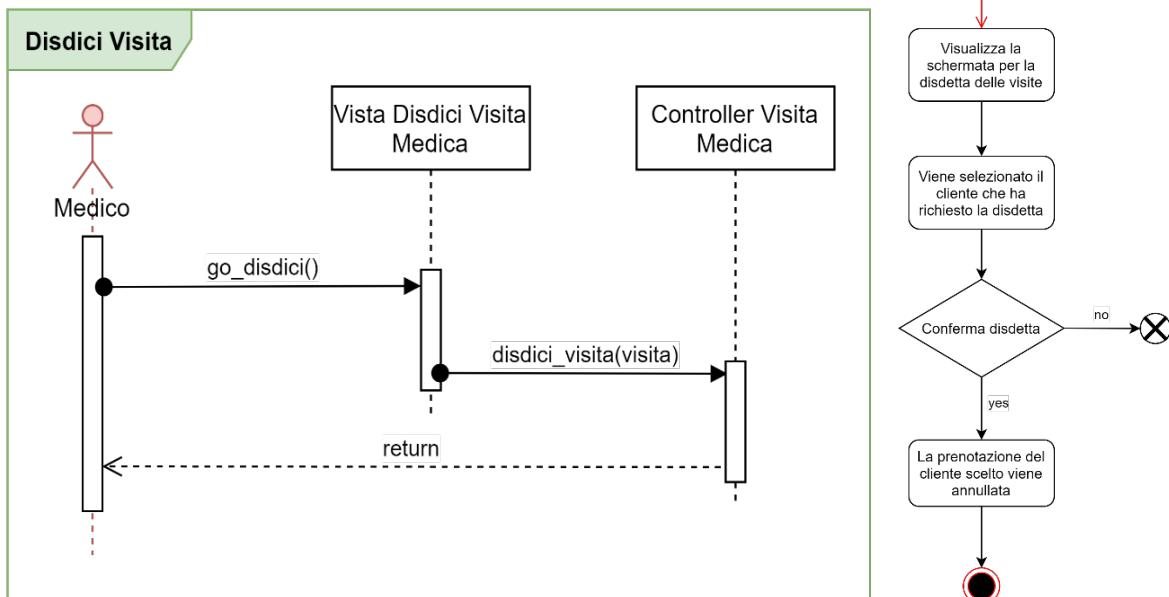


## 12.4 UC: Gestione Visite Mediche

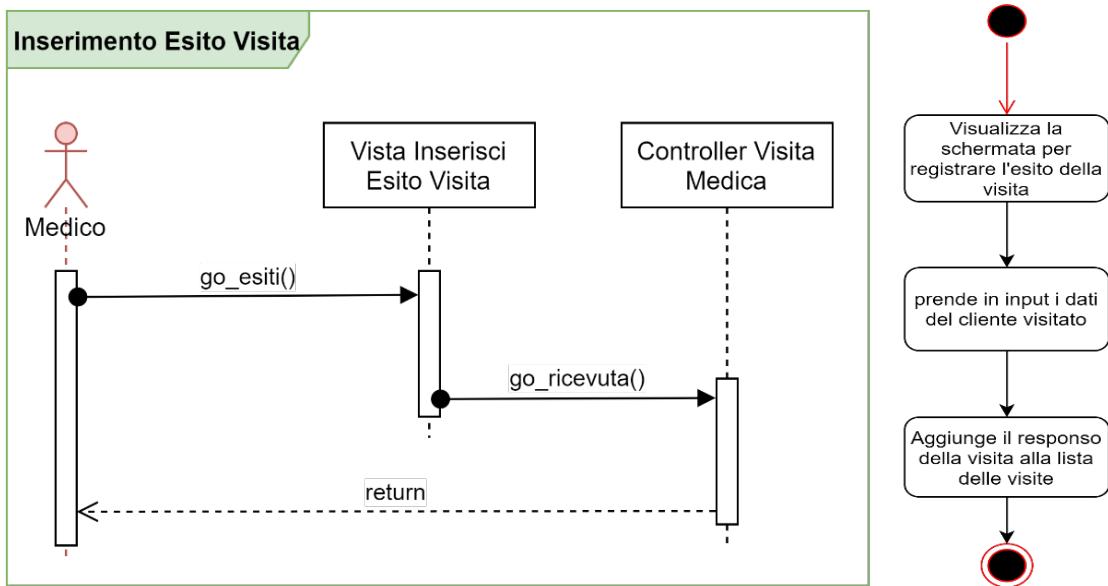
### 1) PrenotaVisita



### 2) DisdiciVisita

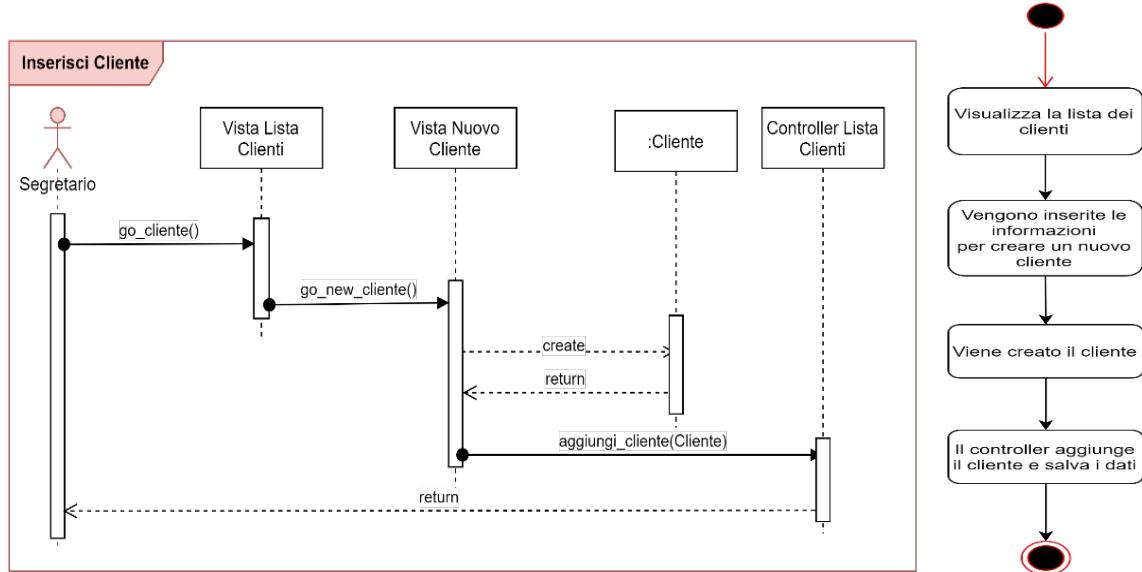


### 3) InserimentoEsitoVisita

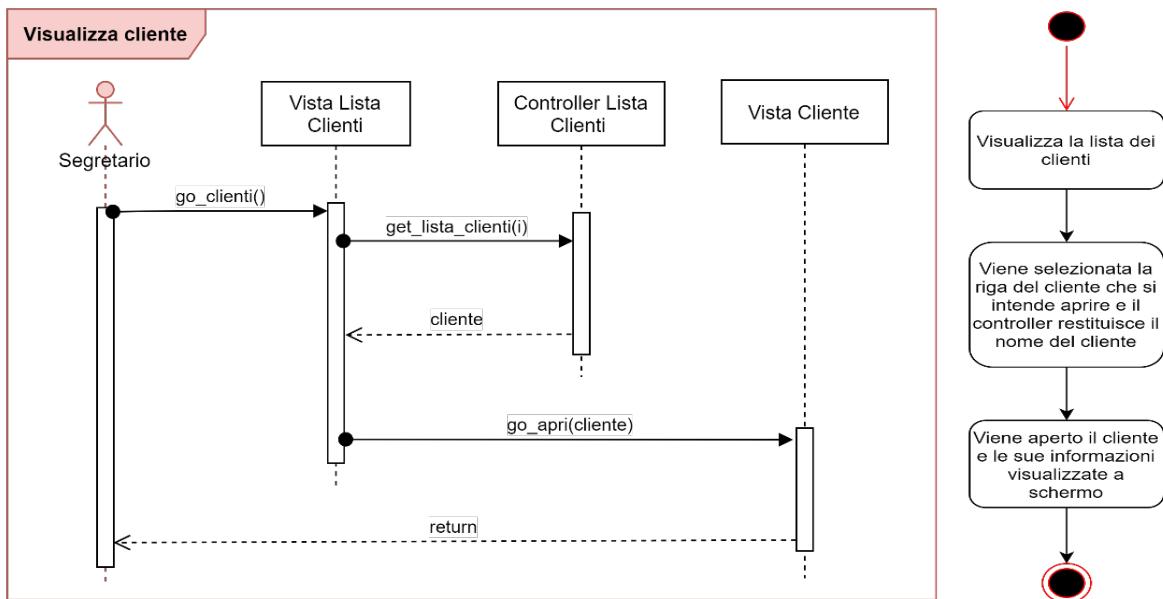


## 12.5 UC: Gestione Clienti

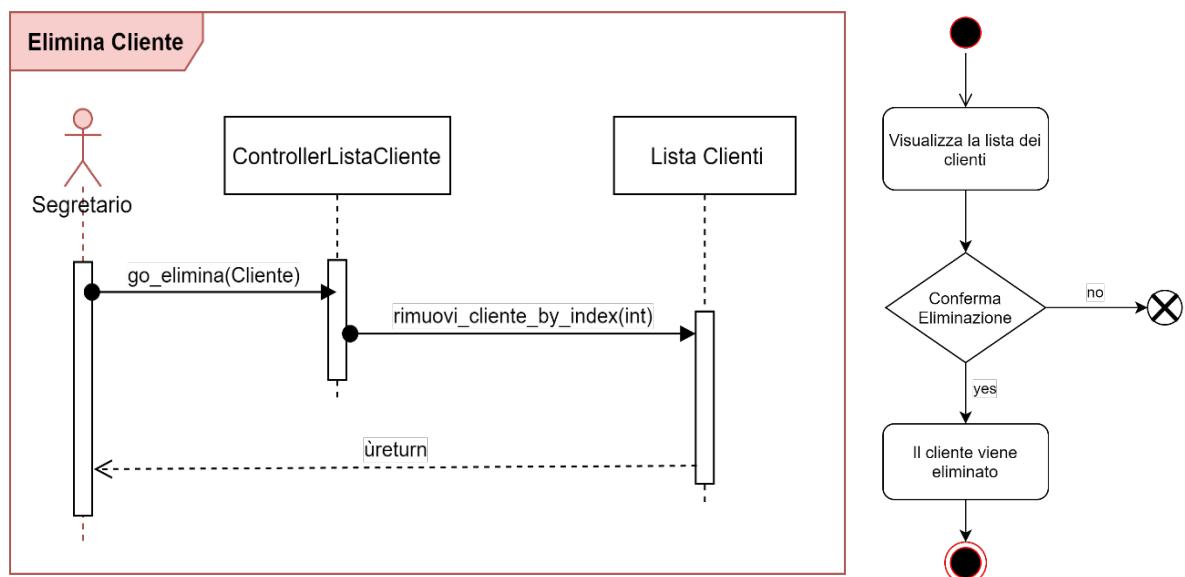
### 1) InserisciCliente



### 2) VisualizzaCliente

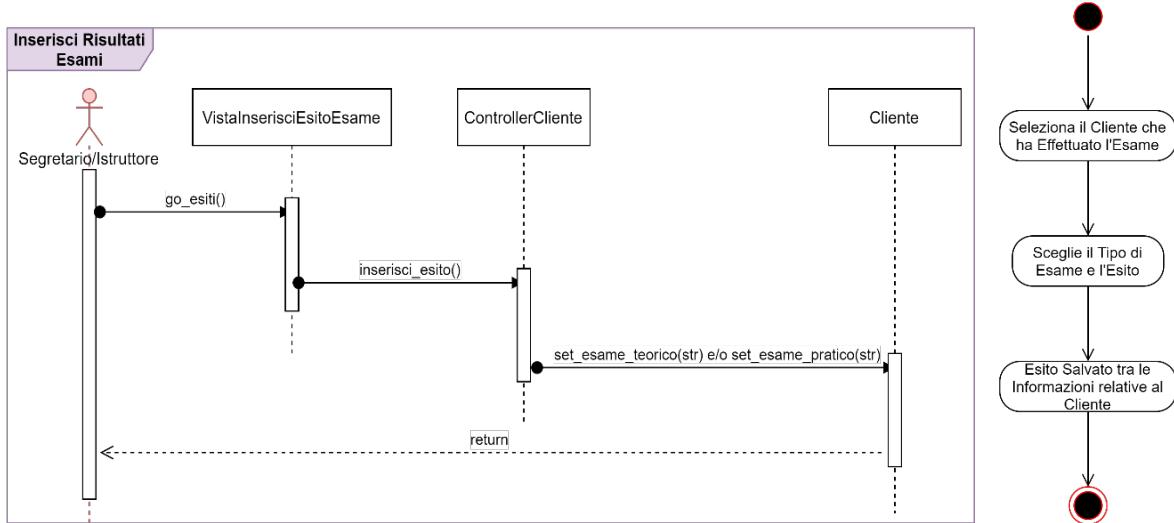


### 3) EliminaCliente

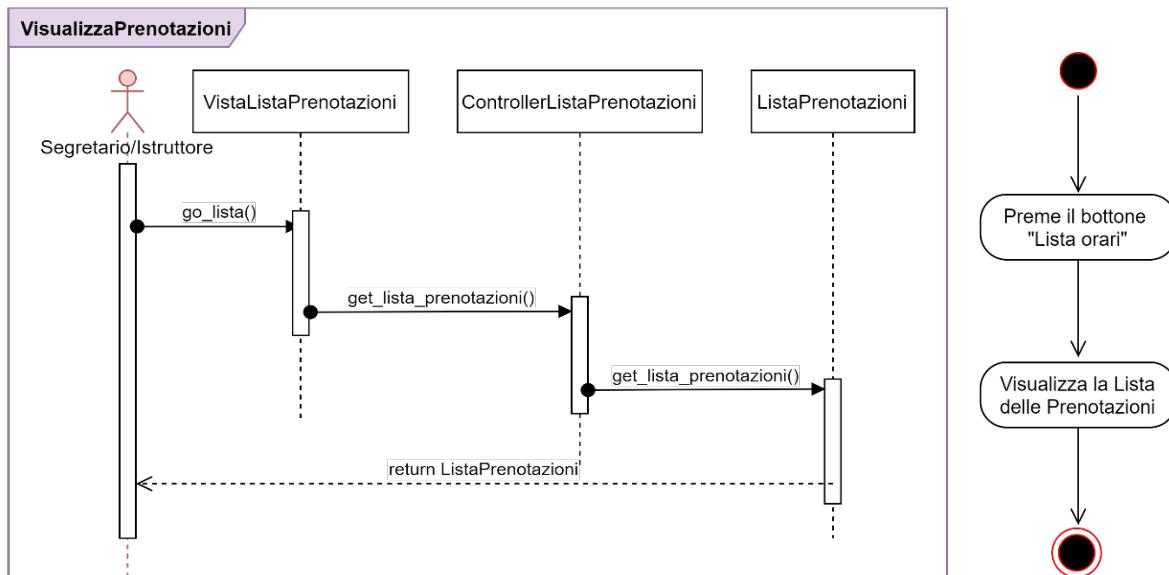


## 12.6 UC: Gestione Attività

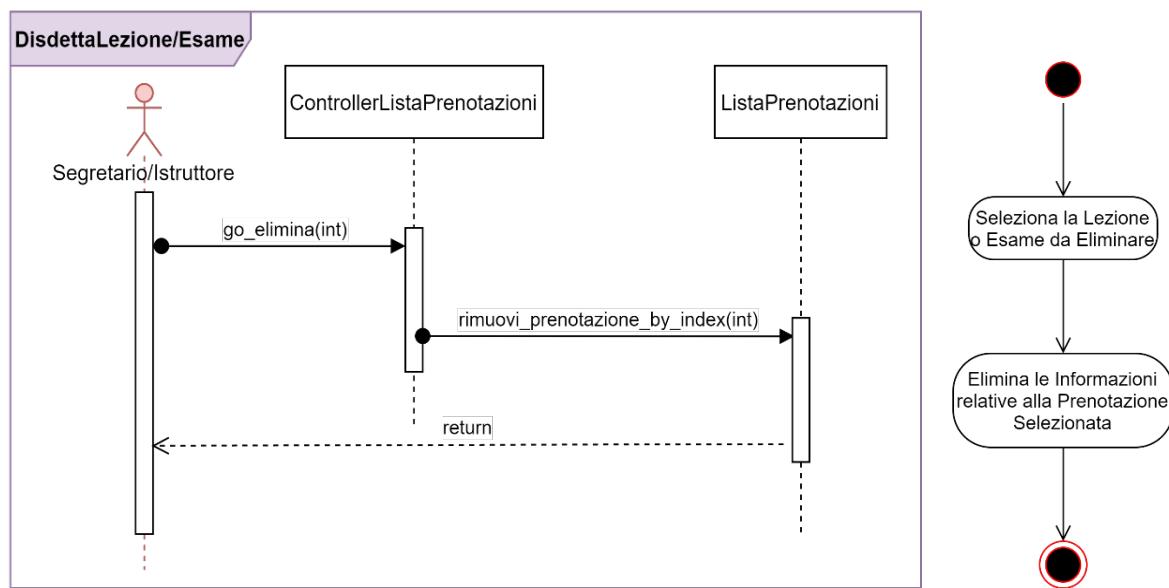
### 1) InserisciRisultatiEsami



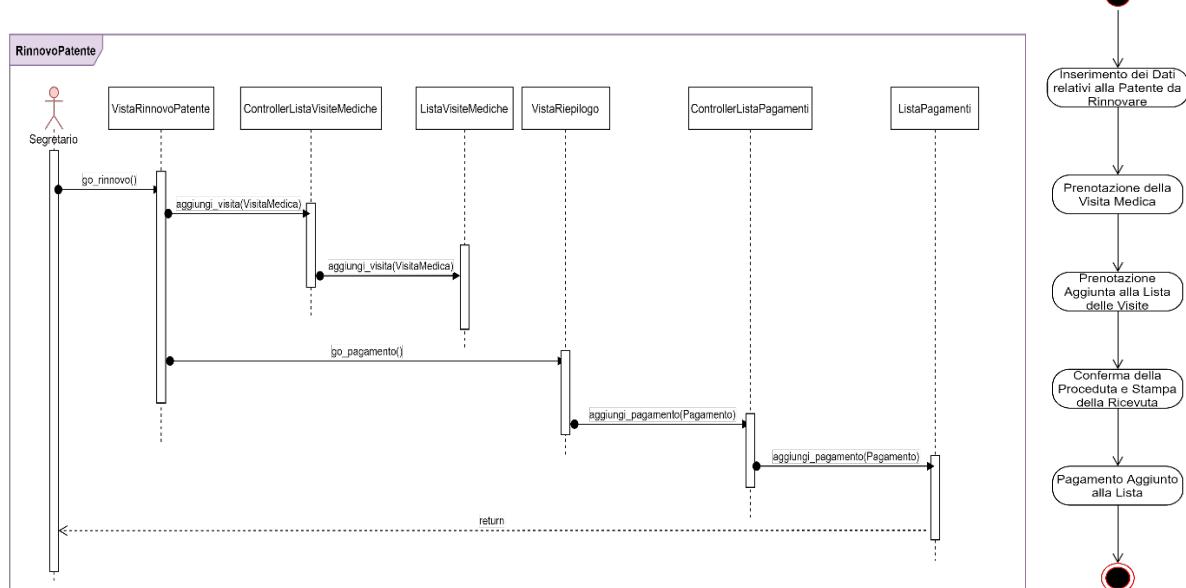
### 2) VisualizzaPrenotazioni



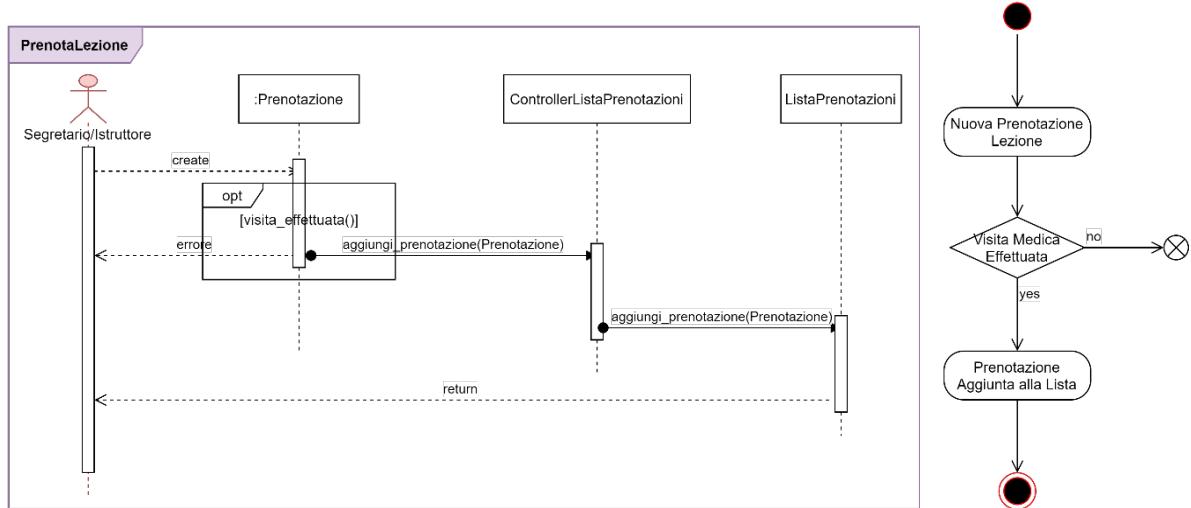
### 3) DisdettaLezione/Esame



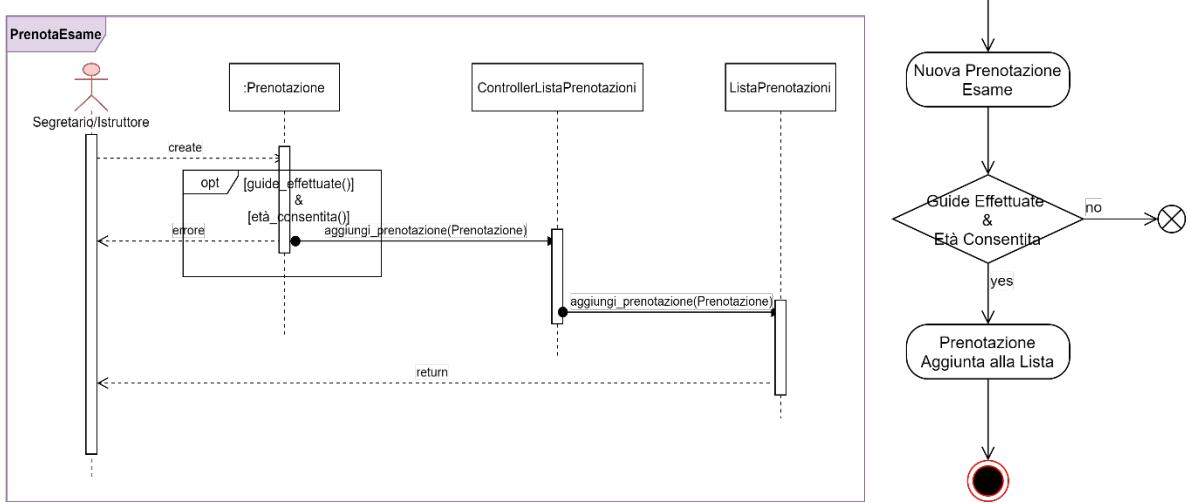
### 4) RinnovoPatente



## 5) PrenotaLezione



## 6) PrenotaEsame



## 13. Mock-up

Un mockup è una realizzazione a scopo illustrativo o meramente espositivo di un oggetto o un sistema, senza le complete funzioni dell'originale; può rappresentare la totalità o solo una parte dell'originale di riferimento. Nelle pratiche dell'ingegneria del software vengono utilizzati per dare un'idea al cliente su quale sarà il risultato del processo software.

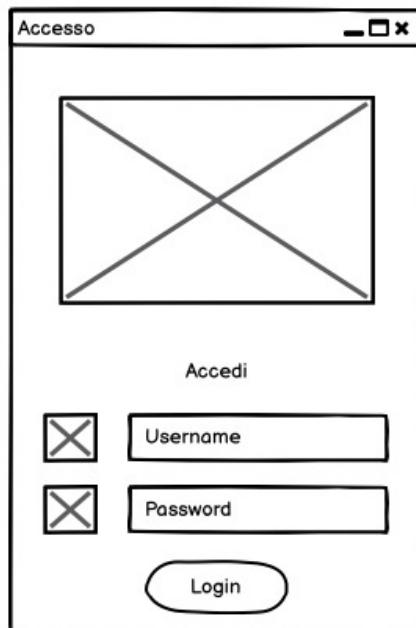


Figura 33: Vista accesso

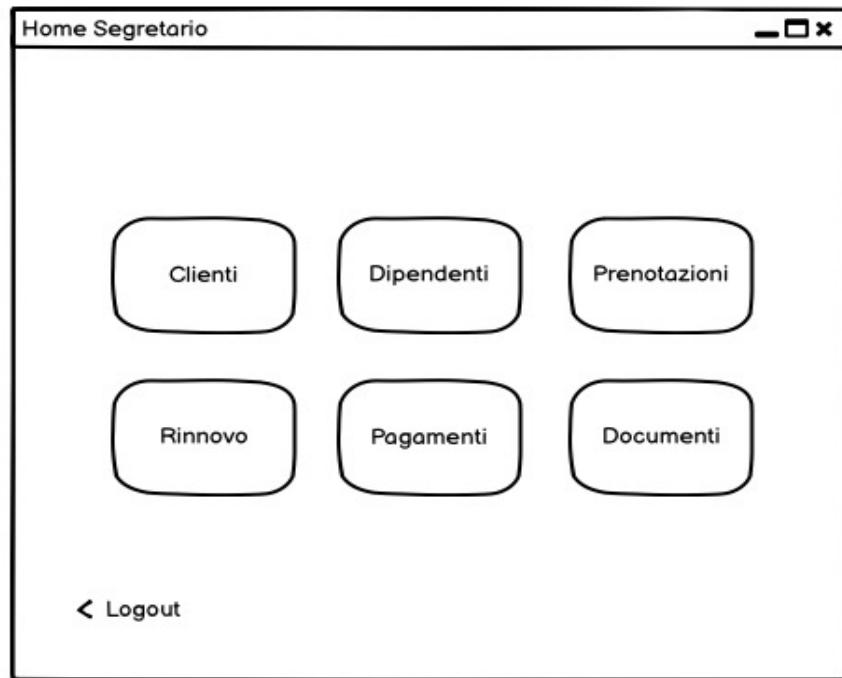


Figura 34: Vista home segretario

### Vista HomeIstruttore/Vista Menù Prenotazioni

Questa vista può essere aperta anche dal segretario tramite il bottone “Prenotazioni”. In questo caso la finestra sarà visualizzata come “Menù Prenotazioni” e non come “Home Istruttore”.

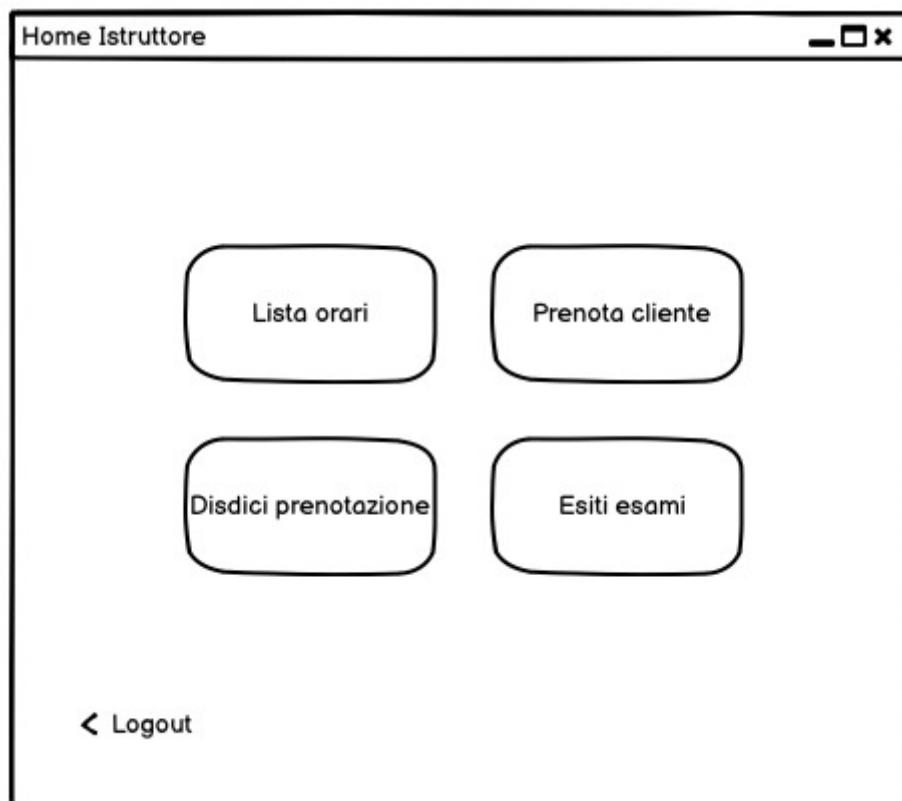


Figura 35: Vista home segretario

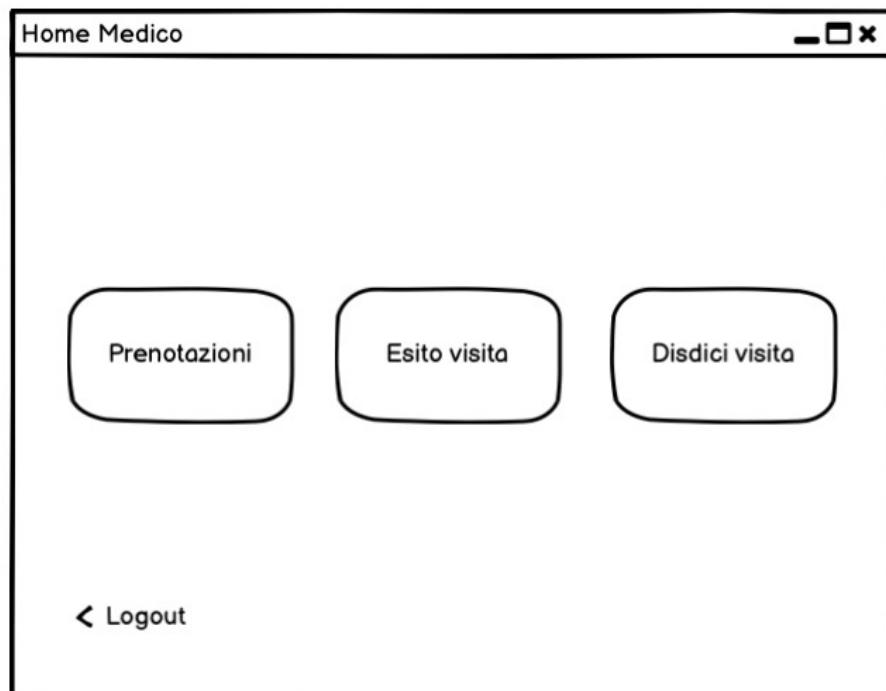


Figura 36: Vista home medico

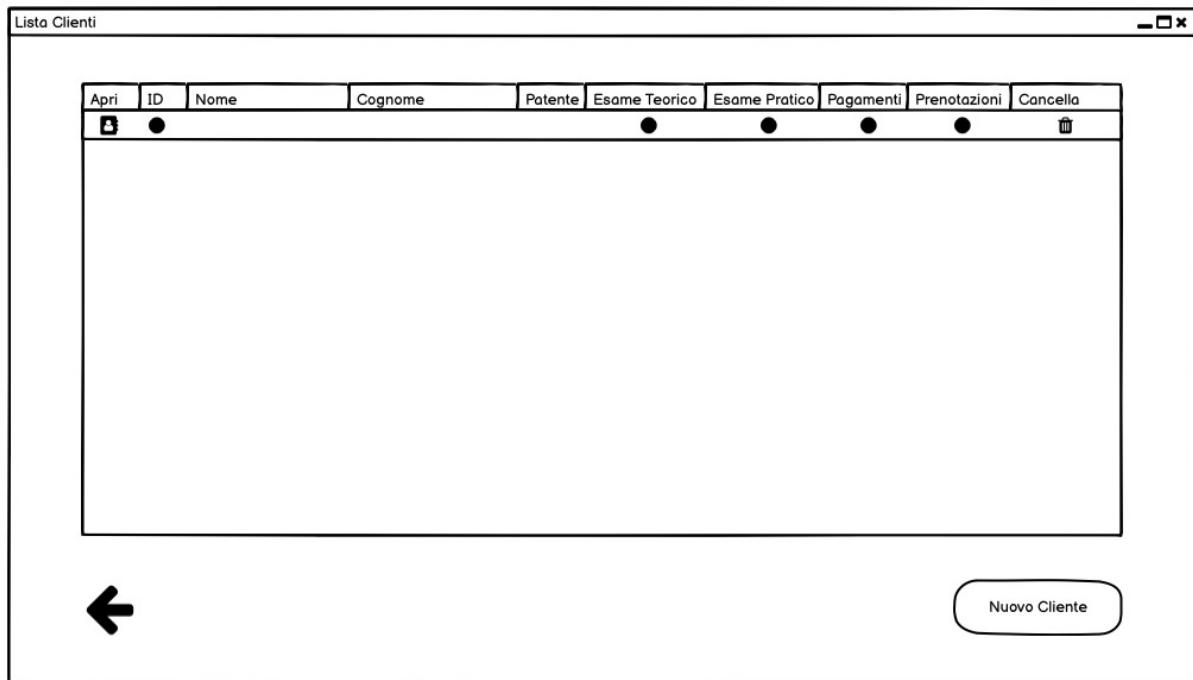


Figura 37: Vista lista clienti

The screenshot shows a software application window titled 'Info Cliente'. On the left side, there is a vertical list of fields: 'Fototessera', 'Nome', 'Cognome', 'Tipo patente', 'Codice Fiscale', 'Indirizzo', 'Email', 'Telefono', 'Data di nascita', and 'Sesso'. To the right of each field is a rectangular input box. Above the 'Fototessera' field, there is a small icon of a camera with a pencil next to it, and below it is a trash can icon. To the right of the input boxes for 'Nome', 'Cognome', 'Tipo patente', 'Codice Fiscale', 'Indirizzo', 'Email', 'Telefono', 'Data di nascita', and 'Sesso' are small icons of a camera with a pencil and a trash can. In the bottom right corner of the window frame, there is a button labeled 'Modifica dati' (Modify data). On the left side of the window, there is a large black arrow pointing to the left.

Figura 38: Vista cliente

Lista Dipendenti

Apri	ID	Nome	Cognome	Mansione	Cancella

Nuovo Dipendente

Figura 39: Vista lista dipendenti

Info Dipendente

Fototessera	
Nome	<input type="text"/>
Cognome	<input type="text"/>
Codice Fiscale	<input type="text"/>
Luogo di nascita	<input type="text"/>
Email	<input type="text"/>
Telefono	<input type="text"/>
Data di nascita	<input type="text"/>
Tipo contratto	<input type="text"/>
Mansione	<input type="text"/>
Sesso	<input type="text"/>

Modifica dati

Figura 40: Vista dipendente

Nuovo Cliente

Nome	<input type="text"/>	Data di nascita	<input type="button" value="CANCEL"/> <input type="button" value="OK"/>	
Cognome	<input type="text"/>			
Codice Fiscale	<input type="text"/>			
Indirizzo	<input type="text"/>			
Email	<input type="text"/>			
Telefono	<input type="text"/>			
Tipo patente	<input type="text"/>	<input type="radio"/> Uomo <input type="radio"/> Donna		

**←** **Salva**

Figura 41: Vista inserisci nuovo cliente

Nuovo Dipendente

Nome	<input type="text"/>	Data di nascita	<input type="button" value="CANCEL"/> <input type="button" value="OK"/>	
Cognome	<input type="text"/>			
Codice Fiscale	<input type="text"/>			
Luogo di nascita	<input type="text"/>			
Email	<input type="text"/>			
Telefono	<input type="text"/>			
<input type="radio"/> Uomo	<input type="radio"/> Donna	Mansione	<input type="text"/>	
Tipo di contratto			<input type="text"/>	
Username	<input type="text"/>	Password	<input type="text"/>	
Conferma password			<input type="text"/>	

**←** **Salva**

Figura 42: Vista inserisci nuovo dipendente

Lista Prenotazioni

ID	Tipo	Data	Elimina

← Aggiungi orari

Figura 43: Vista lista prenotazioni

Nuova Prenotazione

Cliente	<input type="text"/>	Data	<input type="text"/>
Tipo	<input type="text"/>		

← Salva

Figura 44: Vista inserisci nuova prenotazione

Disdici Prenotazione

Seleziona il cliente

▼

←

Disdici

This screenshot shows a window titled "Disdici Prenotazione". Inside, there is a label "Seleziona il cliente" followed by a dropdown menu icon. At the bottom right is a rounded rectangular button labeled "Disdici". A large black arrow pointing left is located at the bottom left.

Figura 45: Vista disdici prenotazione

Esito Esame

Seleziona il cliente

▼

Esito

▼

←

Inserisci Esito

This screenshot shows a window titled "Esito Esame". It contains two dropdown menu icons, one labeled "Seleziona il cliente" and another labeled "Esito". At the bottom right is a rounded rectangular button labeled "Inserisci Esito". A large black arrow pointing left is located at the bottom left.

Figura 46: Vista inserisci esito esame

Rinnovo Patente

Nome

Cognome

Numero patente

Visita medica

Sat, Jul 03

< JULY 2021 >

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31

CANCEL OK

←

Procedi al pagamento

This screenshot shows a window titled "Rinnovo Patente". It has input fields for "Nome", "Cognome", and "Numero patente". To the right is a calendar for July 2021, showing the date "Sat, Jul 03" highlighted. At the bottom right is a rounded rectangular button labeled "Procedi al pagamento". A large black arrow pointing left is located at the bottom left.

Figura 47: Vista rinnovo patente

Lista Pagamenti

ID	Prezzo	Descrizione	Cliente

← Nuovo Pagamento

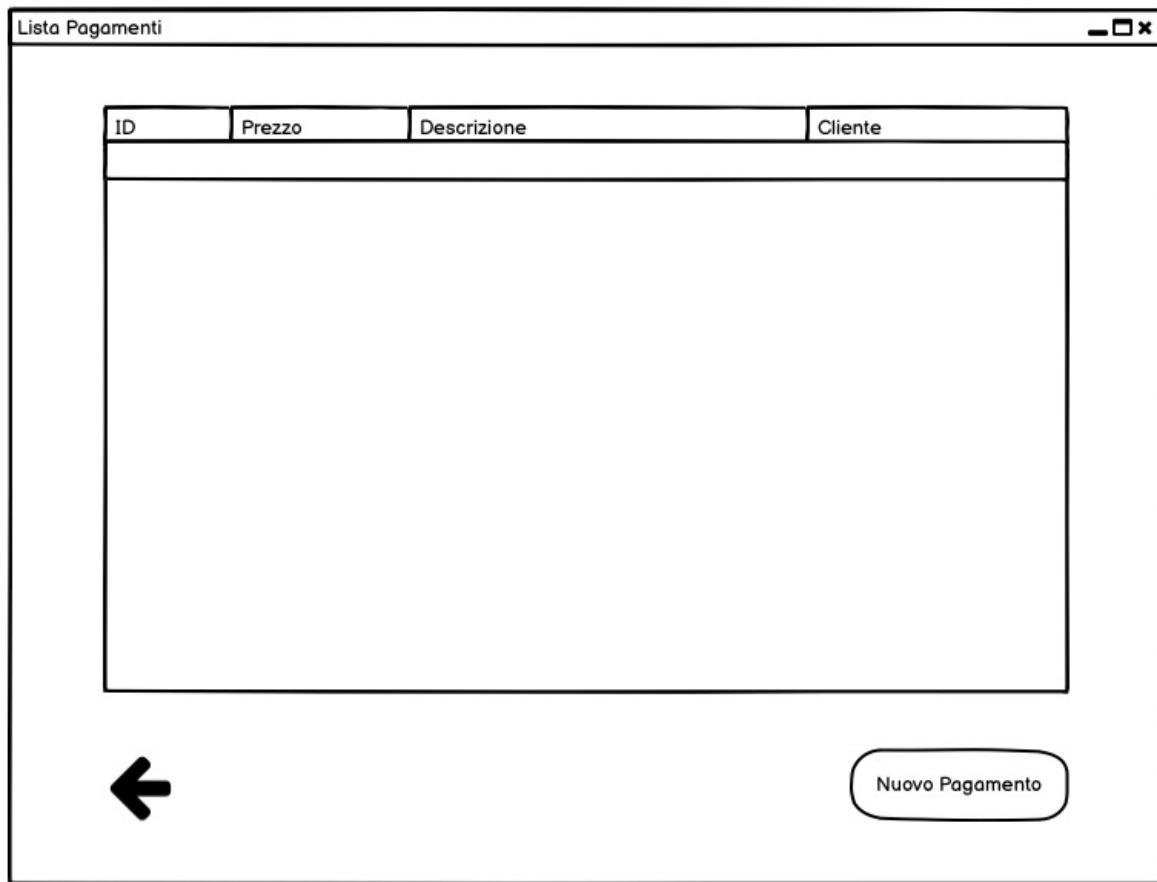


Figura 49: Vista lista pagamenti

Effettua Pagamento

Selezione Cliente	<input type="text"/>	Descrizione Pagamento	<input type="text"/>
Prezzo	<input type="text"/>		

← Procedi ricevuta

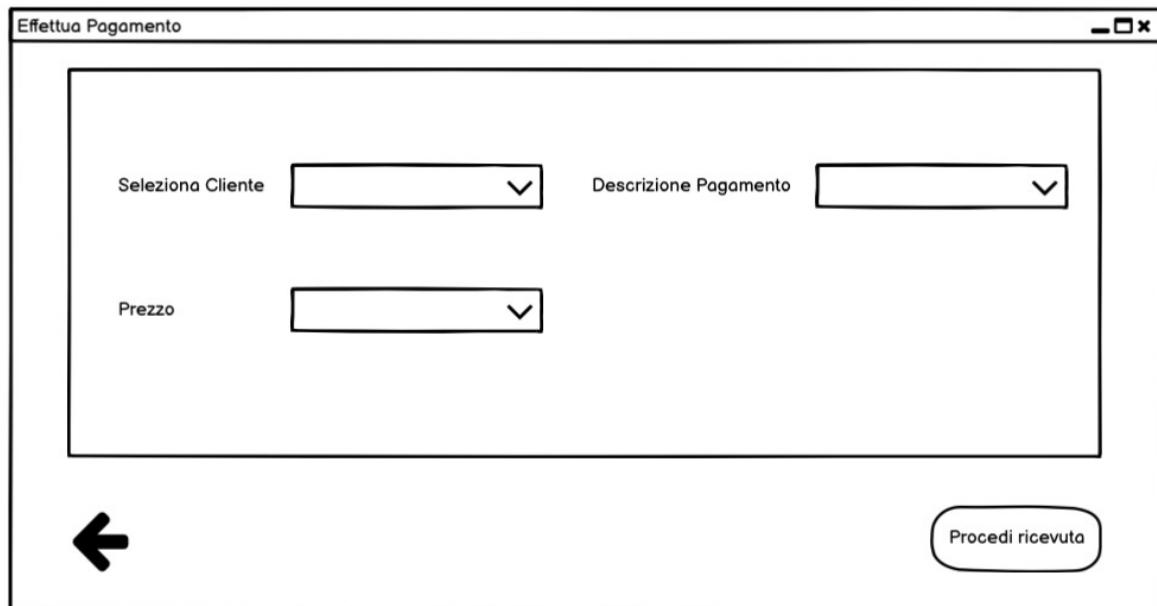


Figura 48: Vista pagamento

Riepilogo

Nome e Cognome

Pagamento N°

Prezzo

Descrizione

Conferma Pagamento

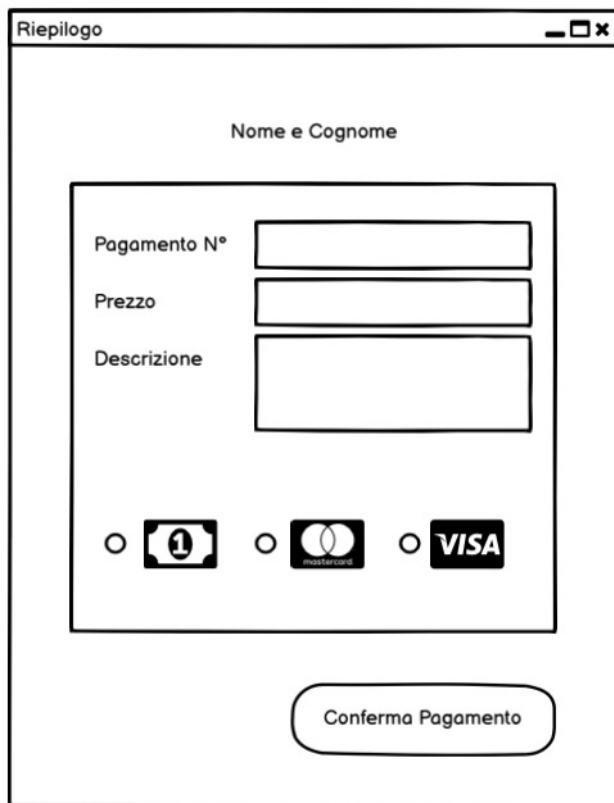


Figura 50: Vista riepilogo

Lista Documenti

Apri	Nome	Elimina

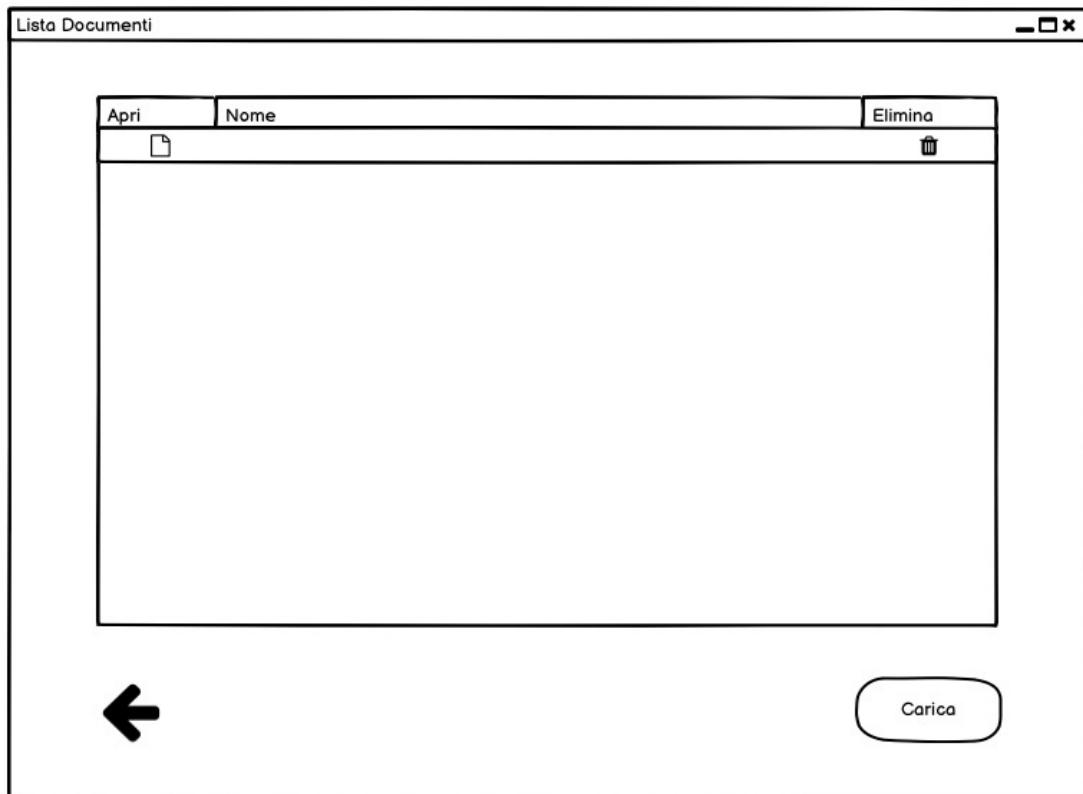


Figura 51: Vista lista documenti

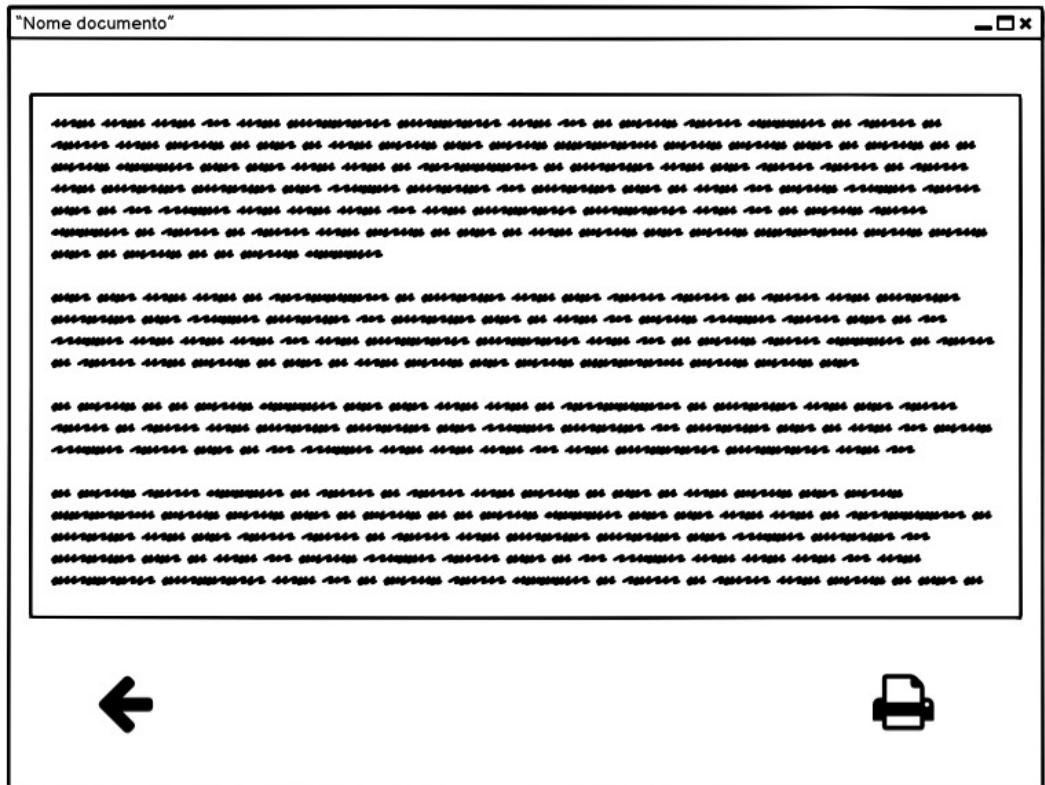


Figura 52: Vista documento

ID	Cliente	Data visita	Descrizione	Esito
			●	●

← Prenota Visita

Figura 53: Vista lista visite mediche

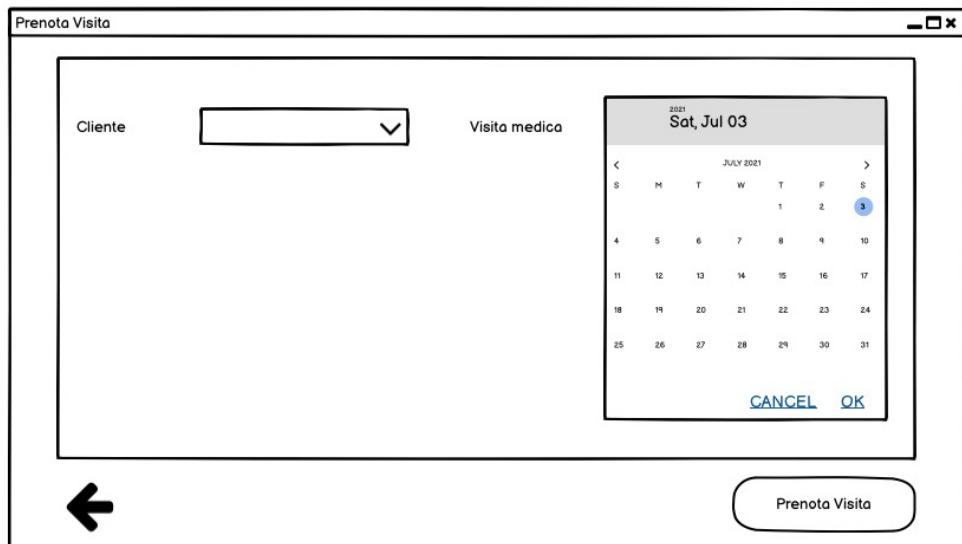


Figura 54: Vista prenota visita medica

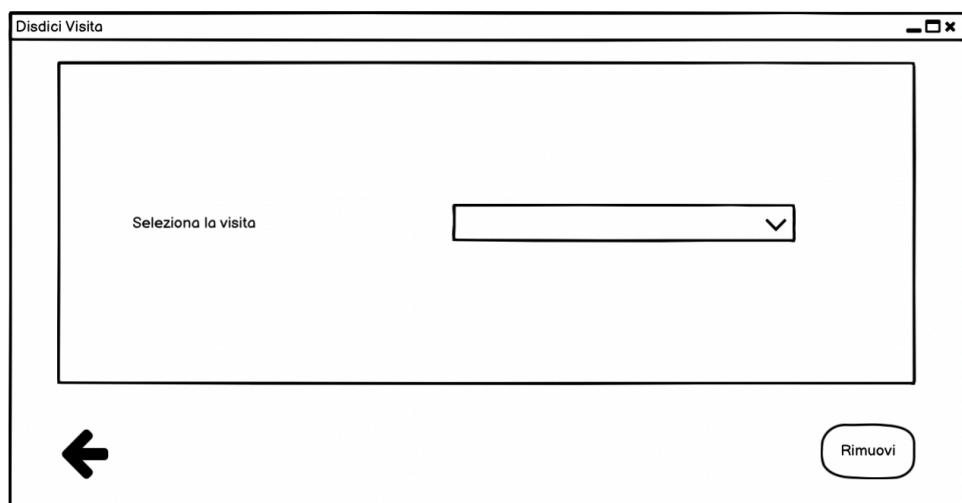


Figura 55: Vista disdici visita medica

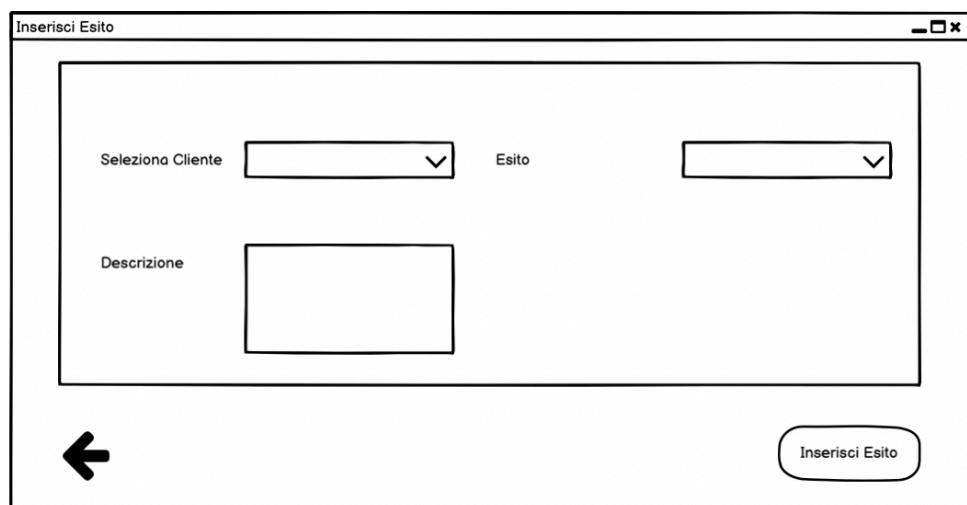


Figura 56: Vista inserisci esito visita

## 14. Implementazione

### Introduzione

Python è un linguaggio di programmazione ad alto livello. È estremamente semplice da utilizzare poiché supporta molti paradigmi di programmazione, tra i quali quello object-oriented, quello imperativo e offre, inoltre, una tipizzazione dinamica molto forte, cosa non presente in molti linguaggi di programmazione. In questo progetto, infatti, è stato utilizzato python3 (ultima versione di Python), in modo da poter lavorare con le migliori caratteristiche messe a disposizione da Python.

La libreria utilizzata per la grafica di questo progetto è PyQt5.

PyQt5 è un insieme di moduli Python che permette di realizzare interfacce grafiche mantenendo la portabilità sulle diverse piattaforme. Inoltre, è possibile aggiungere dei plugin che consentono l'estensione di Qt Designer, realizzando quindi interfacce grafiche senza dover scrivere codice ma soltanto combinando i componenti messi a disposizione da essa.

Per quanto riguarda il salvataggio dei dati nei file, abbiamo deciso di utilizzare la libreria Pickle. Proprio perché utilizziamo Python, Pickle è la scelta più adatta grazie alla sua semplicità d'utilizzo e alla capacità di ricostruire oggetti Python completi. Quindi viene utilizzato per serializzare e deserializzare strutture di oggetti Python. Inoltre, ha anche una caratteristica che lo rende estremamente utile, il multiprocessing. Quest'ultimo rende la scrittura e lettura su file molto più veloce rispetto che al single thread.

### Esempi

Come visto nei grafici precedenti e dai diagrammi, abbiamo scelto di utilizzare una struttura basata sull'object-oriented. Questo perché, già nella fase di progettazione, ci sono state delle caratteristiche che ci hanno permesso di definire quale sarebbe stata la struttura e anche l'architettura più adatta al nostro progetto. Il pattern architettonale utilizzato è il Model-View-Controller (MVC), utilizzato per la gestione di dati e le operazioni associate. Abbiamo individuato inizialmente 6 model principali: Cliente, Dipendente, Pagamento, Prenotazione, VisitaMedica e Documento. Poiché possono esserci tanti oggetti di ogni model, sono state realizzate anche le relative liste, che contengono tutti i dati di uno specifico oggetto.

Di seguito riportiamo alcune parti di codice come esempi:

```

# == set_data ==
# La funzione si occupa di salvare le informazioni relative al cliente e contenute nel file
# 'lista_clienti.pickle' nella tabella della VistaListaClienti. Per ogni cliente è disponibile
# aprire e visualizzare le sue informazioni attraverso un bottone 'Apri' ed è possibile cancellarle
# attraverso un bottone 'Elimina'.
def set_data(self):
    i = 1
    n_righe = len(self.lista_clienti)
    self.table.setRowCount(n_righe + 1)
    for cliente in self.lista_clienti:
        controller_cliente = ControllerCliente(cliente)

        sino = QPushButton()
        sino.setIcon(self.icon_sino)
        self.button_group_pagamenti.addButton(sino, i)

        apri = QPushButton()
        apri.setIcon(self.apri_icon)
        apri.setIconSize(QSize(30, 30))
        self.button_group_apri.addButton(apri, i)
        self.table.setCellWidget(i, 0, apri)

        if controller_cliente.get_id() == "None":
            self.table.setItem(i, 1, QTableWidgetItem(self.no))
        else:
            self.table.setItem(i, 1, QTableWidgetItem(controller_cliente.get_id()))
            self.table.setItem(i, 2, QTableWidgetItem(controller_cliente.get_nome()))
            self.table.setItem(i, 3, QTableWidgetItem(controller_cliente.get_cognome()))
            self.table.setItem(i, 4, QTableWidgetItem(controller_cliente.get_tipo_patente()))
            if controller_cliente.get_esame_teorico() == "None":
                self.table.setItem(i, 5, QTableWidgetItem(self.no))
            else:
                self.table.setItem(i, 5, QTableWidgetItem(self.si))
            if controller_cliente.get_esame_pratico() == "None":
                self.table.setItem(i, 6, QTableWidgetItem(self.no))
            else:
                self.table.setItem(i, 6, QTableWidgetItem(self.si))
            if controller_cliente.get_pagamento_iniziale() == "None" and \
                    controller_cliente.get_pagamento_esame_teorico() == "None" and \
                    controller_cliente.get_pagamento_lezioni_guida() == "None" and \
                    controller_cliente.get_pagamento_esame_pratico() == "None":
                self.table.setItem(i, 7, QTableWidgetItem(self.no))
            else:
                if controller_cliente.get_pagamento_iniziale() != "None" and \
                        controller_cliente.get_pagamento_esame_teorico() != "None" and \
                        controller_cliente.get_pagamento_lezioni_guida() != "None" and \
                        controller_cliente.get_pagamento_esame_pratico() != "None":
                    self.table.setItem(i, 7, QTableWidgetItem(self.si))
            else:
                self.table.setCellWidget(i, 7, sino)
        if controller_cliente.get_prenotazione() == "None":
            if controller_cliente.get_esame_pratico() == "Effettuato" and \
                    controller_cliente.get_esame_teorico() == "Effettuato":
                self.table.setItem(i, 8, QTableWidgetItem(self.si))
            else:
                self.table.setItem(i, 8, QTableWidgetItem(self.no))
        else:
            self.table.setItem(i, 8, QTableWidgetItem(controller_cliente.get_prenotazione()))

        delete = QPushButton()
        delete.setIcon(self.cancella_icon)
        delete.setIconSize(QSize(35, 35))
        self.button_group_elimina.addButton(delete, i)
        self.table.setCellWidget(i, 9, delete)
    i += 1

```

Figura 57: Funzione `set_data()` della `lista_clienti()`

La funzione `set_data()` si occupa di impostare i dati all'interno della tabella. Viene richiamato il `ControllerCliente` per ricavare le informazioni di ogni cliente.

Viene creato il bottone “sino” (di colore arancione) che viene utilizzato per aprire i dettagli dei pagamenti. La stessa icona ma di colore rosso o verde utilizzata in ID, Esame teorico/pratico, Pagamenti e in Prenotazione e serve per indicare lo stato di avanzamento.

Vengono creati 2 bottoni “Apri” e “Elimina” per ogni riga della tabella, utilizzati per aprire/eliminare i dettagli di un particolare cliente utilizzando la funzione `on_selection_apri()`/`on_selection_elimina()` che prendono come parametro il numero della riga a cui corrisponde la posizione del cliente nella lista. L'eliminazione comporta la rimozione di tutti i dati relativi a quel cliente.

```

# == go_pagamento ==
# La funzione apre la VistaRiepilogo e gli passa i parametri di pagamento
# per il rinnovo della patente.
def go_pagamento(self):
    if self.edit_nome.text() == "" or self.edit_cognome.text() == "" or self.edit_patente.text() == "":
        QMessageBox.critical(self, "Attenzione", "Inserisci tutti i dati!")
    else:
        if len(self.edit_patente.text()) != 10:
            QMessageBox.critical(self, "Errore", "Il numero patente immesso non è valido")
            self.edit_patente.clear()
        else:
            self.visitamedica = VisitaMedica(self.edit_nome.text()+" "+self.edit_cognome.text(),
                                              self.calendar.selectedDate().toString("dd-MM-yyyy"))
            self.controller.aggiungi_visita(self.visitamedica)
            self.controller.save_data()

            self.pagamento = VistaRiepilogo("80", "Bollettini + visita medica (Rinnovo patente)", -1,
                                              self.edit_nome.text()+" "+self.edit_cognome.text())
            self.pagamento.show()
            self.close()

```

Figura 58: Funzione go\_pagamento() della VistaRinnovoPatente()

La funzione go\_pagamento() relativo alla classe VistaRinnovoPatente si occupa di prenotare il cliente (non iscritto, ma che intende rinnovare la patente) ad una visita medica ed inserirlo nella lista delle visite mediche, che contiene sia le visite eseguite che prenotate. Una volta prenotato il cliente viene visualizzato il riepilogo del pagamento da effettuare.

```

def inserisci_esito(self):
    for cliente in self.lista_clienti:
        controller_cliente = ControllerCliente(cliente)

    if self.selezione_cliente.currentText() == controller_cliente.get_nome()+" "+controller_cliente.get_cognome():
        if self.esito_esame.currentText() == "IDONEO":
            nome = controller_cliente.get_prenotazione().split("-")
            if nome[2].strip() == "Esame teorico":
                controller_cliente.set_esame_teorico("Effettuato")
            if nome[2].strip() == "Esame pratico":
                controller_cliente.set_esame_pratico("Effettuato")
                controller_cliente.set_prenotazione("None")
            else:
                controller_cliente.set_prenotazione("None")
        QMessageBox.information(self, "Confermato", "Esito inserito con successo")
        self.controller.save_data()
    self.go_home_istruttore = VistaHomeIstruttore.VistaHomeIstruttore(self.tipo)
    self.go_home_istruttore.show()
    self.close()

```

Figura 59: Funzione inserisci\_esito() della VistaInserisciEsitoEsame()

La funzione inserisci\_esito() relativo alla classe VistaInserisciEsitoEsame si occupa di far inserire al segretario/istruttore, secondo i risultati ricevuti dalla motorizzazione, i risultati d'esame di un cliente. L'istruttore/segretario potrà scegliere soltanto un cliente prenotato ad un esame e inserire l'esito dell'esame. Una volta inserito l'esito, se positivo, viene settato l'esame ad "Effettuato" e sulla lista clienti verrà visualizzato il pallino verde, altrimenti se negativo, verrà cancellata la prenotazione e rimarrà il pallino rosso.

## 15. PyUnit

Una volta terminata la scrittura del codice la nostra attenzione si è rivolta al testing del programma. L'obiettivo era dimostrare che il software sviluppato fosse in grado di svolgere i compiti per cui era stato progettato e potesse soddisfare i requisiti funzionali richiesti dal cliente. Ovviamente i test dovevano essere utili anche nel rilevare eventuali errori o bug nel codice, consentendoci di consegnare un prodotto che fosse quanto più possibile all'altezza delle aspettative del cliente.

La nostra attività di testing si è concentrata perlopiù sul testing delle unità, provando quindi singole componenti del programma e testando tutte le operazioni associate a un oggetto. I componenti che siamo andati a testare sono i Controller delle liste che gestiscono i dati relativi a clienti, dipendenti, pagamenti, prenotazioni, documenti e visite mediche. Infatti, dato che sono i Controller a gestire e attuare i comandi che vanno a modificare lo stato dei Model e delle View, abbiamo ritenuto sufficiente controllare che questi componenti funzionassero nel modo corretto per essere sicuri che poi i dati venissero gestiti e presentati nel modo corretto.

Per effettuare i test abbiamo utilizzato PyUnit, un framework di testing per Python.

A seguire sono riportati esempi di TestCase che abbiamo implementato nel programma grazie all'IDE PyCharm.

```
import os
import pickle
from unittest import TestCase
from ProgettoAutoscuola.ListeController.ControllerListaPagamenti import ControllerListaPagamenti
from ProgettoAutoscuola.Model.Pagamento import Pagamento

class TestControllerListaPagamenti(TestCase):

    def test_aggiungi_pagamento(self):
        self.controller = ControllerListaPagamenti()
        self.pagamento = Pagamento("50", "esame teorico", "giuseppe")
        self.controller.aggiungi_pagamento(self.pagamento)

    def test_elimina_pagamento_by_cliente(self):
        self.test_aggiungi_pagamento()
        if os.path.isfile('Data/lista_pagamenti.pickle'):
            with open('Data/lista_pagamenti.pickle', 'rb') as f:
                self.lista_pagamenti = pickle.load(f)
            self.assertTrue(self.controller.elimina_pagamento_by_cliente(self.pagamento.get_cliente()))
            self.assertFalse(self.controller.elimina_pagamento_by_cliente("aaaaaaa"))

    def test_get_lista_pagamenti(self):
        self.test_aggiungi_pagamento()
        self.assertNotEqual(self.controller.get_lista_pagamenti(), [])
```

Figura 60: TestControllerListaPagamenti

```

from ProgettoAutoscuola.ListeController.ControllerListaVisiteMediche import ControllerListaVisiteMediche
from ProgettoAutoscuola.Model.VisitaMedica import VisitaMedica

class TestControllerListaVisiteMediche(TestCase):

    def test_aggiungi_visita(self):
        self.controller = ControllerListaVisiteMediche()
        self.visita = VisitaMedica("giuseppe", "30/07/2021")
        self.controller.aggiungi_visita(self.visita)

    def test_disdici_visita(self):
        self.test_aggiungi_visita()
        if os.path.isfile('Data/lista_visite.pickle'):
            with open('Data/lista_visite.pickle', 'r') as f:
                self.lista_visite = pickle.load(f)
                self.visita_da_eliminare = self.visita.get_cliente() + self.visita.get_data()
            self.assertFalse(self.controller.disdici_visita("aaaaa"))
            self.assertTrue(self.controller.disdici_visita(self.visita_da_eliminare))

    def test_get_lista_visite(self):
        self.test_aggiungi_visita()
        self.assertNotEqual(self.controller.get_lista_visite(), [])

```

Figura 61: *TestControllerListaVisiteMediche*

```

from ProgettoAutoscuola.ListeController.ControllerListaPrenotazioni import ControllerListaPrenotazioni
from ProgettoAutoscuola.Model.Prenotazione import Prenotazione

class TestControllerListaPrenotazioni(TestCase):

    def test_aggiungi_prenotazione(self):
        self.controller = ControllerListaPrenotazioni()
        self.prenotazione = Prenotazione("lezione teorica", "27/07/2021")
        self.controller.aggiungi_prenotazione(self.prenotazione)

    def test_rimuovi_prenotazione_by_id(self):
        self.test_aggiungi_prenotazione()
        if os.path.isfile('Data/lista_prenotazioni.pickle'):
            with open('Data/lista_prenotazioni.pickle', 'r') as f:
                self.lista_prenotazioni = pickle.load(f)
                a = self.lista_prenotazioni.index(self.prenotazione)
            self.assertTrue(self.controller.rimuovi_prenotazione_by_index(a))
            self.assertFalse(self.controller.rimuovi_prenotazione_by_index(10000))

    def test_get_lista_prenotazioni(self):
        self.test_aggiungi_prenotazione()
        self.assertNotEqual(self.controller.get_lista_prenotazioni(), [])

```

Figura 62: *TestControllerListaPrenotazioni*

```

from ProgettoAutoscuola.ListeController.ControllerListaDipendenti import ControllerListaDipendenti
from ProgettoAutoscuola.Model.Dipendente import Dipendente

class TestControllerListaDipendenti(TestCase):

    def test_add_dipendente(self):
        self.controller = ControllerListaDipendenti()
        self.dipendente = Dipendente("test", "test", "234", "13/05/1999", "pescara", "ciao@ciao.com", "23435",
                                     "segretario", "indeterminato", "maschio")
        self.controller.add_dipendente(self.dipendente)

    def test_elimina_dipendente_by_index(self):
        self.test_add_dipendente()
        if os.path.isfile('Data/lista_dipendenti.pickle'):
            with open('Data/lista_dipendenti.pickle', 'r') as f:
                self.lista_dipendenti = pickle.load(f)
            a = self.lista_dipendenti.index(self.dipendente)
            self.assertFalse(self.controller.rimuovi_dipendente_by_index(10000))
            self.assertTrue(self.controller.rimuovi_dipendente_by_index(a))

    def test_get_lista_dipendenti(self):
        self.test_add_dipendente()
        self.assertNotEqual(self.controller.get_lista_dipendenti(), [])

```

Figura 63: *TestControllerListaDipendenti*

```

from ProgettoAutoscuola.Model.Cliente import Cliente
from ProgettoAutoscuola.ListeController.ControllerListaClienti import ControllerListaClienti

class TestControllerListaClienti(TestCase):

    def test_aggiungi_cliente(self):
        self.controller = ControllerListaClienti()
        self.cliente = Cliente("Luca", "dambro", "2345", "via ciao", "email@ciao.it", "2134", "23/03/2000", "maschio")
        self.controller.aggiungi_cliente(self.cliente)

    def test_rimuovi_cliente_by_index(self):
        self.test_aggiungi_cliente()
        if os.path.isfile('Data/lista_clienti.pickle'):
            with open('Data/lista_clienti.pickle', 'rb') as f:
                self.lista_clienti = pickle.load(f)
            a = self.lista_clienti.index(self.cliente)
            self.assertFalse(self.controller.rimuovi_cliente_by_index(10000))
            self.assertTrue(self.controller.rimuovi_cliente_by_index(a))

    def test_get_lista_clienti(self):
        self.test_aggiungi_cliente()
        self.assertNotEqual(self.controller.get_lista_clienti(), [])

```

Figura 64: *TestControllerListaClienti*