ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA - DISI

CORSO DI LAUREA TRIENNALE IN INGEGNERIA INFORMATICA

TESI DI LAUREA

in

INGEGNERIA DEL SOFTWARE T

Progettazione di un software gestionale per la gestione delle prenotazioni nell'ambito della ristorazione

CANDIDATO:

Leonardo Ruberto

RELATORE:

Chiar.mo Prof. Marco Patella

Anno Accademico 2021/2022

Sessione II

Sommario

Abstract

Il progetto riguarda la realizzazione di un software gestionale per la gestione delle prenotazioni nell'ambito della ristorazione. L'idea nasce dalla mia esperienza lavorativa di cinque anni in un ristorante dove i proprietari gestivano le prenotazioni scrivendole giorno per giorno su carta. Questo, oltre ad essere un metodo lento e macchinoso, ha provocato spesso errori e disguidi con i clienti.
La mia intenzione è quindi quella di informatizzare tutto questo processo aggiungendo delle funzionalità volte a risparmiare tempo e ad ottimizzare l'occupazione dei tavoli. Verranno gestiti i turni di ogni tavolo, le prenotazioni ricorrenti di clienti fissi e la modifica della disposizione dei tavoli con un'interfaccia grafica all'avanguardia per far risultare tutto fluido.
Con un sistema distribuito risulta anche possibile decentralizzare la gestione, permettendo ad ogni utente autorizzato di prendere prenotazioni, di visualizzare quelle attuali e di disporre il cliente al proprio tavolo senza necessità di passare per il proprietario che gli fornisca le informazioni.

Analisi dei requisiti

Requisiti del sistema

- L'amministratore di sistema definisce lo staff ed il suo titolare. Questo a sua volta può registrare nuovi dipendenti. Ogni dipendente ha un ruolo: titolare, gestore, utente normale.
  Titolare e gestori possono gestire le prenotazioni e lo schema dei tavoli, mentre gli utenti normali possono solo visualizzare, senza effettuare modifiche.
  Il titolare può e deve almeno una volta definire lo schema standard dei tavoli.
- Deve essere presente una sezione per prendere la prenotazione ed un'altra per visualizzare la mappa dei tavoli giorno per giorno. Ad ogni tavolo possono essere associate più prenotazioni. In più deve esserci la possibilità di interagire con lo schema, confermando l'arrivo di clienti prenotati ad un determinato tavolo e notificando quando questi se ne vanno.
- La dimensione e la disposizione dei tavoli deve essere personalizzabile e avere un riscontro grafico affidabile; i tavoli possono essere di varie dimensioni e questo deve essere possibile da definire, anche graficamente, da parte del titolare e/o dei gestori.
- Ogni prenotazione deve contenere dei dati, come il nome e il cognome se necessario, data e ora, il numero di persone, la presenza di bambini o animali, il numero di cellulare, la preferenza per una determinata sala o tavolo. Il nome e la data di una prenotazione devono essere univoci.
- Essendo che molti clienti prenotano in certi giorni per un lungo periodo, deve essere possibile effettuare delle prenotazioni multiple all'interno di un range di date.
- Un problema noto è quello della gestione dei turni di ogni tavolo; bisogna definire un procedimento per l'assegnamento che non faccia sovrapporre prenotazioni tra loro e che non lasci libero un tavolo per troppo tempo. Al momento di prendere la prenotazione per un determinato giorno il sistema deve mostrare all'utente i tavoli disponibili e indicativamente un orario nel quale il tavolo dovrebbe essere libero. Per fare ciò è necessario definire un tempo medio durante il quale si suppone il tavolo rimanga occupato.
- Bisogna anche gestire il fatto che non tutti i tavoli vengono occupati da clienti che hanno effettuato una prenotazione. Per cui, al momento della richiesta, il sistema deve fornire il tavolo più adatto, se disponibile.

Ho deciso di dividere i requisiti sulla base del contesto a cui si riferiscono:

- gestione degli **utenti**
- gestione delle **prenotazioni**
- gestione dello **schema dei tavoli**
- altro

Requisiti funzionali

ID	Requisito
R1F	Un ristorante che utilizza il software deve avere associato uno staff
R2F	Un ristorante ed il relativo staff può essere inserito solo dall'amministratore di sistema
R3F	Alla creazione dello staff viene definito anche un utente titolare
R4F	Uno staff è identificato da un codice di accesso
R5F	Gli utenti si identificano tramite username e codice di accesso dello staff
R6F	Gli utenti si autenticano tramite credenziali di username e password e codice di accesso dello staff
R7F	La registrazione e la rimozione degli utenti sono a carico del titolare
R8F	Il titolare può concedere/revocare ruoli agli utenti
R9F	Ogni utente può cambiare la propria password personale
R10F	Ogni utente ha un ruolo: titolare, gestore, utente normale
R11F	Il ruolo di utente normale permette la sola visualizzazione delle prenotazioni e dello schema dei tavoli
R12F	Il ruolo di gestore permette la gestione delle prenotazioni, la gestione dello schema dei tavoli e la modifica dello schema stesso
R13F	Il ruolo di titolare è come quello di gestore, ma in più può gestire i dipendenti e definire lo schema standard.
R14F	Aggiungendo una prenotazione la si inserisce in una lista con tutte le restanti
R15F	Possibilità di aggiungere più prenotazioni contemporaneamente, ovvero le prenotazioni multiple
R16F	Ogni prenotazione deve avere un nome, una data, un numero di persone, un orario ed eventualmente il numero di bambini o la presenza di animali. Inoltre, può avere una nota associata ed un numero di cellulare
R17F	Ogni prenotazione può essere modificata o rimossa
R18F	Ogni prenotazione può essere piazzata o meno ad un tavolo
R19F	Se una prenotazione risulta non piazzata "tot" tempo prima del suo orario, l'utente va notificato
R20F	Ogni prenotazione ha associato un tempo stimato nel quale si suppone il tavolo si liberi
R21F	Il tempo medio di occupazione di un tavolo può essere definito solo dal titolare
R22F	Ogni giorno ha associato uno ed uno solo schema di tavoli
R23F	Lo schema standard va definito dal titolare
R24F	Lo schema dei tavoli può essere modificato dal titolare o dai gestori
R25F	Lo schema può essere resettato a quello standard
R26F	Ogni tavolo ha una dimensione che corrisponde al numero di persone che può contenere normalmente
R27F	Allo schema possono essere aggiunti o rimossi tavoli
R28F	Ogni tavolo può avere più prenotazioni associate nello stesso giorno
R29F	È possibile occupare un tavolo senza prenotazione, per quei clienti che si accomodano senza aver prenotato
R30F	Per ogni giornata deve essere visualizzabile il numero di tavoli occupati, di quelli rimanenti, di prenotazioni effettuate e di persone totali che hanno prenotato

Requisiti non funzionali

ID	Requisito
R1NF	Può esserci un solo titolare per staff
R2NF	La password degli utenti deve essere lunga almeno 8 caratteri e contenere almeno un numero
R3NF	Il codice di accesso allo staff deve essere lungo almeno 4 caratteri e univoco
R4NF	La password fornita dall'amministratore di sistema a tempo di registrazione va cambiata dal titolare immediatamente dopo il primo login
R5NF	La password fornita dal titolare a tempo di registrazione va cambiata dagli utenti immediatamente dopo il primo login
R6NF	Il sistema deve verificare le autorizzazioni degli utenti
R7NF	Il titolare non può rimuovere sé stesso
R8NF	La data di prenotazione deve essere successiva al momento nel quale si prende
R9NF	Prenotazioni possono essere filtrate nella visualizzazione per giorno, ora, numero persone, piazzate o no
R10NF	Le prenotazioni possono essere anche filtrate per nome
R11NF	Le prenotazioni devono avere nome e data univoci, per non incorrere in errori
R12NF	Quando un utente prende o piazza una prenotazione, l'evento viene associato alla prenotazione
R13NF	Una prenotazione può essere modificata o rimossa da un solo utente alla volta
R14NF	I nomi delle prenotazioni si salvano nel sistema dopo l'inserimento della prenotazione, così da poterli selezionare in fase di aggiunta
R15NF	Il tempo stimato per ogni prenotazione è a default il tempo medio
R16NF	Il tempo medio è a default 1h15min, ma può essere modificato dal titolare
R17NF	La dimensione standard per un tavolo è di 4 persone, ma può essere modificato dal titolare
R18NF	Ogni tavolo ha anche un aspetto grafico coerente con la sua dimensione
R19NF	L'interfaccia dello schema deve essere comoda da utilizzare sia in fase di modifica che di visualizzazione
R20NF	Selezionando un tavolo all'interno dello schema devono essere mostrate le prenotazioni associate ad esso
R21NF	Lo schema dei tavoli può essere modificato da un solo utente alla volta
R22NF	L'aggiornamento del sistema deve essere immediato dopo ogni modifica
R23NF	Le interfacce del sistema devono essere pensate anche per un controllo touchscreen
R24NF	Velocità di memorizzazione e recupero dati
R25NF	Velocità nella sincronizzazione delle modifiche

Requisiti aggiunti dopo una seconda analisi

ID	Requisito
R31F	L'amministratore vede la lista di tutti gli staff presenti nel sistema
R26NF	Per la fase di autenticazione l'amministratore avrà un codice di accesso staff ad hoc
R27NF	I nomi degli staff devono essere univoci
R28NF	Una prenotazione può essere aggiunta dallo schema di un determinato giorno inserendo in automatico la data
R29NF	Dalla lista di prenotazioni di un determinato giorno deve essere rapido l'accesso allo schema di quel giorno
R30NF	Quando una prenotazione viene rimossa dallo schema, questa deve risultare non piazzata
R31NF	Lo schema standard deve essere stato definito prima di poter utilizzare lo schema di un qualsiasi giorno
R32NF	Resettare lo schema a quello standard deve essere possibile se e solo se il numero di tavoli di destinazione è maggiore o uguale a quello attuale

Analisi del dominio

Vocabolario

Voce	*Definizione*	*Sinonimi*
Amministratore di sistema	Utente con privilegi di sistema aggiuntivi	
Privilegio di sistema	Autorizzazione intrinseca concessa ad un amministratore di sistema che riguarda la gestione del software stesso. Non riguarda gli staff	
Staff	Gruppo di utenti legato ad un ristorante	
Cliente	Persona che prenota il tavolo al ristorante	
Utente	Persona registrata nel software	
Ruolo	Autorizzazione che ha l'utente all'interno dello staff	
Titolare	Membro dello staff con privilegi massimi	
Dipendenti	Membri dello staff sotto il titolare	
Gestore	Membro dello staff che si occupa della gestione delle prenotazioni e dello schema dei tavoli	
Utente normale	Membro dello staff che può solo visualizzare le prenotazioni e lo schema dei tavoli	
Prenotazione	Quando un cliente chiede di riservare un tavolo	
Tavolo	Rappresenta il tavolo di un ristorante al quale può essere associata una prenotazione	
Tavolo libero	Senza nessuna prenotazione associata in nell'orario selezionato	
Schema dei tavoli	Mappa che rappresenta la disposizione dei tavoli all'interno del ristorante in un determinato giorno	Mappa dei tavoli
Schema standard dei tavoli	Schema dei tavoli usato a default in tutti i giorni	
Disposizione dei tavoli	Corrisponde alla posizione e all'orientamento dei tavoli all'interno dello schema	
Prenotazione multipla	Insieme di prenotazioni tutte uguali (stesso nome, ora, numero persone, …) ma ognuna con data differente	
Turni di un tavolo	Corrisponde al numero di volte in cui un tavolo viene occupato da persone diverse in una giornata	
Tavoli disponibili	Tavoli che in un determinato momento non hanno associate prenotazioni	
Tempo medio	Tempo durante il quale si suppone che un tavolo rimanga occupato	
Reset schema	Riportare la disposizione e dimensione di ogni tavolo a quella standard	

Analisi dei requisiti

Modello dei casi d'uso

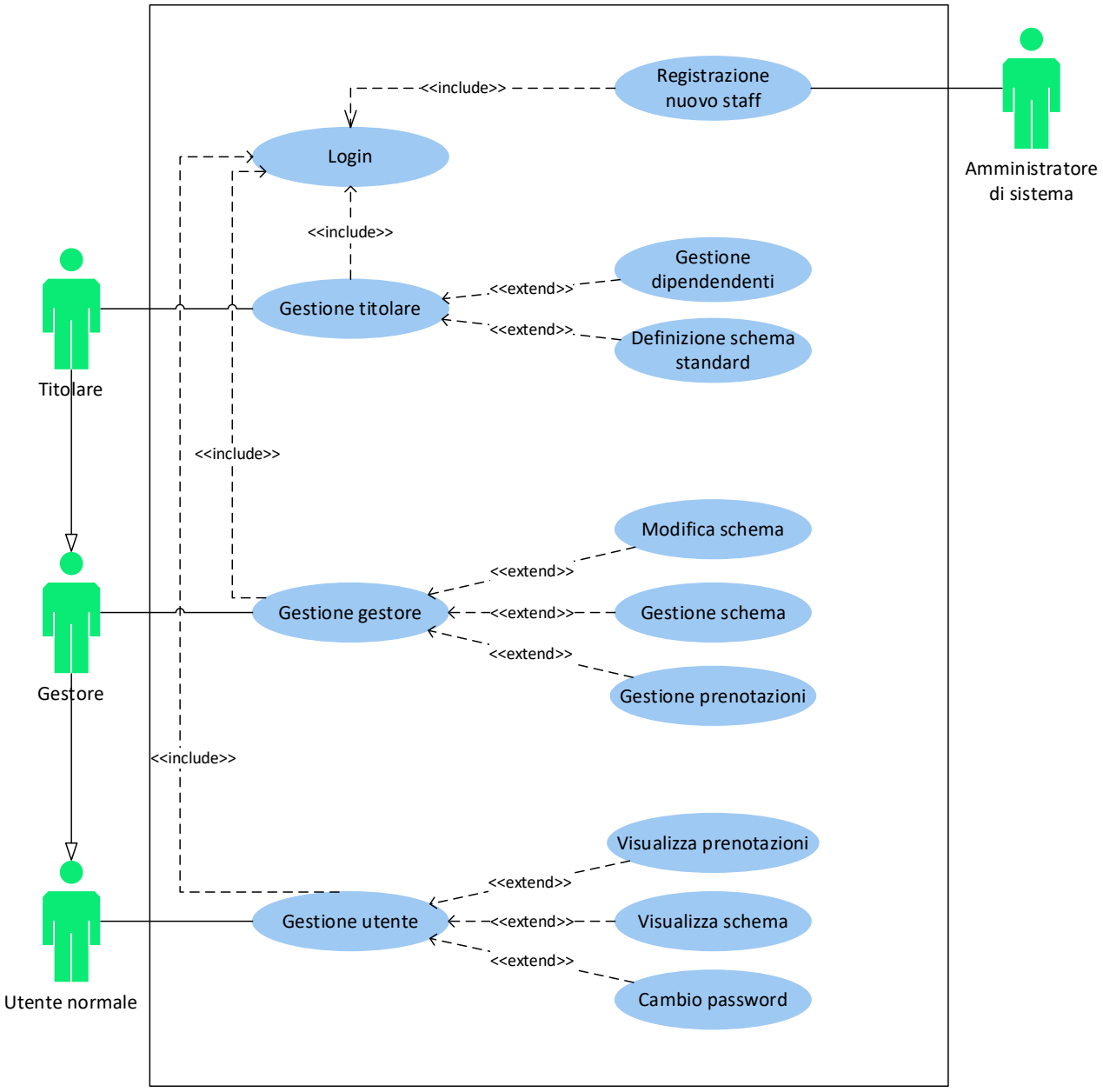


Figura 1 - analisi_dei_requisiti /modello_dei_casi_d'uso.vsdx

I casi d'uso "Gestione schema" e "Gestione prenotazioni" sono stati inseriti per semplificare lo schema e comprendono rispettivamente le seguenti funzionalità:

- Assegna e rimuovi una prenotazione da un tavolo all'interno dello schema di un giorno
- Aggiungi, modifica, rimuovi prenotazioni

Grazie all'ereditarietà "Gestore" potrà accedere a tutte le funzionalità disponibili per "Utente normale". A sua volta "Titolare" avrà disponibili tutte le funzionalità di "Gestore".

Scenari

Registrazione nuovo staff

Titolo	Registrazione nuovo staff
Descrizione	Viene registrato un nuovo staff legato ad un ristorante
Attori	Amministratore di sistema
Relazioni	
Precondizioni	
Post condizioni	Viene creato uno staff con solo un utente titolare all'interno
Scenario principale	1. Login 2. Il sistema offre all'attore una maschera per l'aggiunta di un nuovo staff 3. L'amministratore di sistema crea lo staff inserendone il nome ed il ristorante a cui è legato 4. Crea un utente con ruolo di titolare 5. Il sistema genera un codice di accesso e lo associa allo staff e al titolare
Scenari alternativi	A. Nome staff già presente: • Il sistema notifica l'attore dell'errore • Viene ripresentata la maschera per l'inserimento dei dati
Requisiti non funzionali	R1NF, R2NF, R3NF, R27NF
Punti aperti	

Login

Titolo	Login
Descrizione	Sistema di autenticazione
Attori	Utente normale, Gestore, Titolare, Amministratore di sistema
Relazioni	Gestione utente, Gestione gestore, Gestione titolare, Cambio password
Precondizioni	
Post condizioni	L'utente è autenticato nel sistema ed ha accesso a tutte le funzionalità che il suo ruolo gli permette
Scenario principale	1. Il sistema presenta all'attore una maschera per l'inserimento delle credenziali: codice staff, username e password 2. L'attore inserisce i dati 3. Il sistema verifica la correttezza dei dati 4. L'attore viene autenticato e gli viene mostrata la maschera consona
Scenari alternativi	A. Password non valida: • Il sistema notifica l'attore dell'errore con un messaggio consono • Viene ripresentata la maschera per l'inserimento dei dati B. L'utente è già autenticato: • Viene mostrata la schermata consona C. L'attore è il titolare e ha la password impostata dall'amministratore: • L'attore deve cambiare la password D. L'attore è un utente normale e ha la password impostata dal titolare: • L'attore deve cambiare la password
Requisiti non funzionali	R4NF, R5NF, R6NF, R26NF
Punti aperti	

Gestione titolare

Titolo	Gestione titolare
Descrizione	Sezione ad hoc per il titolare
Attori	Titolare
Relazioni	Login, Gestione dipendenti, Definizione schema standard
Precondizioni	
Post condizioni	
Scenario principale	1. Login 2. L'attore può scegliere di eseguire una delle funzionalità disponibili
Scenari alternativi	
Requisiti non funzionali	R6NF
Punti aperti	

Definizione schema standard

Titolo	Definizione schema standard
Descrizione	Viene definita la dimensione e la disposizione dei tavoli che verrà usata a default per lo schema di ogni giorno
Attori	Titolare
Relazioni	Gestione titolare
Precondizioni	
Post condizioni	
Scenario principale	1. Il sistema offre all'attore una schermata per definire il numero di tavoli e la relativa posizione 2. Per ogni tavolo va definita una dimensione (se non è quella standard) 3. L'attore può anche modificare il tempo medio di occupazione e la dimensione standard dei tavoli 4. Ad ogni tavolo l'attore associa un numero identificativo
Scenari alternativi	A. Lo schema è già definito: • L'attore può effettuare le modifiche allo schema già presente B. Uno o più tavoli non sono numerati: • Il sistema notifica l'attore dell'errore con un messaggio consono • Viene ripresentata la maschera per l'inserimento dei dati
Requisiti non funzionali	R16NF, R17NF, R18NF, R19NF
Punti aperti	

Gestione dipendenti

Titolo	Gestione dipendenti
Descrizione	L'attore può registrare un nuovo utente, rimuoverlo o cambiarne il ruolo
Attori	Titolare
Relazioni	Gestione titolare
Precondizioni	
Post condizioni	
Scenario principale	1. L'attore sceglie quale funzionalità compiere 2. Se vuole registrare un nuovo utente inserisce le credenziali del nuovo dipendente (username e password) e il sistema collega in modo automatico il codice staff del titolare al cliente 3. Se vuole effettuare un'altra operazione seleziona il dipendente e decide se rimuoverlo o cambiarne il ruolo
Scenari alternativi	A. Registrazione: username già presente nel sistema: • Il sistema notifica l'attore dell'errore • Viene ripresentata la maschera per l'inserimento dei dati
Requisiti non funzionali	R7NF, R2NF
Punti aperti	

Gestione gestore

Titolo	Gestione gestore
Descrizione	Sezione ad hoc per i gestori
Attori	Titolare, Gestore
Relazioni	Login, Modifica schema, Gestione schema, Gestione prenotazioni
Precondizioni	
Post condizioni	
Scenario principale	1. Login 2. L'attore può scegliere di eseguire una delle funzionalità disponibili
Scenari alternativi	
Requisiti non funzionali	R6NF
Punti aperti	

Modifica schema

Titolo	Modifica schema
Descrizione	Viene modificata la dimensione e la disposizione dei tavoli relativa ad un determinato giorno
Attori	Titolare, Gestore
Relazioni	Gestione gestore
Precondizioni	Lo schema standard deve essere già stato definito
Post condizioni	Lo schema viene modificato solo per il giorno selezionato
Scenario principale	1. L'attore seleziona un giorno ed il relativo schema 2. Lo schema viene modificato, aggiungendo, rimuovendo e modificando i tavoli 3. Al salvataggio il sistema aggiorna lo schema
Scenari alternativi	A. Uno o più tavoli non sono numerati: • Il sistema notifica l'attore dell'errore con un messaggio consono • Viene ripresentata la maschera per l'inserimento dei dati
Requisiti non funzionali	R18NF, R19NF, R21NF, R30NF, R31NF
Punti aperti	

Gestione schema

Titolo	Gestione schema
Descrizione	Gestione delle prenotazioni da piazzare all'interno di uno schema
Attori	Titolare, Gestore
Relazioni	Gestione gestore
Precondizioni	Deve essere stato definito lo schema standard
Post condizioni	
Scenario principale	1. Il sistema mette a disposizione dell'attore una maschera per aggiungere o rimuovere una prenotazione da un tavolo 2. L'attore può confermare l'arrivo di un tavolo e notificare quando se ne va 3. L'attore può occupare il tavolo per clienti senza prenotazione
Scenari alternativi	
Requisiti non funzionali	R12NF, R19NF, R20NF, R21NF
Punti aperti	

Gestione utente

Titolo	Gestione utente
Descrizione	Sezione per tutti gli utenti
Attori	Titolare, Gestore, Utente normale
Relazioni	Login, Visualizza prenotazioni, Visualizza schema, Cambio password
Precondizioni	
Post condizioni	
Scenario principale	1. Login 2. L'attore può scegliere di eseguire una delle funzionalità disponibili
Scenari alternativi	
Requisiti non funzionali	R6NF
Punti aperti	

Gestione prenotazioni

Titolo	Gestione prenotazioni
Descrizione	Aggiunta, modifica e rimozione delle prenotazioni
Attori	Titolare, Gestore
Relazioni	Gestione gestore, Visualizza prenotazioni
Precondizioni	
Post condizioni	
Scenario principale	1. Il sistema mette a disposizione dell'attore una maschera con la visualizzazione delle prenotazioni 2. L'attore può scegliere se aggiungere una nuova prenotazione oppure modificare o rimuovere una di quelle esistenti 3. In caso di aggiunta o modifica il sistema offre una maschera per l'inserimento dei dati 4. L'attore inserisce i dati e salva
Scenari alternativi	A. Aggiunta/modifica, l'utente non inserisce dati obbligatori (nome, data, ora, pax.): • Il sistema avvisa l'attore con un messaggio consono • Gli viene ripresentata la maschera per l'inserimento dei dati B. Aggiunta/modifica, l'utente inserisce una coppia nome-data già presente nel sistema: • Il sistema avvisa l'attore con un messaggio consono • Gli viene ripresentata la maschera per l'inserimento dei dati
Requisiti non funzionali	R8NF, R11NF, R12NF, R13NF, R14NF, R15NF
Punti aperti	

Visualizza schema

Titolo	Visualizza schema
Descrizione	Visualizzazione dello schema dei tavoli con le prenotazioni associate
Attori	Titolare, Gestore, Utente
Relazioni	Gestione utente
Precondizioni	
Post condizioni	
Scenario principale	1. L'attore sceglie la data relativa allo schema 2. Il sistema mostra lo schema scelto
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Cambio password

Titolo	Cambio password
Descrizione	L'attore può cambiare la propria password personale
Attori	Titolare, Gestore, Utente normale
Relazioni	Gestione utente
Precondizioni	
Post condizioni	
Scenario principale	1. Il sistema offre all'attore una maschera per l'inserimento della nuova password e della conferma della stessa 2. L'attore inserisce i dati 3. Il sistema controlla la conformità dei dati 4. La nuova password viene associata all'attore che l'ha modificata
Scenari alternativi	A. Le due password non corrispondono: • Il sistema allerta l'utente con un messaggio • Viene invitato l'attore a re-inserire i dati B. La nuova password è uguale alla vecchia: • Il sistema allerta l'utente con un messaggio • Viene invitato l'attore a re-inserire i dati
Requisiti non funzionali	
Punti aperti	

Visualizza prenotazioni

Titolo	Visualizza prenotazioni
Descrizione	L'attore può visualizzare le prenotazioni filtrandole
Attori	Titolare, Gestore, Utente normale
Relazioni	Gestione utente, Gestione prenotazioni
Precondizioni	
Post condizioni	
Scenario principale	1. Il sistema offre all'attore una maschera con tutte le prenotazioni e con la possibilità di porre filtri 2. I filtri disponibili sono: • Data e ora • Piazzate o meno • Numero persone • Nome della prenotazione
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Analisi del rischio

Valutazione dei beni

Bene	*Valore*	*Esposizione*
Sistema gestionale	Alto Registra tutte le prenotazioni di un ristorante, quindi molto critico dal punto di vista della sicurezza	Alta Perdita d'immagine del software e degli staff, spesa in tempo per il ripristino del backup. Se il sistema fallisce o viene manomesso perdite finanziare dovute a disguidi con i clienti
Informazioni su utenti	Medio-Alto Le uniche informazioni degli utenti sono le credenziali, che possono permettere l'accesso e la manomissione dei dati	Alta Perdita d'immagine del software e degli staff, spesa in tempo per il ripristino del backup. Se i dati delle prenotazioni vengono manomessi perdite finanziare dovute a disguidi con i clienti
Informazioni su prenotazioni	Medio-Basso Le informazioni sono ridotte al nome e in alcuni casi al numero di cellulare del cliente	Bassa
Informazioni su amministratore di sistema	Medio-Alto Credenziali dell'amministratore di sistema	Bassa L'amministratore di sistema può solo definire nuovi staff

Analisi minacce e controlli

Minaccia	*Probabilità*	*Controllo*	*Fattibilità*
Furto d'identità (utente)	Media Username e password scelti dal titolare, con password modificabile dall'utente	Log degli accessi, blocco dell'utente.	Costo minimo
Furto d'identità (amministratore di sistema)	Bassa	Log degli accessi	Costo minimo
Alterazioni dei dati nella comunicazione Man-In-The-Middle	Media	Uso di un canale di comunicazione sicura	Costo basso
Dos/DDos	Bassa	Backup locale delle prenotazioni ogni giorno	Costo medio

Analisi delle tecnologie della sicurezza

Tecnologia	*Vulnerabilità*
Cifratura delle comunicazioni	Utilizzo di una cifratura ibrida, simmetrica e asimmetrica, con le loro relative vulnerabilità: Cifratura Simmetrica: • Tempo di vita della chiave • Più informazioni vengono cifrate con la stessa chiave, più materiale è offerto per l'analisi del testo a un attaccante • Memorizzazione della chiave • Lunghezza della chiave Cifratura Asimmetrica: • Lunghezza della chiave • Memorizzazione della chiave privata. L'utilizzo di cifrature standard non recenti potrebbero causare vulnerabilità ulteriori.
Autenticazione tramite credenziali	• Social engineering: un utente potrebbe essere vittima di un attacco di social engineering, volto al furto delle credenziali. • Password cracking: attraverso metodologie di cracking delle password, l'attaccante risale alla password di un utente o amministratore di sistema. • Negligenza: l'utente volontariamente o per sbaglio fornisce le credenziali a terzi. • Password banali
Architettura Client/Server	• Attacco DDoS/DoS: l'obiettivo dell'attacco è quello di negare il servizio offerto agli utenti. • Attacco Man in the Middle: un malintenzionato potrebbe riuscire a sottrarre informazioni o effettuare operazioni illecite. • Sniffing della comunicazione: un malintenzionato potrebbe riuscire ad intercettare una comunicazione e a sottrarne informazioni. • Deadlock: il servitore potrebbe trovarsi in una situazione di stallo generata da una richiesta ciclica. • Esposizione dei client: i client sono esposti, dato che devono connettersi al servitore.

Security Use Case & Misuse Case

Modello

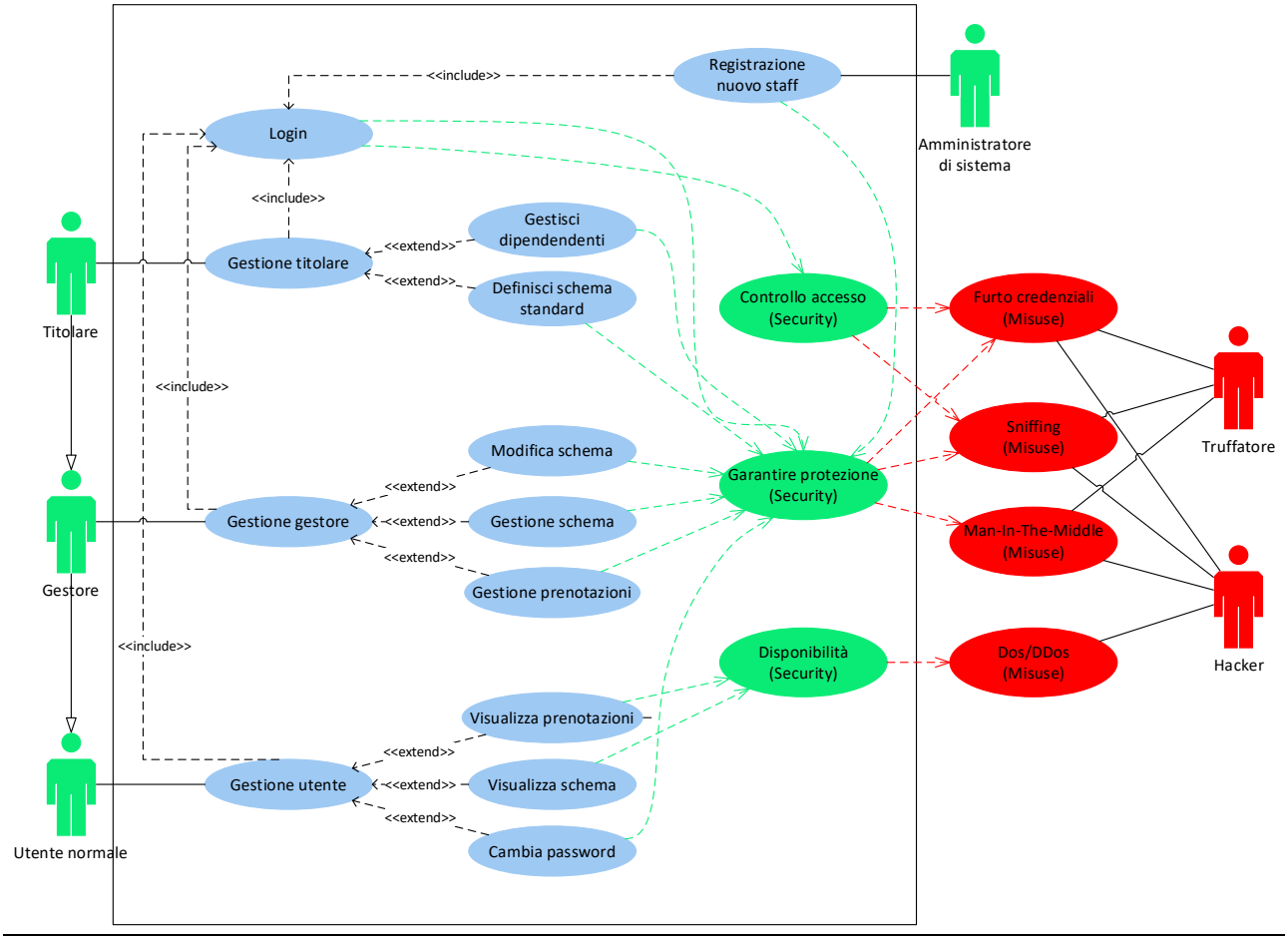


Figura 2 - analisi_dei_requisiti/modello_dei_security_e_misuse_case.vsdx

Scenari

Caso d'uso	Controllo accesso	
Percorso d'uso	Controllo sull'autenticazione dell'utente	
Misuse case	Furto credenziali	
Descrizione	Il sistema deve impedire l'accesso a malintenzionati	
Rischi alla sicurezza	Un malintenzionato ruba i dati di identificazione e autenticazione ad un utente	
Precondizioni	1. Il malintenzionato ha la possibilità di trovare l'username. 2. Il malintenzionato ha la possibilità di trovare il codice di accesso dello staff 3. Il malintenzionato ha la possibilità di tentare l'accesso al sistema	
Post condizioni	Il sistema rileva il tentativo di accesso fraudolento e blocca l'utente	
Scenario principale	**Sistema**	**Attaccante**
		L'attaccante tenta un attacco di password cracking
	Il sistema rileva l'attacco e blocca l'account dell'utente	
Scenario di attacco avvenuto con successo	**Sistema**	**Attaccante**
		L'attaccante tenta un attacco e trova la password
	Il sistema fornisce l'accesso all'utente	
		L'attaccante manomette le informazioni
	Il sistema notifica nei log le operazioni effettuate.	
	L'amministratore di sistema nota i log anomali e opera di conseguenza.	

Caso d'uso	Garantire protezione	
Percorso d'uso	Garantire protezione dei dati della comunicazione	
Misuse case	Sniffing, Man-In-The-Middle, Furto credenziali	
Descrizione	I dati che viaggiano nelle comunicazioni devono essere protetti	
Rischi alla sicurezza	Un utente malintenzionato potrebbe leggere ed alterare i dati in transito in una comunicazione	
Precondizioni	1. Il malintenzionato ha la possibilità di intercettare i dati. 2. Il malintenzionato ha la possibilità di modificare i dati. 3. Il malintenzionato ha i mezzi per spedire il messaggio modificato al destinatario.	
Post condizioni	Il sistema rileva il tentativo di modifica della comunicazione	
Scenario principale	**Sistema**	**Attaccante**
		L'attaccante rileva un messaggio in transito, lo intercetta, lo modifica e lo inoltra all'utente finale
	Il sistema riceve il messaggio modificato e rileva il tentativo di lettura o modifica, rifiutandolo e segnalandolo nei log	
Scenario di attacco avvenuto con successo	**Sistema**	**Attaccante**
		L'attaccante riesce ad alterare la comunicazione senza che il sistema se ne accorga
	Il sistema registra nei log le modifiche effettuate	
	L'amministratore di sistema nota i log anomali e opera di conseguenza	

Caso d'uso	Garantire protezione	
Percorso d'uso	Garantire protezione dei dati persistenti	
Misuse case	Sniffing, Man-In-The-Middle, Furto credenziali	
Descrizione	I dati persistenti devono essere protetti	
Rischi alla sicurezza	Un utente malintenzionato potrebbe alterare i dati persistenti di cui fa uso il software	
Precondizioni	1. Sono presenti i dati di uno staff e le relative prenotazioni 2. L'attaccante ha i mezzi per tentare di modificare i dati persistenti.	
Post condizioni	Il sistema blocca il tentativo di modifica o compromissione dei dati persistenti	
Scenario principale	**Sistema**	**Attaccante**
		L'attaccante cerca di modificare dati persistenti del sistema gestionale
	Il sistema blocca l'attacco e notifica l'amministratore di sistema	
Scenario di attacco avvenuto con successo	**Sistema**	**Attaccante**
		L'attaccante riesce ad accedere e a modificare dati persistenti
	Il sistema registra nei log le modifiche effettuate	
	L'amministratore di sistema nota i log anomali e opera di conseguenza	

Caso d'uso	Disponibilità	
Percorso d'uso	Garantire la disponibilità delle prenotazioni e dello schema dei tavoli	
Misuse case	DoS/DDoS	
Descrizione	Le prenotazioni e lo schema dei tavoli devono essere sempre visibili	
Rischi alla sicurezza	Un malintenzionato vuole impedire l'accesso al sistema	
Precondizioni	1. Il malintenzionato ha i mezzi per effettuare questo tipo di attacco	
Post condizioni	Il sistema rileva il tentativo di attacco	
Scenario principale	**Sistema**	**Attaccante**
		L'attaccante tenta un attacco DoS/DDoS
	Il sistema rileva l'attacco e informa l'amministratore di sistema tramite i log.	
	Il sistema effettua un backup locale dei dati	
	L'amministratore agisce per bloccare l'attacco	
Scenario di attacco avvenuto con successo	**Sistema**	**Attaccante**
		L'attaccante tenta un attacco DoS/DDoS
	Il sistema rileva l'attacco e informa l'amministratore di sistema tramite i log.	
	Il sistema effettua un backup locale dei dati	
	Il sistema va fuori uso	
	L'amministratore fa ripartire il sistema dal backup locale	

Requisiti di protezione dati

Dall'analisi del rischio sono emersi altri requisiti riguardanti la protezione dei dati:

- Creazione di un sistema di log per il tracciamento delle operazioni svolte nel sistema gestionale. L'amministratore di sistema visualizza i log per gestire la situazione.
- Blocco dell'utente dopo 3 tentativi di login errati. L'amministratore può sbloccarlo
- I dati scambiati devono essere protetti con cifratura sicura per evitare alterazione e lettura di dati privati
- I dati persistenti del sistema devono essere protetti da eventuali intrusioni
- Il sistema deve essere in grado di ridurre l'impatto di attacchi DoS
- Inoltre, il sistema deve consentire il ripristino locale nel caso che l'attacco DoS funzioni

Requisiti di sistema aggiornati

R32F	Creare un sistema di log per il tracciamento delle operazioni automatico
R33F	L'amministratore di sistema deve avere il pieno controllo dei log.
R34F	Blocco dell'utente dopo un certo numero di login falliti.
R33NF	I dati della comunicazione devono essere protetti
R34NF	I dati persistenti devono essere protetti
R35NF	Il sistema deve cercare di garantire la disponibilità del servizio
R36NF	Il blocco dell'account deve avvenire dopo 3 tentativi
R37NF	L'amministratore di sistema ha l'autorizzazione a sbloccare l'account
R38NF	Il sistema deve essere strutturato in modo da essere replicabile localmente

Vocabolario aggiornato

Voce	*Definizione*	*Sinonimi*
Log	Oggetto dove vengono salvate le informazioni relative ad un evento avvenuto nel sistema	
Blocco dell'utente	Operazione effettuata in automatico dal sistema, impedendo l'accesso all'utente	
Sblocco dell'utente	Operazione effettuata dall'amministratore di sistema, permettendo all'utente di accedere nuovamente al sistema	

Casi d’uso aggiornati

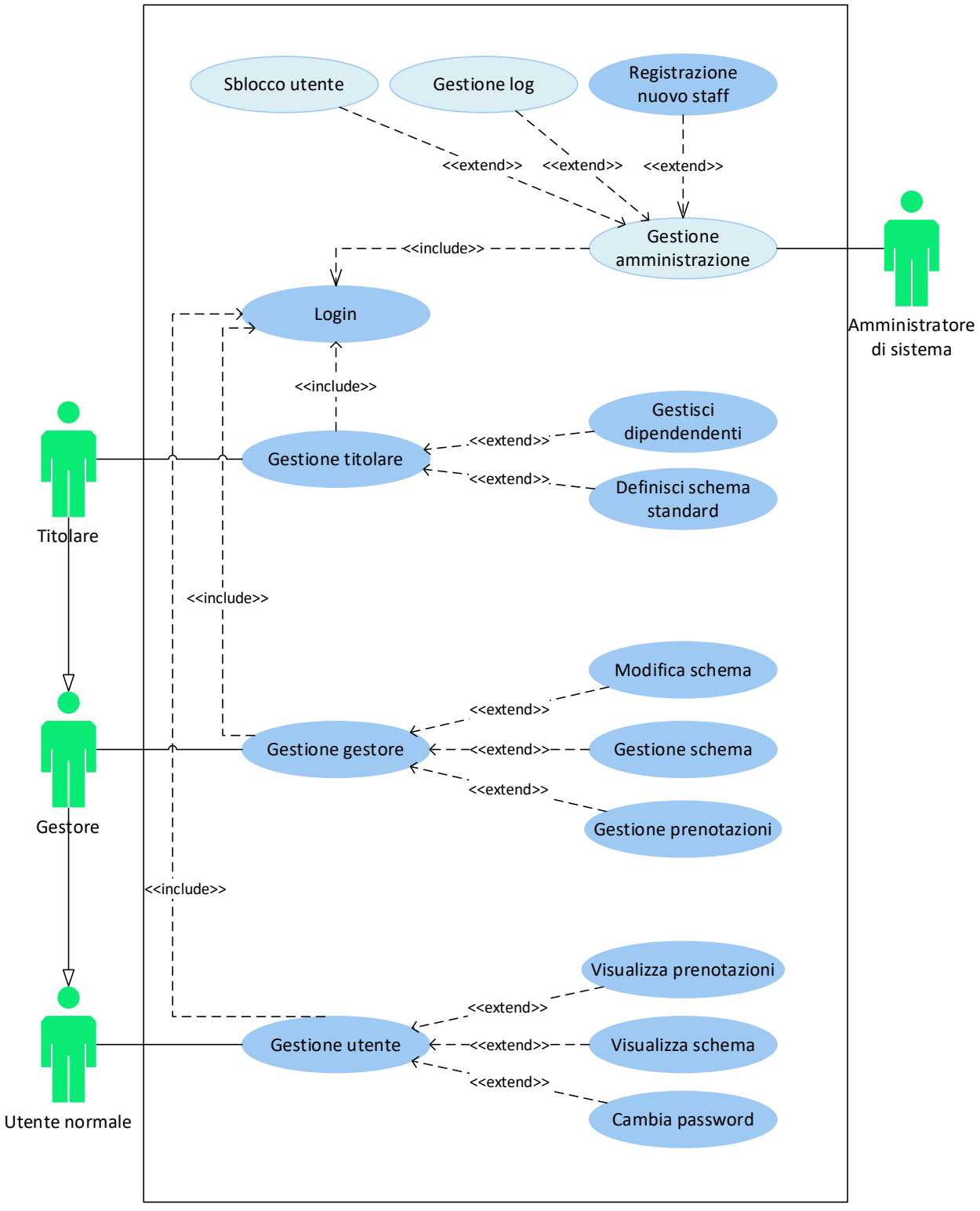


Figura 3 - analisi_dei_requisiti/modello_dei_casi_d'uso_aggiornato.vsdx

Sono stati aggiunti 3 casi d’uso, tutti legati all’attore “Amministratore di sistema”. Mentre “Gestione Amministrazione” serve solo da container per tutte le funzionalità da admin, “Sblocco utente” e “Gestione log” sono le vere e proprie funzionalità aggiunte. Cambia anche “Login” in quanto viene inserito il concetto di blocco dell’utente dopo 3 tentativi di autenticazione errati.

Scenari aggiornati

Gestione amministrazione

Titolo	Gestione amministrazione
Descrizione	Sezione ad hoc per l'amministratore di sistema
Attori	Amministratore di sistema
Relazioni	Login, Gestione log, Sblocco utente, Registrazione nuovo staff
Precondizioni	
Post condizioni	
Scenario principale	1. Login 2. L'attore vede la lista di tutti gli staff presenti nel sistema 3. L'attore sceglie quale funzionalità usare tra quelle disponibili
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Gestione Log

Titolo	Gestione Log
Descrizione	Visualizzazione con filtri di tutti i log
Attori	Amministratore di sistema
Relazioni	Gestione amministrazione
Precondizioni	
Post condizioni	
Scenario principale	1. L'attore visualizza tutti i log 2. Il sistema offre la possibilità di filtrarli per data, per funzione e per utente, per staff e per importanza
Scenari alternativi	
Requisiti non funzionali	
Punti aperti	

Sblocco utente

Titolo	Sblocco utente
Descrizione	L'amministratore di sistema sblocca un utente bloccato
Attori	Amministratore di sistema
Relazioni	Gestione amministrazione
Precondizioni	L'utente è stato bloccato
Post condizioni	
Scenario principale	1. L'attore si informa sulla situazione e decide cosa fare 2. Se vuole sbloccare l'utente seleziona lo staff e poi l'utente da sbloccare
Scenari alternativi	
Requisiti non funzionali	R38NF
Punti aperti	

Login

Titolo	Login
Descrizione	Gli utenti si loggano all'interno del sistema
Attori	Utente normale, Gestore, Titolare
Relazioni	Gestione utente, Gestione gestore, Gestione titolare, Cambio password, Gestione amministrazione
Precondizioni	L'utente deve essere stato registrato nel sistema
Post condizioni	L'utente è autenticato nel sistema ed ha accesso a tutte le funzionalità che il suo ruolo gli permette
Scenario principale	1. Il sistema presenta all'attore una maschera per l'inserimento delle credenziali: codice staff, username e password 2. L'attore inserisce i dati 3. Il sistema verifica la correttezza dei dati 4. L'attore viene autenticato e gli viene mostrata la maschera consona
Scenari alternativi	A. Credenziali non valide: • Il sistema notifica l'attore dell'errore con un messaggio consono al tipo di credenziale errata • Viene ripresentata la maschera per l'inserimento dei dati B. L'utente è già autenticato: • Viene mostrata la schermata consona C. L'utente è il titolare e ha la password impostata dall'amministratore: • Reindirizzamento alla schermata di cambio password D. L'attore è un utente normale e ha la password impostata dal titolare: • Reindirizzamento alla schermata di cambio password E. L'utente ha sbagliato 3 volte le credenziali: • Blocco dell'account • Il sistema notifica l'attore del blocco dell'account • Viene ripresentata la maschera di login F. L'utente ha l'account bloccato • Il sistema notifica l'attore del blocco dell'account • Viene ripresentata la maschera di login
Requisiti non funzionali	R4NF, R5NF, R6NF, R26NF, R37NF
Punti aperti	

Analisi del problema

Analisi delle funzionalità

Tabella delle funzionalità

Funzionalità	Tipo	Grado di complessità	Requisiti collegati
Gestione amministrazione	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R31F
Registrazione nuovo staff	Memorizzazione dei dati, interazione con l'esterno	Semplice	R1F, R2F, R3F, R4F
Gestione log	Gestione dei dati, interazione con l'esterno	Semplice	R32F, R33F
Sblocco utente	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Semplice	R34F
Login	Lettura di dati, interazione con l'esterno	Semplice	R5F, R6F, R34F
Gestione titolare	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R13F
Gestione dipendenti	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R7F, R8F
Definizione schema standard	Gestione dei dati, memorizzazione dei dati	Semplice	R23F, R27F, R21F
Gestione gestore	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R12F
Modifica schema	Gestione dei dati, memorizzazione dei dati	Semplice	R22F, R24F, R25F, R27F
Gestione schema	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R18F
Gestione prenotazioni	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R14F, R15F, R16F, R17F, R18F, R19F, R20F
Gestione utente	Gestione dei dati, memorizzazione dei dati, interazione con l'esterno	Complesso	R11F
Visualizza prenotazioni	Lettura dei dati	Semplice	
Visualizza schema	Lettura dei dati	Semplice	
Cambio password	Memorizzazione dei dati, interazione con l'esterno	Semplice	R9f

Tabelle informazioni-flusso

Gestione amministrazione

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Staff Composto da:	Complesso	Alto	Output	
- nome	Semplice	Basso		Minimo 4, massimo 32 caratteri
- nome ristorante	Semplice	Basso		Minimo 4, massimo 64 caratteri
- indirizzo ristorante	Semplice	Basso		Minimo 4, massimo 256 caratteri
- codice di accesso	Semplice	Alto		4 caratteri alfanumerici

Registrazione nuovo staff

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Staff	Complesso	Alto	Input	
Utente Composto da:	Complesso	Alto	Input	
- nome	Semplice	Basso		Minimo 4, massimo 32 caratteri
- username	Semplice	Alto		Minimo 4, massimo 16 caratteri
- password	Semplice	Alto		Minimo 8 caratteri, di cui almeno 1 speciale
- ruolo	Complesso	Basso		**Enum** {titolare, gestore, utente normale}
- codice di accesso del relativo staff	Semplice	Alto		

Gestione log

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Log Composto da:	Complesso	Alto	Output	
- timestamp	Semplice			DD/MM/YY-HH:MM:SS
- codice staff	Semplice			
- username utente	Semplice			
- descrizione	Semplice			
- importanza	Complesso			**Enum** {basso, medio, alto}

Sblocco utente

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Codice di accesso staff	Semplice	Alto	Input	
Username utente bloccato	Semplice	Alto	Input	
Log	Complesso	Alto	Output	

Login

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Utente	Complesso	Alto	Input	

Gestione titolare

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli

Gestione dipendenti

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Utente	Complesso	Alto	Input/Output	

Definizione schema standard

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Dimensione standard tavoli	Semplice	Basso	Input/Output	Intero maggiore di 0
Tempo medio di occupazione	Semplice	Basso	Input/Output	HH:mm
Schema standard Composto da: - posizione tavoli	Complesso Complesso	Basso Basso	Input/Output	
Tavolo Composto da: - n° identificativo - dimensione - posizione	Complesso Semplice Semplice Complesso	Basso Basso Basso Basso	Input/Output	 Intero maggiore di 0 Intero maggiore di 1

Gestione gestore

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli

Gestione utente

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli

Modifica schema

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Dimensione standard tavoli	Semplice	Basso	Output	Intero maggiore di 0
Schema Composto da: - posizione tavoli - data	Complesso Complesso Semplice	Basso Basso Basso	Input/Output	

Gestione schema

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Schema	Complesso	Basso	Input	
Prenotazione Composto da: - nome - data - ora - numero persone - numero bambini - note - numero cellulare - tempo stimato di occupazione	Complesso Semplice Semplice Semplice Semplice Semplice Semplice Semplice Semplice	Medio Medio Basso Basso Basso Basso Basso Medio Basso	Input/Output	 Massimo 16 caratteri DD-MM-YY HH:mm Intero maggiore di 0 Intero >= 0 Massimo 256 caratteri HH:mm
Prenotazione non piazzata	Complesso	Medio	Input/Output	
Prenotazione piazzata Composto da: - numero tavolo	Complesso	Medio	Input/Output	 Intero maggiore di 0

Gestione prenotazioni

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Prenotazione multipla Composto da: - prenotazioni - range di date	Complesso	Medio	Input/Output	
Prenotazione	Complesso	Medio	Input/Output	

Visualizza prenotazioni

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Prenotazione	Complesso	Medio	Output	

Visualizza schema

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Schema	Complesso	Basso	Output	
Tavolo	Complesso	Basso	Output	

Cambia password

Informazione	Tipo	Livello protezione/privacy	Input/Output	Vincoli
Utente	Complesso	Alto	Input/Output	
Nuova password	Semplice	Alto	Input	Min. 8 carat., almeno 1 speciale

Analisi delle interazioni

Tabella delle maschere

Maschera	Informazioni	Funzionalità
Home amministrazione	Scelta delle funzionalità disponibili e lista degli staff	Gestione amministrazione
View registrazione staff	Input per i dati di uno staff	Registrazione nuovo staff
View registrazione titolare	Input per i dati dell'utente titolare	Registrazione nuovo staff
View log	Log con filtri di visualizzazione	Gestione log
View sblocco utente	Dati utente da sbloccare	Sblocca utente
View autenticazione	Input per username, password e codice di accesso dello staff	Login
Home titolare	Scelta delle funzionalità disponibili	Gestione titolare
Home gestione dipendenti	Lista di dipendenti	Gestione dipendenti
View aggiunta dipendente	Input per dati dell'utente (username, password e ruolo)	Gestione dipendenti
View definizione schema standard	Mappa dei tavoli standard modificabile dal titolare	Definizione schema standard
View impostazioni standard	Dimensione standard dei tavoli e tempo medio delle prenotazioni	Definizione schema standard
Home gestore	Scelta delle funzionalità disponibili	Gestione gestore
View modifica schema	Mappa dei tavoli modificabile	Modifica schema
Home gestione schema	Lista di prenotazioni del giorno selezionato e mappa dei tavoli a cui assegnarle	Gestione schema
Home gestione prenotazioni	Lista di prenotazioni	Gestione prenotazioni
View aggiunta/modifica prenotazione	Input per i dati della prenotazione e mappa dei tavoli relativa al giorno scelto	Gestione prenotazioni
Home	Lista delle prenotazioni e mappa dei tavoli del giorno	Gestione utente
View prenotazioni	Lista di prenotazioni	Visualizza prenotazioni
View schema	Mappa dei tavoli	Visualizza schema
View cambio password	Dati dell'utente e input per la nuova password	Cambio password

Analisi dei vincoli

Tabella dei vincoli

Requisito	Categoria	Impatto	Funzionalità
Facilità di navigazione delle schermate	Usabilità	Rende il sistema più intuitivo da utilizzare	Gestione amministrazione, Gestione titolare, Gestione gestore, Gestione utente
Chiarezza e semplicità delle interfacce	Usabilità	Rende le interfacce più semplici ed intuitive da utilizzare	Gestione schema, Gestione prenotazioni, Gestione dipendenti, Gestione log, Modifica schema, Definizione schema standard
Velocità nel caricamento dei dati	Tempo di risposta	Migliora il riscontro che ha l'utente con l'interfaccia	Gestione schema, Gestione prenotazioni, Gestione dipendenti, Gestione log, Modifica schema
Velocità nella memorizzazione e nell'aggiornamento dei dati	Tempo di risposta, usabilità	Migliora il riscontro che ha l'utente con l'interfaccia e la velocità a leggere le modifiche di altri	Gestione schema, Gestione prenotazioni, Gestione dipendenti, Gestione log, Modifica schema, Definizione schema standard
Protezione ed integrità dei dati	Sicurezza	Peggiora tempi di risposta ma migliora la privacy dei dati	Login, Gestione schema, Gestione prenotazioni, Gestione dipendenti, Gestione log, Cambio password, Sblocco utente
Controllo degli accessi	Sicurezza	Peggiora tempi di risposta ed usabilità ma migliora la privacy dei dati	Gestione amministrazione, Gestione titolare, Gestione gestore, Gestione utente

Analisi dei ruoli e delle responsabilità

Tabella dei ruoli

Ruolo	Responsabilità	Maschere	Riservatezza	Numerosità
Amministratore di sistema	Gestisce il log, lo sblocco di utenti bloccati e la registrazione di un nuovo staff	Home amministrazione, View registrazione staff, View log, View sblocco utente, Login	Massimo grado di riservatezza	Uno, ma possono essercene di più per un controllo più efficiente dei log
Titolare	Gestisce i dipendenti e definisce lo schema standard	Tutte eccetto quelle riservate all'amministratore di sistema (Home amministrazione, View registrazione staff, View log, View sblocco utente)	Alto grado di riservatezza	Uno per ogni staff
Gestore	Gestisce lo schema dei tavoli e le prenotazioni	Home, Home gestore, View modifica schema, Home gestione schema, Home gestione prenotazioni, View aggiunta/modifica prenotazione, View prenotazioni, View schema, View sblocco utente	Medio grado di riservatezza	Numero massimo limitato dalle risorse del sistema
Utente normale	Può solamente visualizzare lo schema dei tavoli e le prenotazioni	Home, View prenotazioni, View schema, View sblocco utente	Basso grado di riservatezza	Numero massimo limitato dalle risorse del sistema

Tabelle ruolo-informazioni

Amministratore di sistema

Informazione	**Tipo di accesso**
Log	**Lettura**
Staff	**Lettura/Scrittura**
Utenti	**Lettura**
Stato utente (bloccato/sbloccato)	**Scrittura**
Utente titolare	**Scrittura**

Titolare

Informazione	**Tipo di accesso**
Schema standard	Lettura/Scrittura
Codice di accesso staff	Lettura
Utenti	Lettura/Scrittura
Schema del giorno	Lettura/Scrittura
Prenotazioni	Lettura/Scrittura
Password personale	Scrittura

Gestore

Informazione	**Tipo di accesso**
Schema del giorno	Lettura/Scrittura
Prenotazioni	Lettura/Scrittura
Password personale	Scrittura

Utente normale

Informazione	**Tipo di accesso**
Schema del giorno	Lettura
Prenotazioni	Lettura
Password personale	Scrittura

Scomposizione del problema

Tabella scomposizione funzionalità

Funzionalità	Scomposizione
Gestione amministrazione	Visualizza staff, Registrazione nuovo staff, Sblocco utente, Gestione Log
Gestione log	Visualizza log, Filtra log
Login	Autenticazione, Blocco utente
Gestione titolare	Definizione schema standard, Gestione dipendenti
Gestione dipendenti	Aggiungi dipendente, Rimuovi dipendente, Cambia ruolo dipendente, Visualizza dipendenti
Gestione gestore	Gestione schema, Gestione prenotazioni, Modifica schema
Gestione schema	Assegna prenotazione al tavolo, Rimuovi prenotazione/clienti dal tavolo, Assegna clienti senza prenotazione al tavolo, Visualizza Tavolo, Visualizza Schema
Gestione prenotazioni	Aggiungi prenotazione, Modifica prenotazione, Rimuovi prenotazione, Visualizza prenotazioni
Gestione utente	Visualizza prenotazioni, Visualizza schema, Cambio password

Tabelle sotto-funzionalità

Gestione amministrazione

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Registrazione nuovo staff	Visualizza staff	La lista degli staff è condizionata dagli staff registrati nel sistema	Staff, Utenti

Login

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Autenticazione	Blocco utente	Un utente può essere bloccato solo dopo che ha provato ad autenticarsi 3 volte, fallendo	Utente

Gestione dipendenti

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Aggiungi dipendente	Visualizza dipendenti	La lista dei dipendenti è condizionata dai dipendenti registrati nel sistema	Utenti
Aggiungi dipendente	Cambia ruolo dipendente	Il ruolo di un dipendente può essere cambiato solo se questo esiste	Utenti
Aggiungi dipendente	Rimuovi dipendente	Il dipendente può essere rimosso solo se questo esiste	Utenti

Gestione schema

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Assegna prenotazione /clienti non prenotati al tavolo	Rimuovi prenotazione/clienti dal tavolo	Una prenotazione o dei clienti non prenotati possono essere rimossi solo se prima sono stati aggiunti ad un tavolo	Schema, Tavolo
Visualizza Tavolo	Visualizza Schema	Solo dopo aver selezionato lo schema è possibile scegliere di visualizzare un tavolo	Schema, Tavolo

Gestione dipendenti

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Aggiungi prenotazione	Visualizza prenotazioni	La lista delle prenotazioni è condizionata dalle prenotazioni registrate nel sistema	Prenotazione
Aggiungi prenotazione	Modifica prenotazione	Una prenotazione può essere modificata solo se esiste	Prenotazione
Aggiungi prenotazione	Rimuovi prenotazione	Una prenotazione può essere rimossa solo se esiste	Prenotazione

Ulteriori legami tra sotto-funzionalità non appartenenti alla stessa funzionalità

Sotto-funzionalità	Sotto-funzionalità	Legame	Informazioni
Modifica schema	Definizione schema standard	Lo schema standard deve essere definito prima di poter modificare quello giornaliero	Schema, Schema standard
Gestione schema	Definizione schema standard	Lo schema standard deve essere definito prima di poter usare lo schema dei tavoli in qualsiasi maniera	Schema, Schema standard
Sblocco utente	Blocco utente	Un utente deve essere stato bloccato prima di essere sbloccato	Utente

Modello del dominio

Analizzando il vocabolario le entità che compaiono sono quelle relative ai vari tipi di utenti, alle prenotazioni, allo schema e ai tavoli. Si aggiunge poi il log, che compare dopo la fase di analisi della sicurezza. Per motivi di chiarezza ho suddiviso il modello in due, la prima che si focalizza sugli utenti e sul log, mentre la seconda che si concentra su prenotazioni, schema e tavoli.

Utenti e log

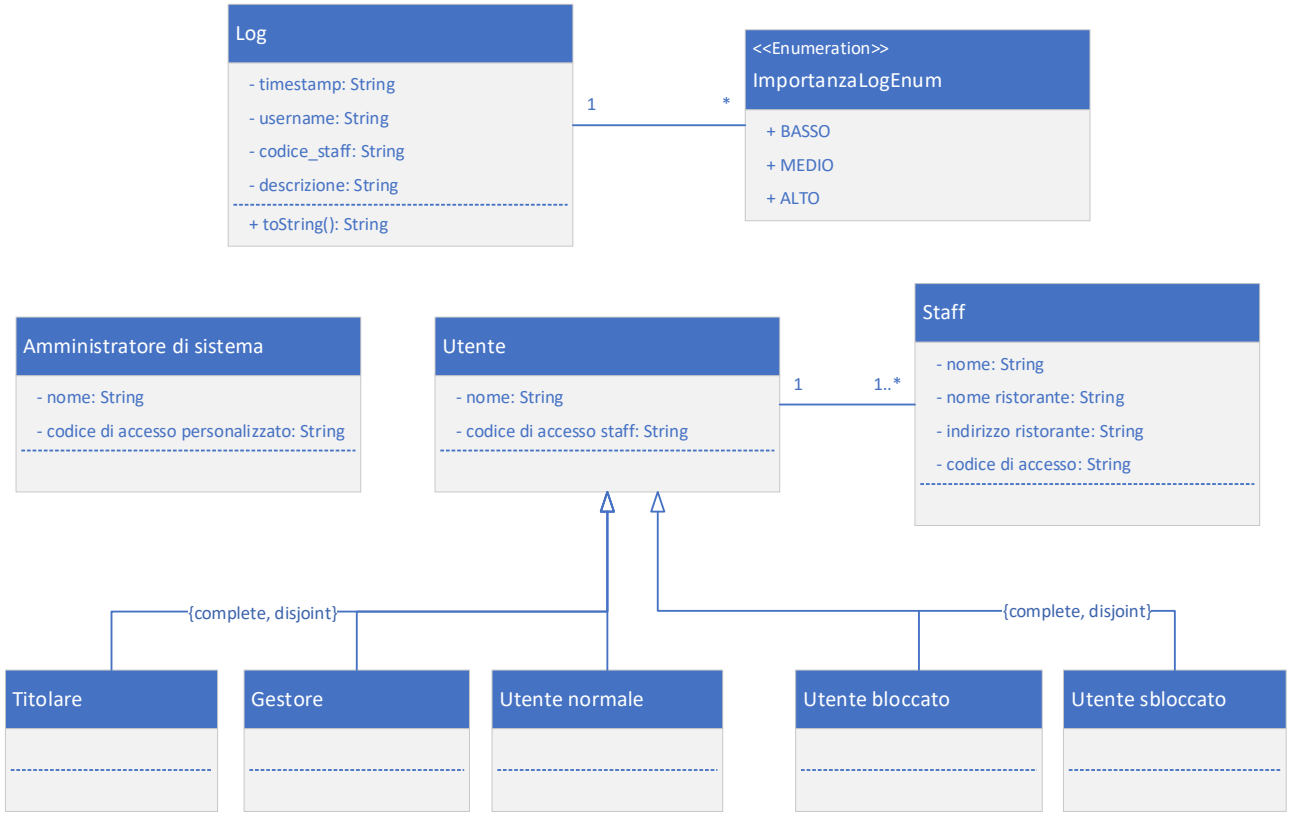


Figura 4 - analisi_del_problema/modello_dominio_utenti.vsdx

Utenti

È stata sfruttata l'ereditarietà per quanto riguarda l'entità *"Utente";* in questo modo è possibile avere ad esempio un utente *"Gestore"* bloccato. Vediamo poi che un utente può essere associato ad un solo *"Staff",* che ovviamente può avere più utenti associati a sua volta.
L'amministratore di sistema non è stato messo come sottoclasse di *"Utente"* in quanto non si associa a *"Staff"* ed è concettualmente un'entità diversa. Si potrà magari pensare in fase di progettazione di avere un'entità madre da cui ereditano sia l'amministratore che gli altri utenti.

Log

Per quanto riguarda *"Log"* non è stata usata l'ereditarietà ma si è preferito l'enumerativo; questo perché anche con livelli di importanza diversi sono tutti la stessa entità concettuale.
È presente poi il metodo *"toString()"* per trasformare l'oggetto rapidamente in una stringa che verrà poi letta in un file di log.

Schema e prenotazioni

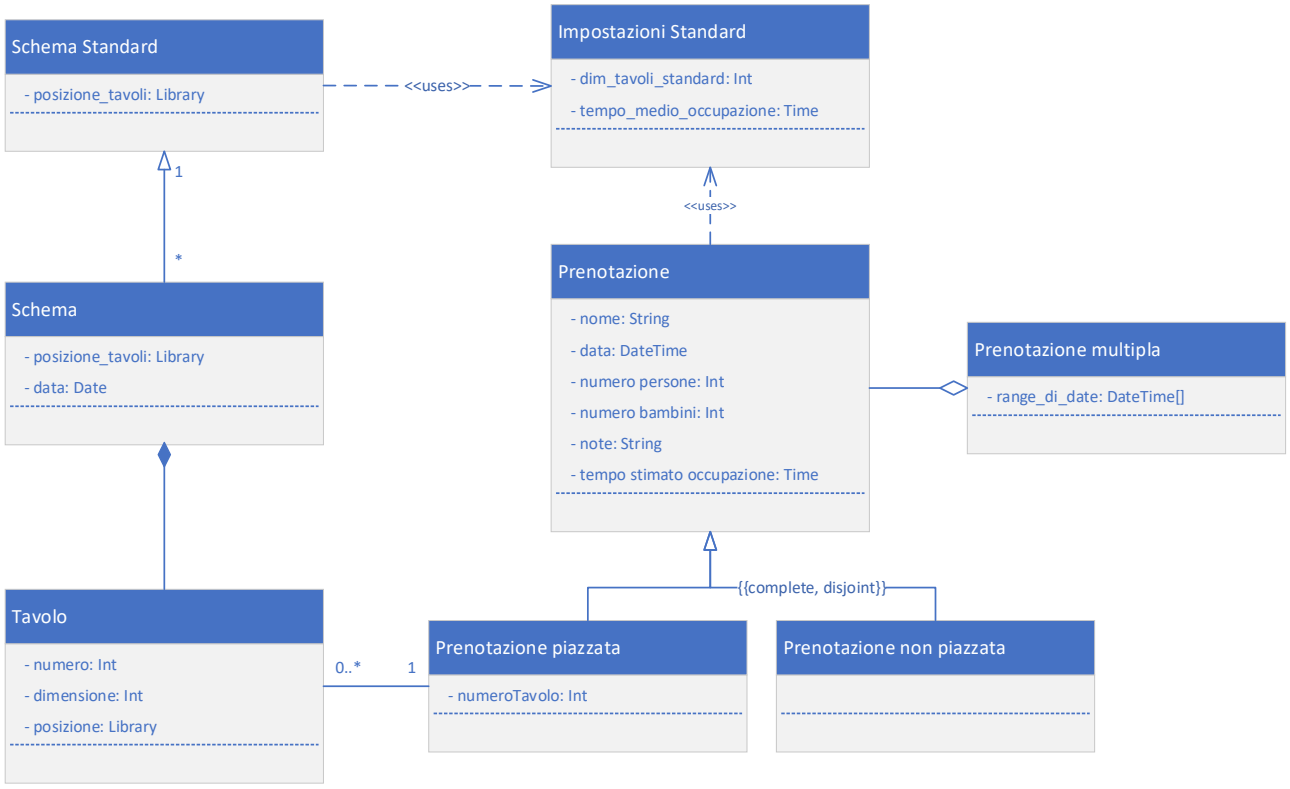


Figura 5 - analisi_del_problema/modello_dominio_prenotazioni.vsdx

Schema

Vediamo esserci l'entità *"Schema"* che eredità da *"Schema standard"*. Il primo, infatti, oltre ad essere associato ad una data, può ridefinire lo schema dei tavoli. La relazione di composizione tra *"Tavolo"* e "Schema" è data dal fatto che un tavolo esiste solamente per una specifica data, quindi all'interno di uno specifico Schema.
Inoltre, è anche presente il tipo "Library"; questo ovviamente si riferisce ad una libreria con il quale definirò la configurazione dei tavoli.

Prenotazioni

Come per gli utenti, è stata usata l'ereditarietà per distinguere tra prenotazioni piazzate o meno. In questo modo è possibile aggiungere l'attributo *"numeroTavolo"* a quei soli tipi di prenotazione che sono stati piazzati ad un tavolo. La relazione di aggregazione tra *"Prenotazione"* e *"Prenotazione multipla"* è data dal fatto che la prima può esistere anche senza la seconda.

Architettura logica: Struttura

Diagramma dei package

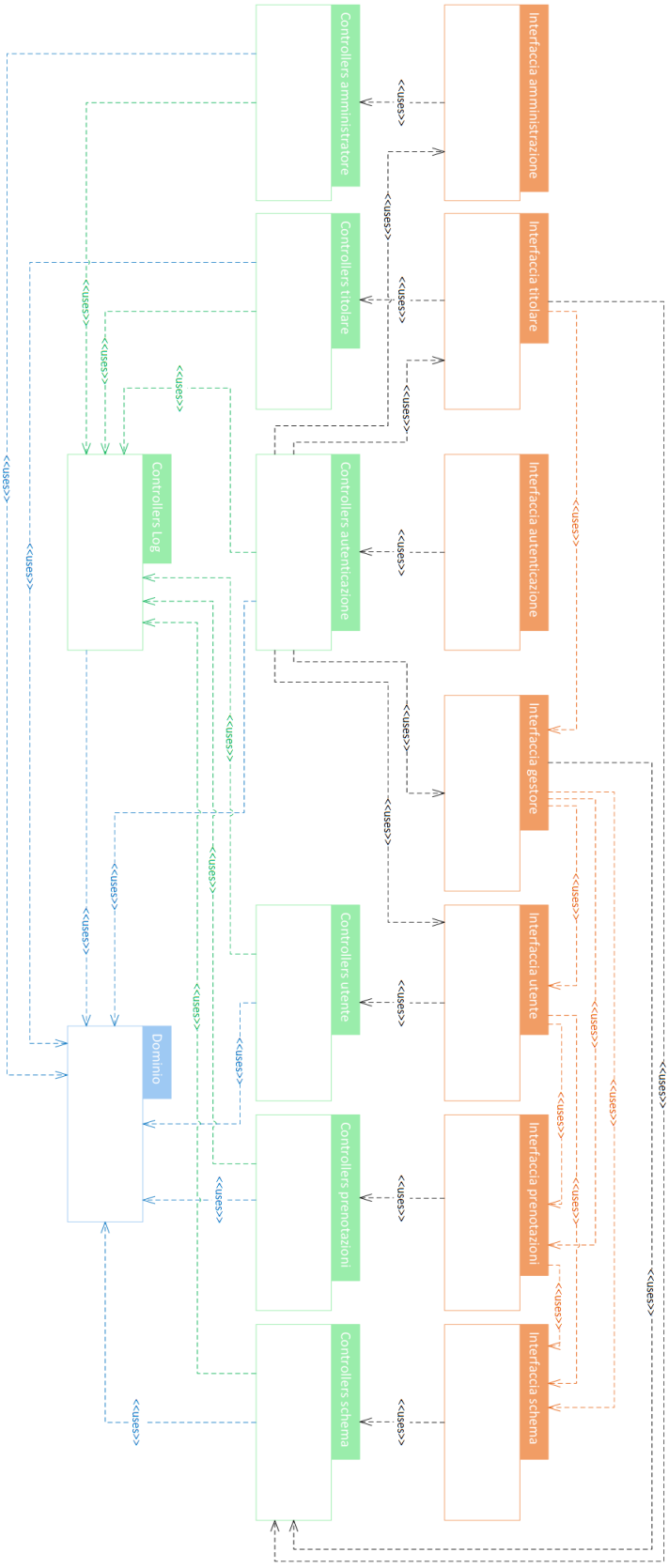


Figura 6 - analisi_del_problema/struttura/diagramma_package.vsdx

Nel diagramma a lato vengono mostrate tutte le interazioni possibili tra i package della nostra architettura logica.

Sono stati usati colori diversi per le frecce in modo da semplificare la comprensione. La suddivisione è avvenuta in base al tipo di interazione:

- Interfaccia – Controller
- Interfaccia – Interfaccia
- Controller – Controller
- Controller – Dominio

Tutti i controller utilizzano il package *"Controllers Log"* in quanto per ogni azione significativa viene creato un nuovo log.

Diagrammi delle classi

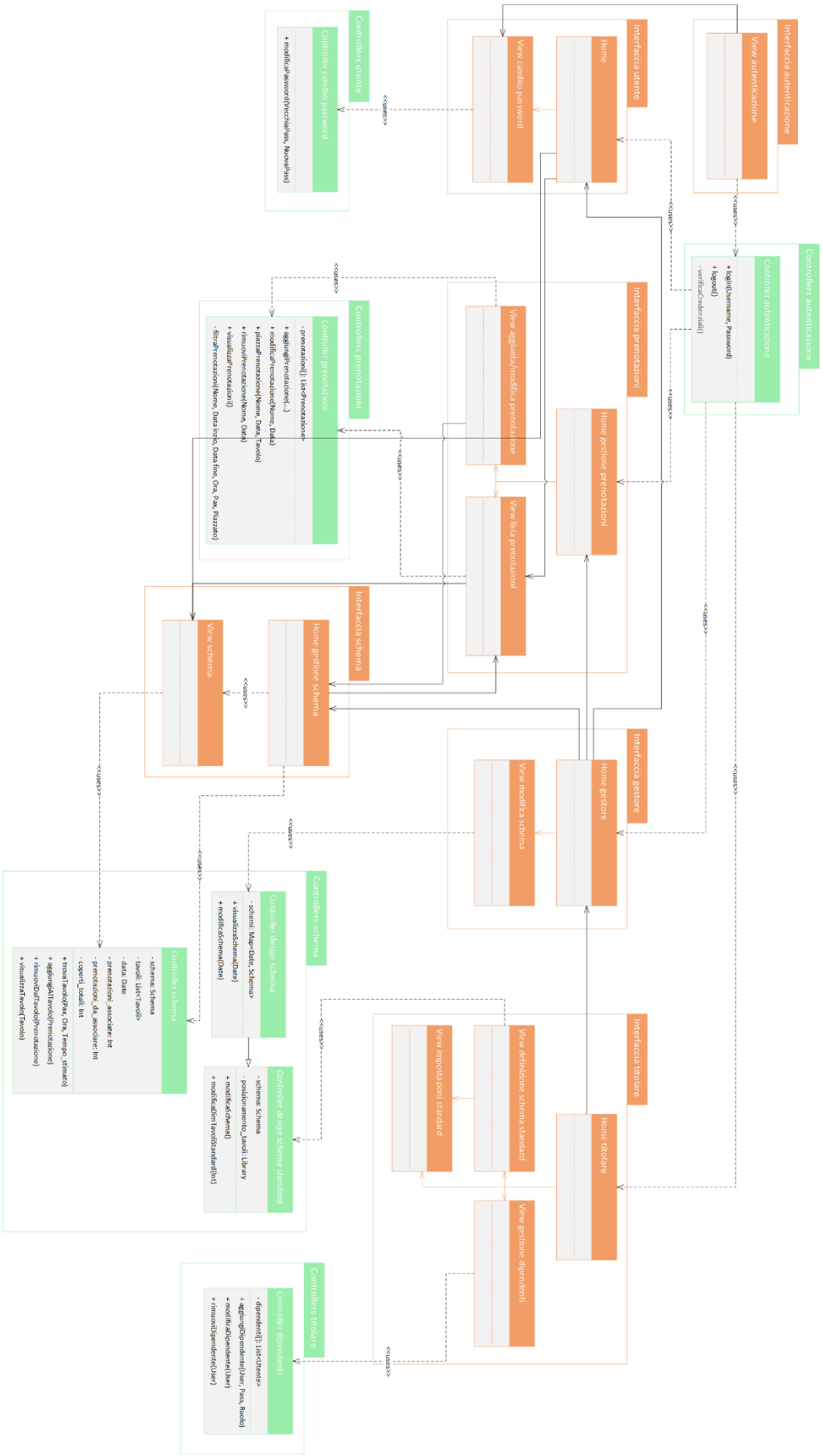


Figura 7 - analisi_del_problema/struttura/classi/classi.vsdx

Nel diagramma mostrato sopra sono presenti tutti i package visti appunto nel diagramma dei package, eccetto per quelli riguardanti l'amministrazione e i log, mostrati più avanti. Sono state dettagliate tutte le possibili interazioni presenti nei package, sia quelle tra le interfacce che tra interfacce e controller.
Non è stato evidenziato nello schema ma tutti i controller utilizzano *"Controllers log"* per aggiungere appunto un log.

Il punto focale è stato quello di rispettare i ruoli degli utenti, in quanto le possibilità di interazione sono diverse. Vediamo che il titolare, oltre alla sua sezione, può interagire con l'interfaccia del gestore, che a sua volta è collegato alla gestione delle prenotazioni e dello schema. *L'utente "normale"* invece, dalla sua interfaccia, può accedere solamente a *"View schema"* e *"View prenotazioni".* Queste, coerentemente con quanto specificato nei requisiti, hanno il solo scopo di mostrare rispettivamente lo schema di un determinato giorno e le prenotazioni, senza la possibilità di apportare modifiche.

Nelle pagine seguenti vengono mostrati i diagrammi delle classi divisi per contesto. Non in tutti è stato mostrato l'uso di "*Controllers log*".

Diagramma delle classi: amministrazione

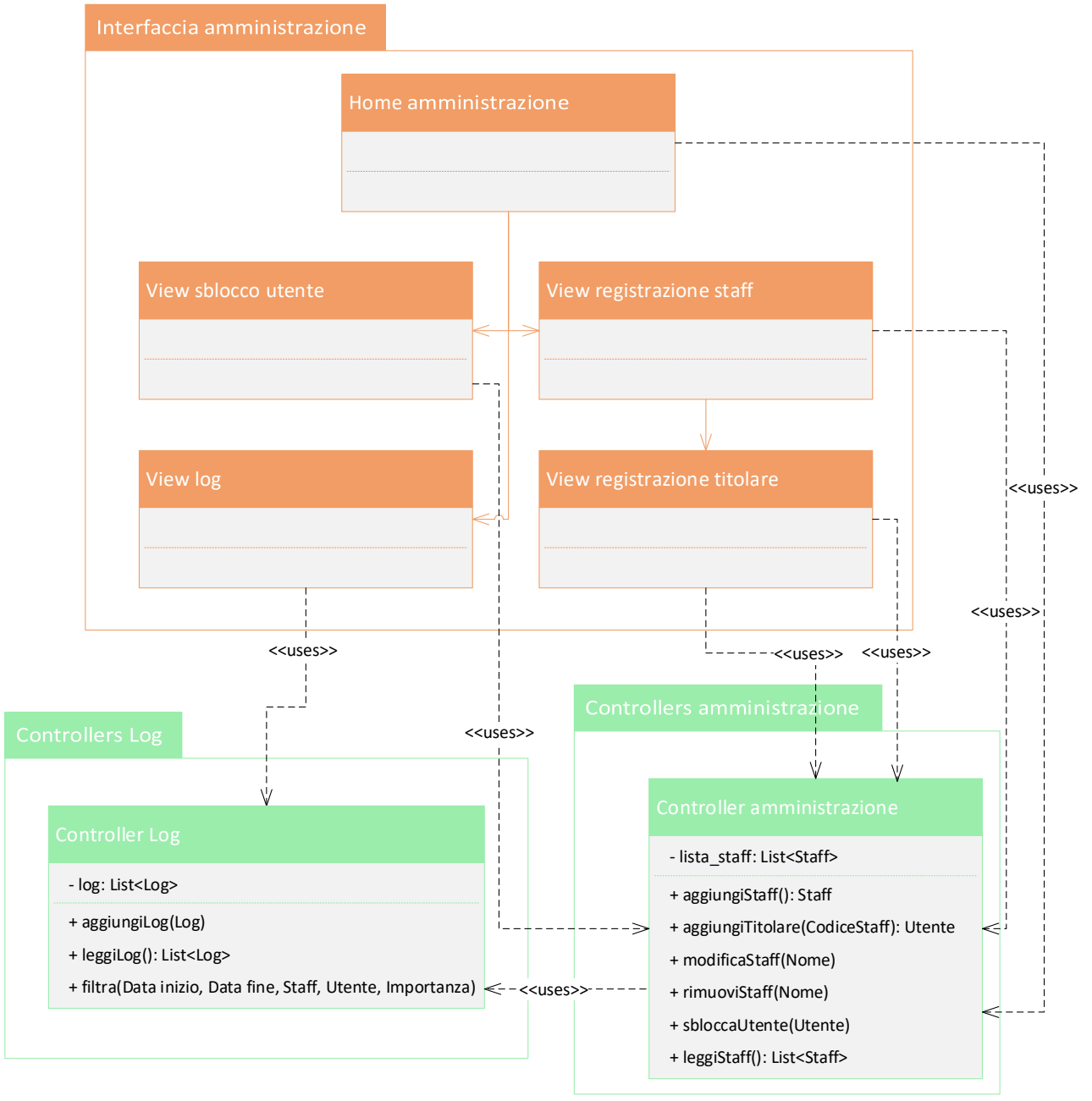


Figura 8 - analisi_del_problema/classi/classe_amministrazione.vsdx

Oltre alle views, anche **“Home amministrazione”** usa il controller. Questo perché è lì che viene mostrata la lista contenente tutti gli staff. Inoltre, “Interfacci amministrazione” è l’unica che al suo interno usa “Controller log”, essendo che in “View log” viene mostrata la lista di tutti i log.

Diagramma delle classi: autenticazione

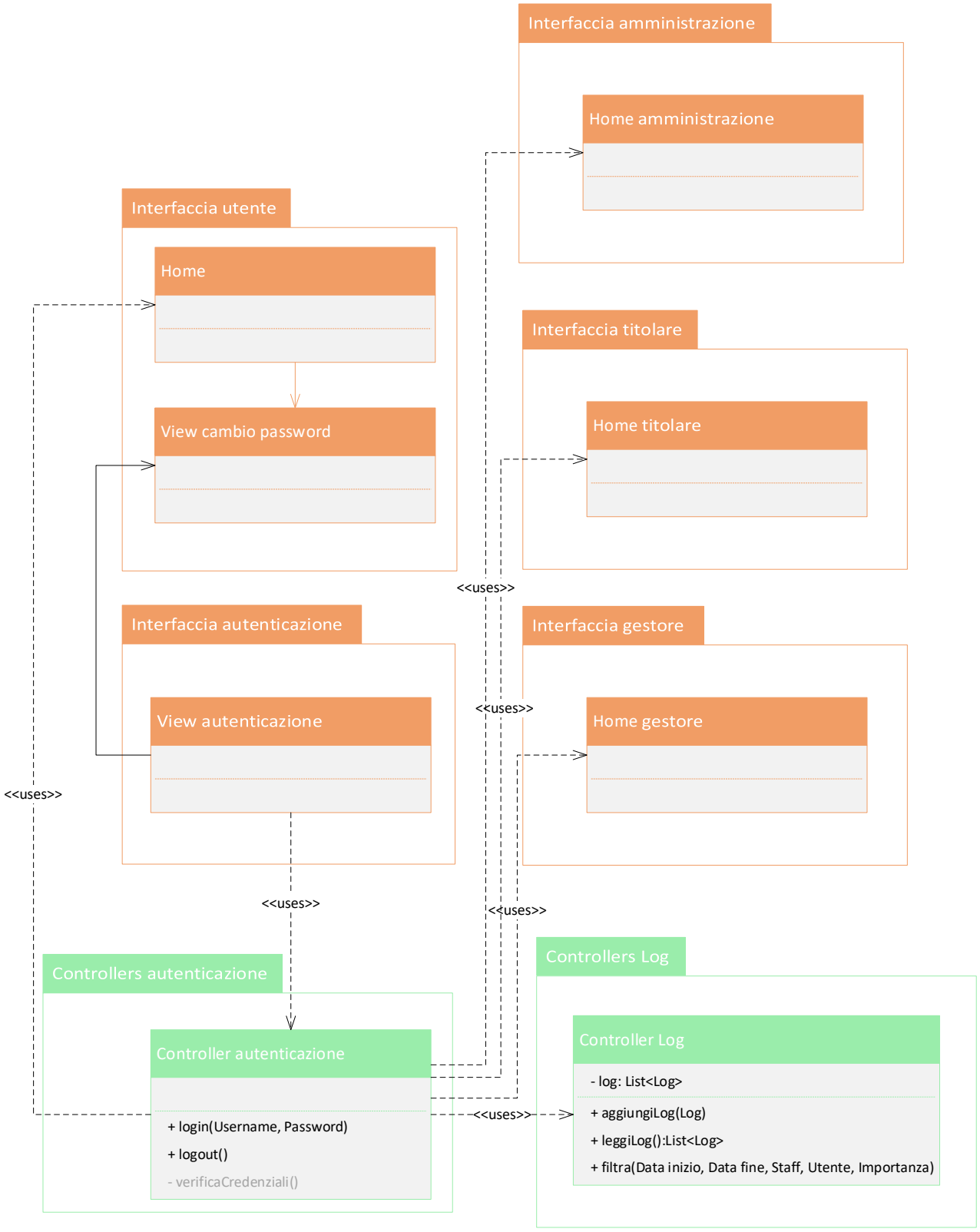


Figura 9 - analisi_del_problema/classi/classe_autenticazione.vsdx

Il controller autenticazione ha il compito di riconoscere il ruolo dell'utente e di indirizzare quest'ultimo all'interfaccia consona.

Diagramma delle classi: gestore

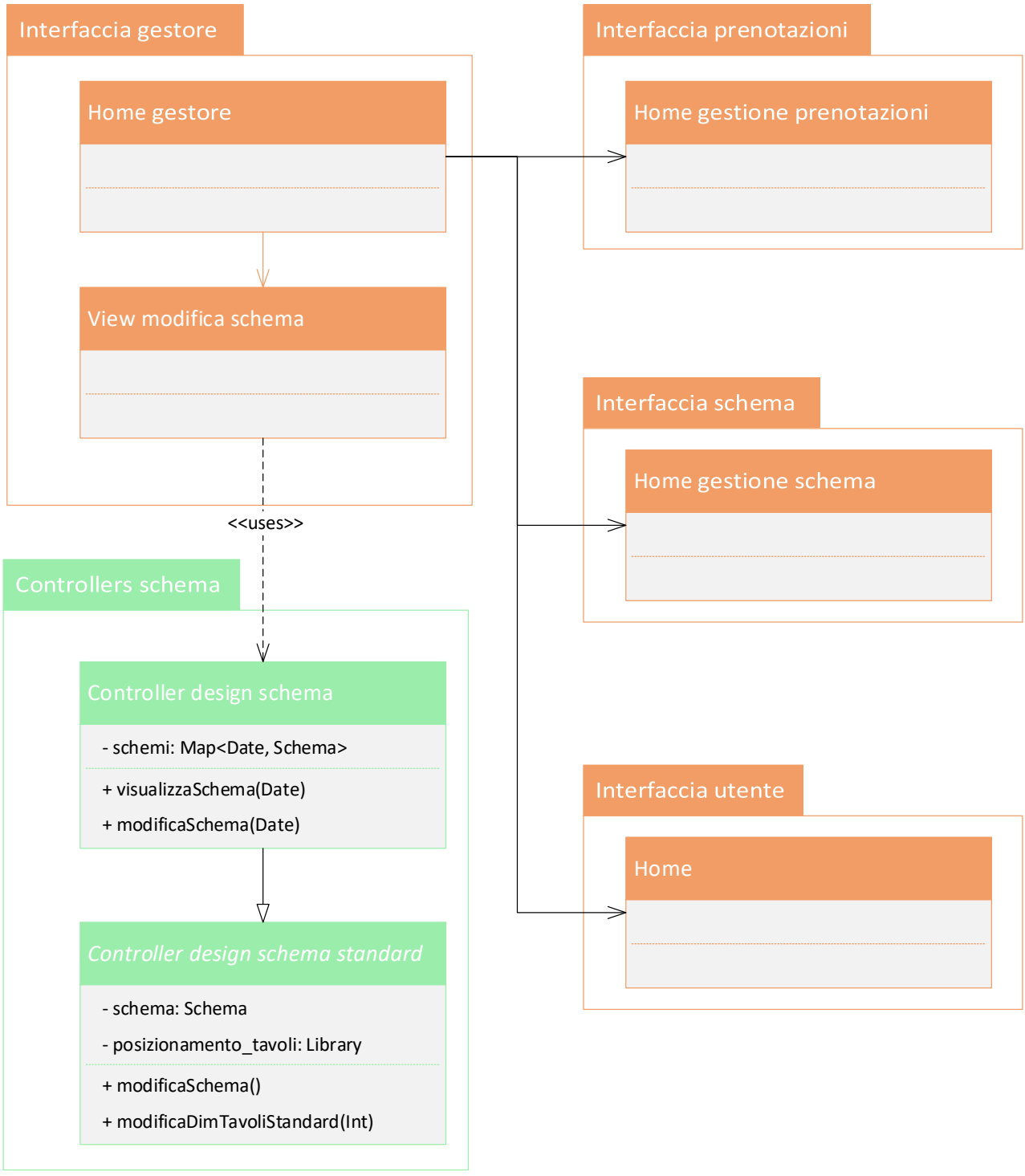


Figura 10 - analisi_del_problema/classi/classe_gestore.vsdx

Vediamo che la definizione del design dello schema e dello schema standard sono stati affidati a due controller differenti. In questo modo vi è una più netta separazione tra quello che può fare *l'utente "gestore"*, modificare solo schema di un determinato giorno, e *l'utente "titolare",* definire lo schema standard.

Diagramma delle classi: titolare

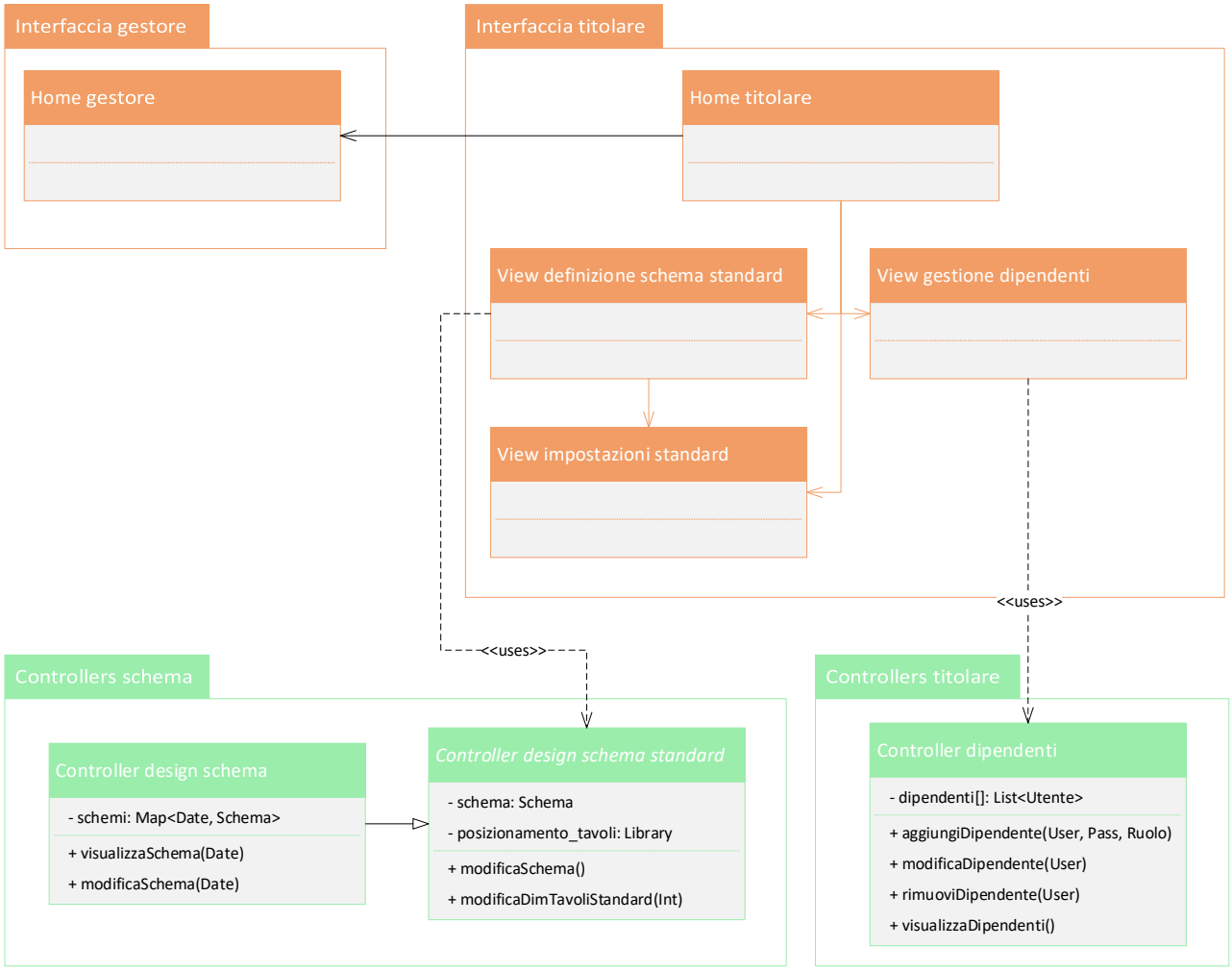


Figura 11 - analisi_del_problema/classi/classe_titolare.vsdx

A differenza del caso precedente, qui abbiamo *l'utente "titolare"* che dalla relativa interfaccia può usare **"Controller design schema standard"** e definire appunto lo schema standard. Ovviamente, tramite l'interfaccia del gestore alla quale può accedere, ha la possibilità di modificare lo schema di un determinato giorno. Più in generale possiamo dire che ha disponibili tutte le interfacce ed i controller ai quale *l'utente "gestore"* può accedere.

Diagramma delle classi: utenti

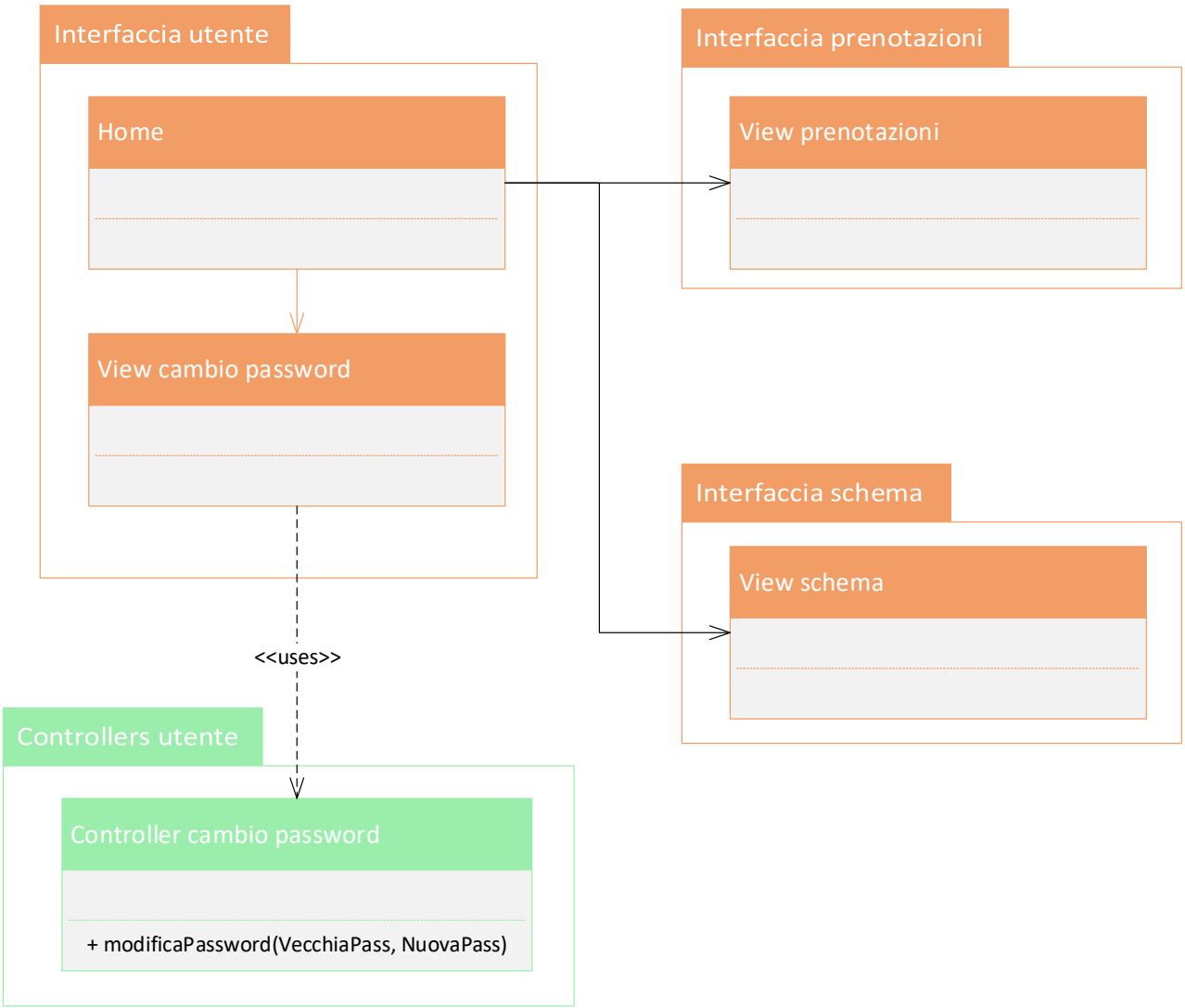


Figura 12 - analisi_del_problema/classi/classe_utenti.vsdx

Nel grafico viene evidenziato più chiaramente come *l'utente "normale"*, oltre al cambio password, ha solo la possibilità di visualizzare schema e prenotazioni senza effettuare prenotazioni.

Diagramma delle classi: prenotazioni e schema

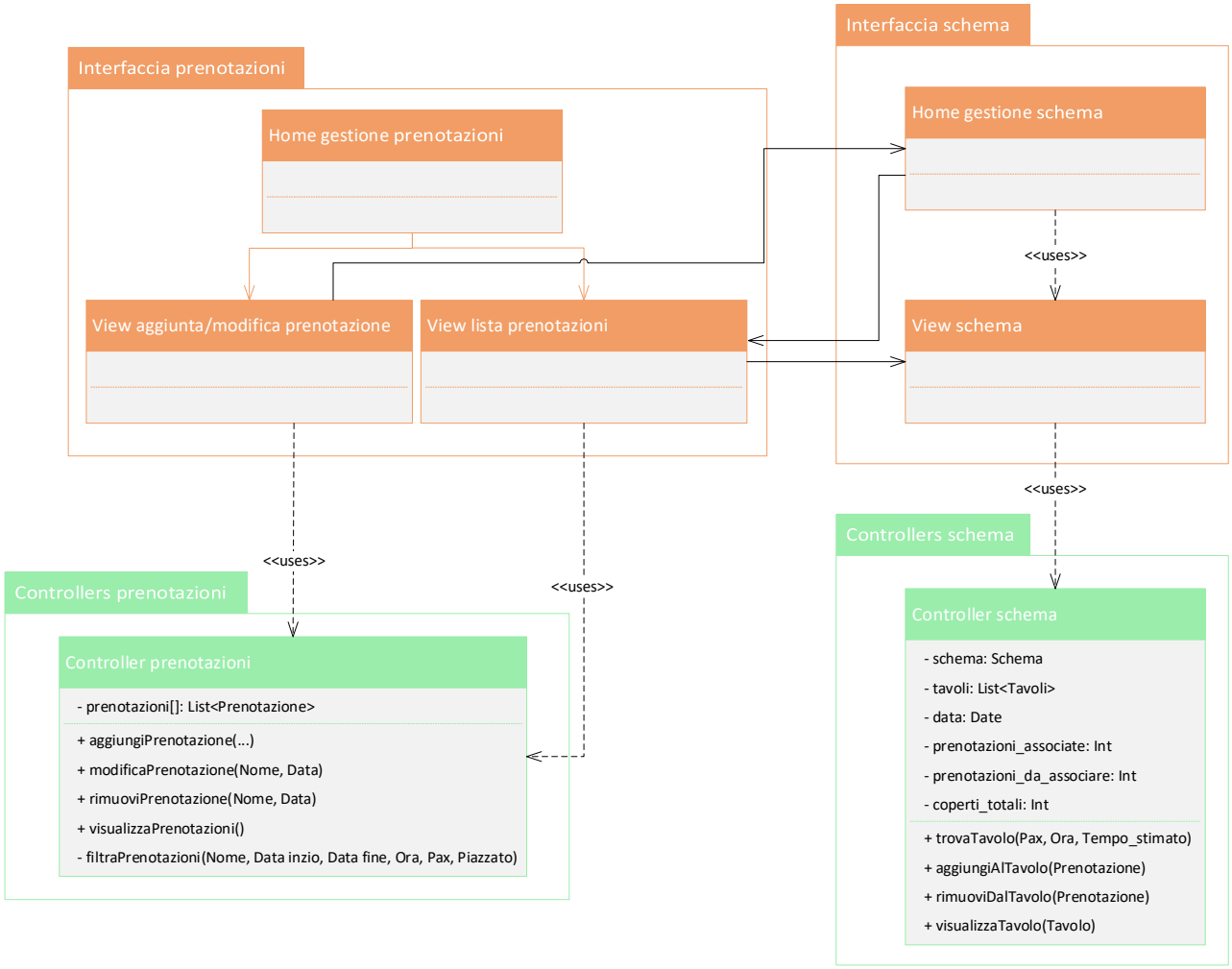


Figura 13 - analisi_del_problema/classi/classe_prenotazioni.vsdx

Si vede come durante l'aggiunta o la modifica di una prenotazione è possibile visualizzare ed effettuare modifiche allo schema. Durante la gestione dello schema si può poi vedere la lista delle prenotazioni relative a quel determinato giorno e dalla lista di prenotazioni è possibile visualizzare uno schema.

"View lista prenotazioni" è collegato a **"View schema"** e non a "Home gestione schema" per rimanere coerenti con l'assegnamento dei ruoli; infatti, in questo modo, *l'utente "normale"* non può effettuare una "privilege escalation" passando per la vista delle prenotazioni.

È anche presente un ulteriore controller del package "*Controllers schema*", il cui compito è quello di gestire le occupazioni ai tavoli e non coinvolge l'aspetto dello schema dei tavoli.

Architettura logica: Interazione

Diagrammi di sequenza

Diagramma di sequenza: autenticazione utenti

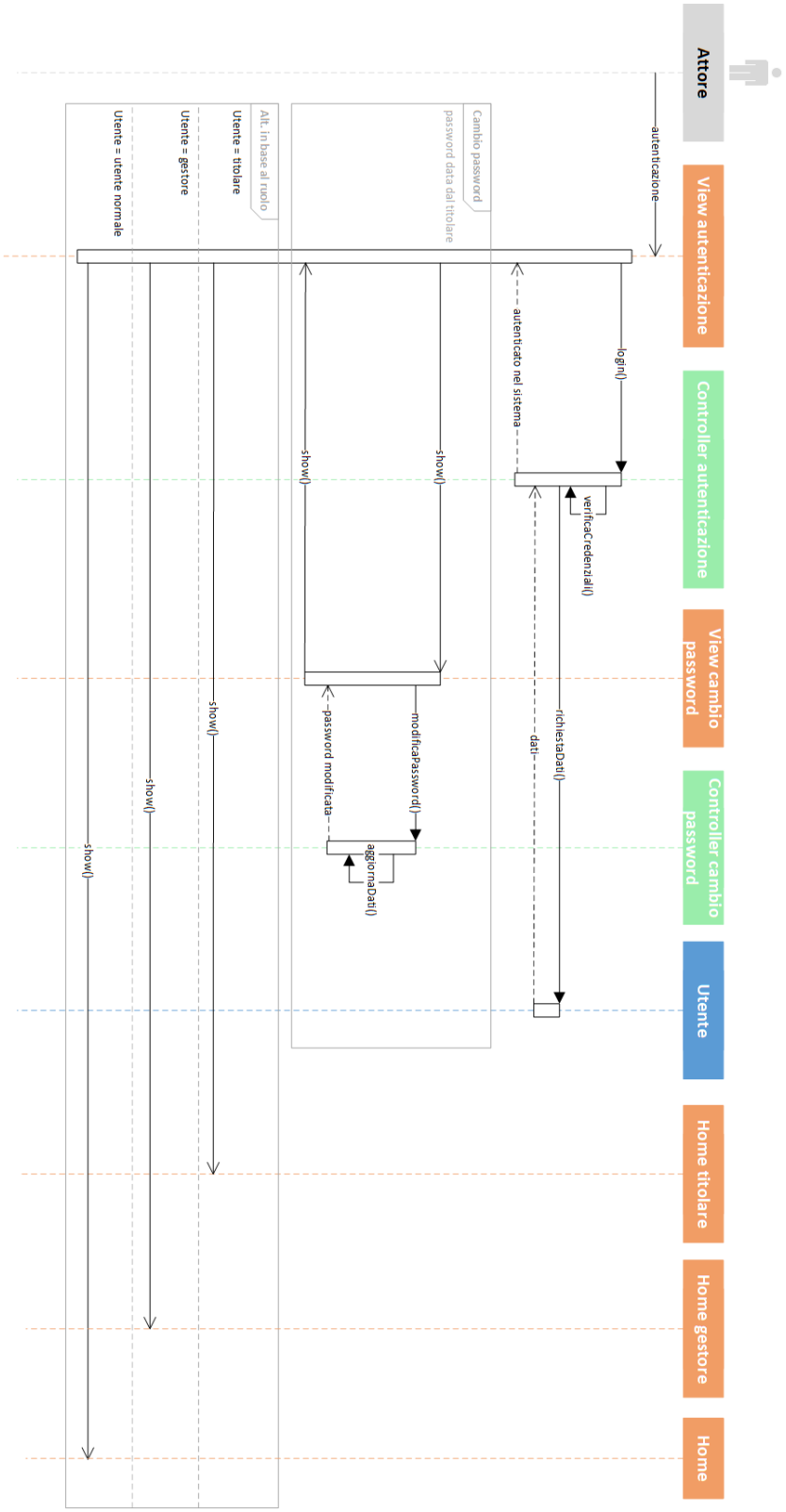


Figura 14 - analisi_del_problema/interazione/sequenza_autenticazione_utenti.vsdx

Diagramma di sequenza: amministrazione

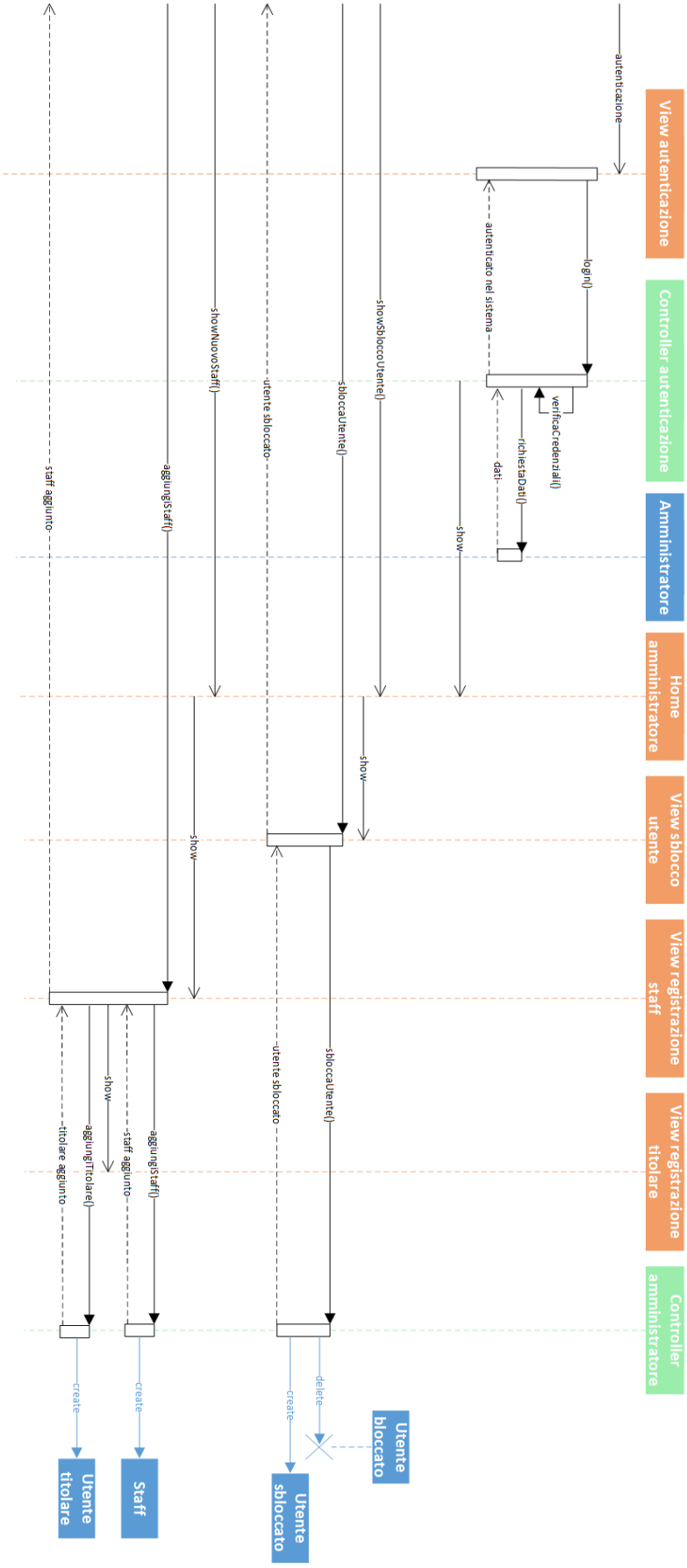


Figura 15 - analisi_del_problema/interazione/sequenza_amministrazione.vsdx

Diagramma di sequenza: aggiunta prenotazione

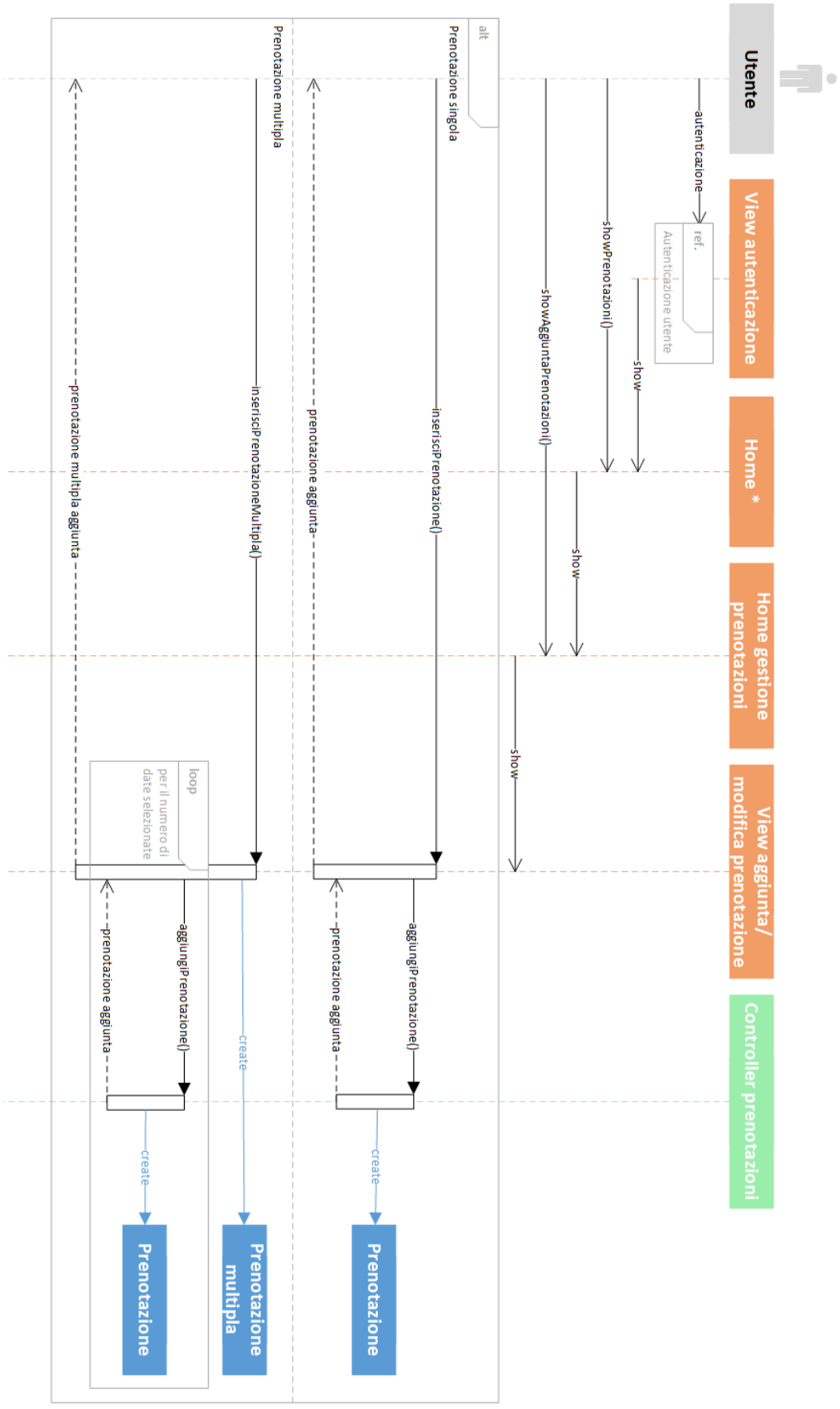


Figura 16 - analisi_del_problema/interazione/sequenza_aggiunta_prenotazione.vsdx

***Home ** simboleggia la Home View specifica di ciascun tipo di utente, in base al ruolo.

Diagramma di sequenza: assegnamento tavolo

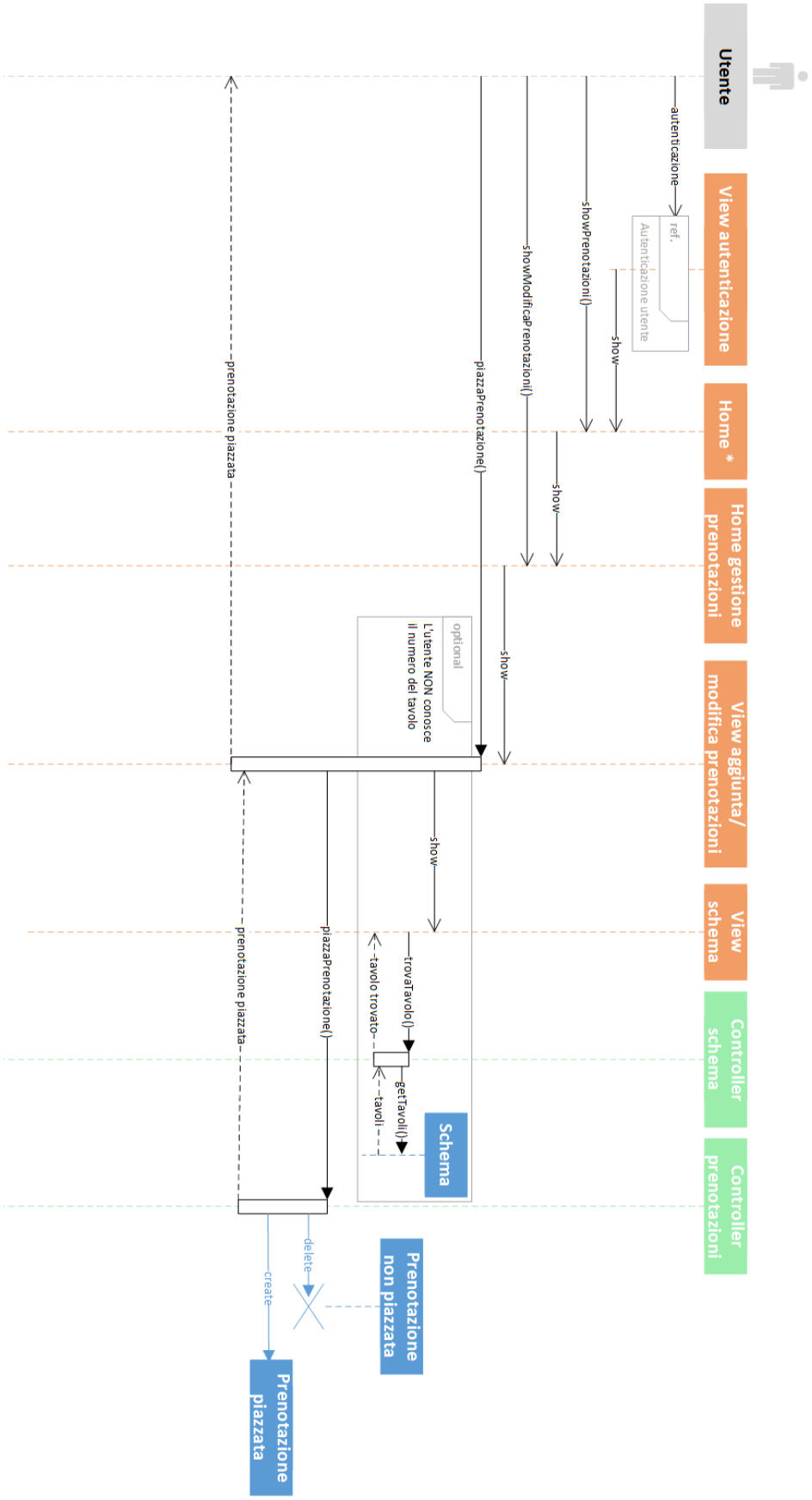


Figura 17 - analisi_del_problema/interazione/sequenza_assegnamento_tavolo.vsdx

Architettura logica: comportamento

Diagrammi di attività

Diagramma di attività: uso e definizione dello schema

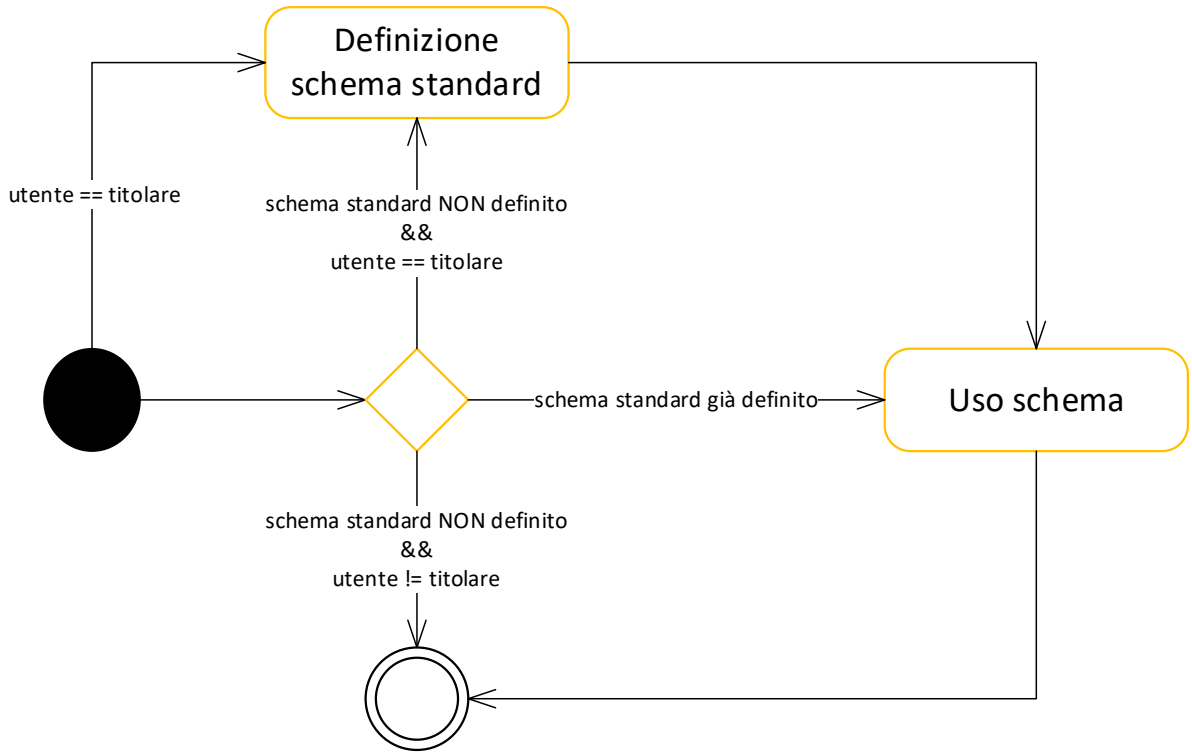


Figura 18 - analisi_del_problema/comportamento/attività_definizione_schema.vsdx

In questo diagramma viene spiegato in dettaglio come deve avvenire l'inizializzazione dello schema. Vediamo che l'uso dello schema è vincolato dal fatto che lo *schema standard* sia già stato definito. Essendo che quest'ultima azione può essere eseguita solamente dal titolare, dovrebbe essere fatta appena si inizia ad usare il software.

Diagramma di attività: piazzamento prenotazioni

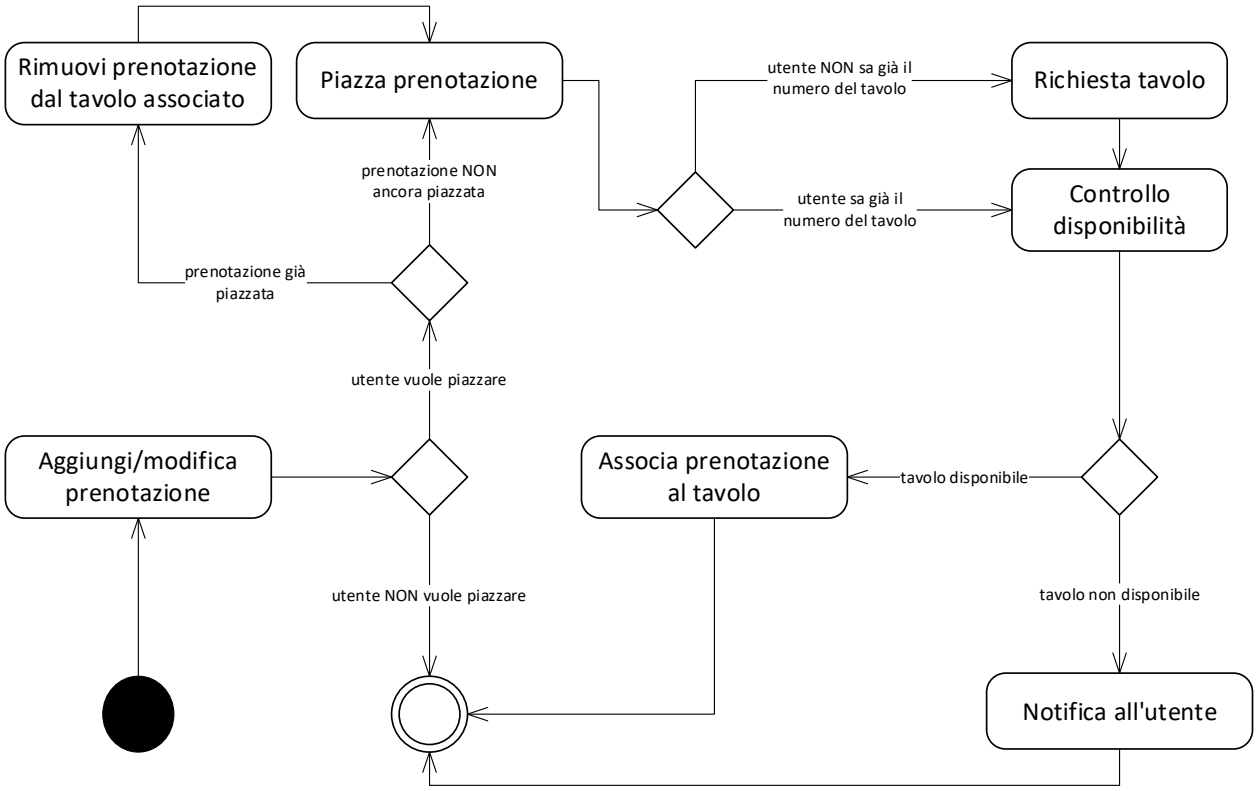


Figura 19 - analisi_del_problema/comportamento/attività_piazzamento_prenotazioni.vsdx

Quando un utente vuole aggiungere o modificare una prenotazione può anche decidere se piazzarla o meno ad un tavolo. Nel primo caso, se la prenotazione era già stata piazzata ad un tavolo, questa associazione va rimossa e si può poi procedere con un nuovo assegnamento.

Inoltre, sia che il tavolo sia stato inserito manualmente dall'utente sia che venga fornito dal sistema, bisogna controllare che sia disponibile; in caso non lo sia l'utente viene notificato.

Progettazione

Progettazione architetturale

Requisiti non funzionali

Dall'analisi dei requisiti sono emersi alcuni vincoli non funzionali da tenere in conto, tra cui:

- Usabilità
- Tempi di risposta bassi
- Protezione ed integrità dei dati
- Controllo degli accessi
- Disponibilità dei servizi
- Replicabilità del sistema
- Basso costo del prodotto software

L'usabilità e i tempi di risposta bassi sono dei requisiti fondamentali per il software in quanto è necessario che l'interazione con esso risulti facile ad utenti senza esperienza informatica. Si pensa ad un'applicazione da usare in un ambiente frenetico, come appunto quello della ristorazione, che sia quindi rapida ed intuitiva da usare. Con questi vincoli si scontrano quelli riguardanti la protezione e l'integrità dei dati e il controllo degli accessi; sappiamo essere fondamentali in quanto delle problematiche derivanti dal software possono portare a perdite economiche e di immagine da parte dall'azienda. L'implementazione delle politiche di sicurezza e protezione deve però essere realizzata tramite soluzioni standard esterne al sistema, sia per rispettare il vincolo di basso costo del prodotto software, sia per la sicurezza che queste possono dare in quanto appunto standard. Data la facilità di utilizzo ed il vasto supporto ho optato per sfruttare il protocollo HTTPS per la comunicazione sicura, in quanto utilizza TLS come protocollo di cifratura sicuro. Infine, la disponibilità dei servizi e la replicabilità del sistema servono a contenere un possibile attacco DoS e possono essere garantiti tramite un sistema di back-up gestito dall'amministratore, replicando il server su più host.

Scelta dell'architettura

L'architettura del sistema è sviluppata su tre livelli seguendo il **modello client/server**. I **tre livelli** in questione sono la base dei dati, un software middleware che funge da server e quindi da intermediario tra i due estremi, e l'applicativo lato client. Si sarebbe potuto pensare ad un sistema a quattro livelli, introducendone uno per la gestione della persistenza, ma questo avrebbe portato ad un rallentamento dell'intera applicazione.

Livello Front-end

Questo livello è quello che va a contatto con l'utilizzatore finale ed ha il solo scopo di rappresentare le informazioni reperite dallo strato server intermedio. Il livello è pensato per essere altamente specializzato e multipiattaforma: ci possono essere più implementazioni che utilizzano in egual modo la piattaforma sottostante. Non è presente logica applicativa, solo collegamenti alle funzioni disponibili lato server.

Livello Middleware

Qui viene gestita tutta la logica applicativa e l'interazione con il database. Sulle informazioni fornite da questo livello si basa la visualizzazione fornita all'utente finale nel livello di front-end. Da quest'ultimo invece il livello middleware riceve le informazioni da elaborare ed eventualmente aggiorna di conseguenza il database.

Livello Persistenza

Ho scelto di adottare un database relazionale indipendente per la persistenza dei dati. Viene utilizzato il **pattern DAO** per effettuare il mapping tra il mondo ad oggetti e quello relazionale della base di dati. Ho scelto di gestire la persistenza in questo modo come via di mezzo tra la metodologia a Forza Bruta, poco estendibile e macchinosa, e l'utilizzo di un framework ORM, comodo ma avrebbe aumentato i tempi di risposta. Per quanto riguarda la disponibilità e la replicabilità del sistema ho pensato di effettuare un backup locale alle 12 e uno alle 19:30, prima di ogni possibile servizio. In questo modo, se qualcosa andasse storto, l'amministratore di sistema potrebbe replicare il servizio su altri host, che deve comunicare ai vari client.
Per motivi di sicurezza si pensava di utilizzare un secondo database per la gestione dei log, ma ho optato infine per appoggiarmi al filesystem.

Patterns & Design Principle

Ho adottato il pattern **Broker**, che rende facile la scalabilità orizzontale dello strato intermedio. Con questa architettura, in un futuro, avrò la possibilità di suddividere l'attuale server in più componenti, o anche aggiungerne uno nuovo, affidando l'interfacciamento al Broker, senza che i vari Client si accorgano di nulla. In più, in caso di replicazione del server da parte dell'amministratore, basterà fornire al Broker il nuovo URL sul quale effettuare le richieste. L'adozione del pattern Broker, inoltre, permette di applicare **The Dependency Inversion Principle** disaccoppiando i client dallo specifico middleware, dato che viene definito a priori il protocollo di comunicazione.
Come detto precedentemente, ho pensato di concentrare tutta la logica applicativa all'interno del servizio middleware. Per questo ho scelto di adottare il pattern architetturale **MVP**, con una parte di view minima, passiva, volta solo alla comunicazione lato front-end. Per quanto riguarda il lato front-end ho invece deciso di adottare il pattern **MVVM**, effettuando il binding tra View e View-Model tramite pattern **Observer**.

Diagramma dei package

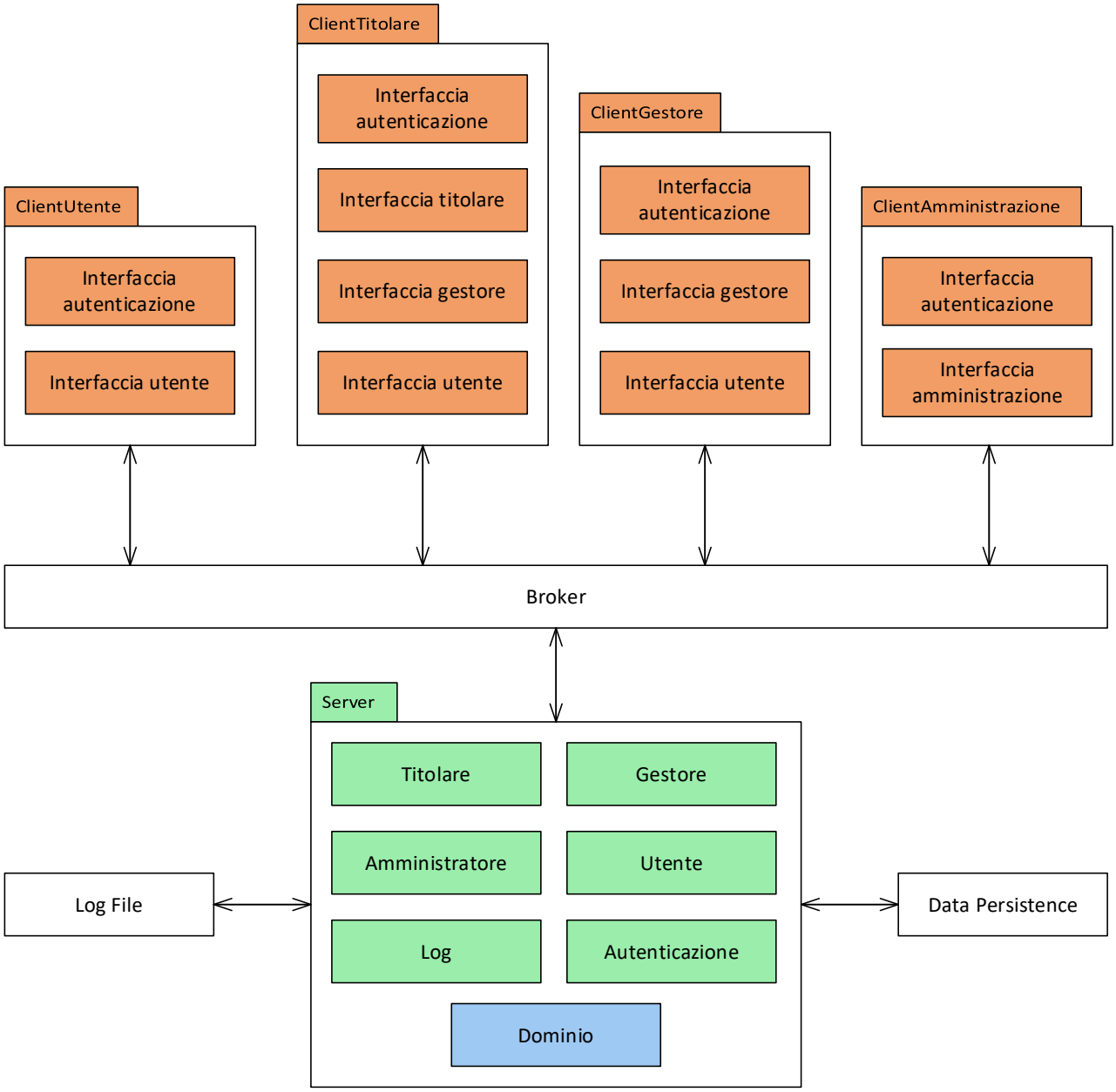


Figura 20 - progettazione/architettura/diagramma_package.vsdx

Ogni client, eccetto quello relativo all'amministratore, ha al suo interno il package *Client Utente*, che comprende tutte quelle funzionalità disponibili agli utenti di qualsiasi ruolo. Avrei potuto ridefinire i metodi del package *Client Utente* negli altri, così da non doverlo reinserire in ogni client; ho pensato però fosse meglio in questo modo per separare gli ambiti in maniera più ligia e per poter riutilizzare quei package anche in vista di modifiche future.

Esempio: voglio aggiungere una nuova funzionalità per l'utente base. Utilizzando l'architettura sopra definita mi basterebbe aggiungere la funzionalità al package *Client Utente*, rendendola disponibile in automatico a tutti i client che hanno al suo interno il suddetto package.

Diagramma a componenti

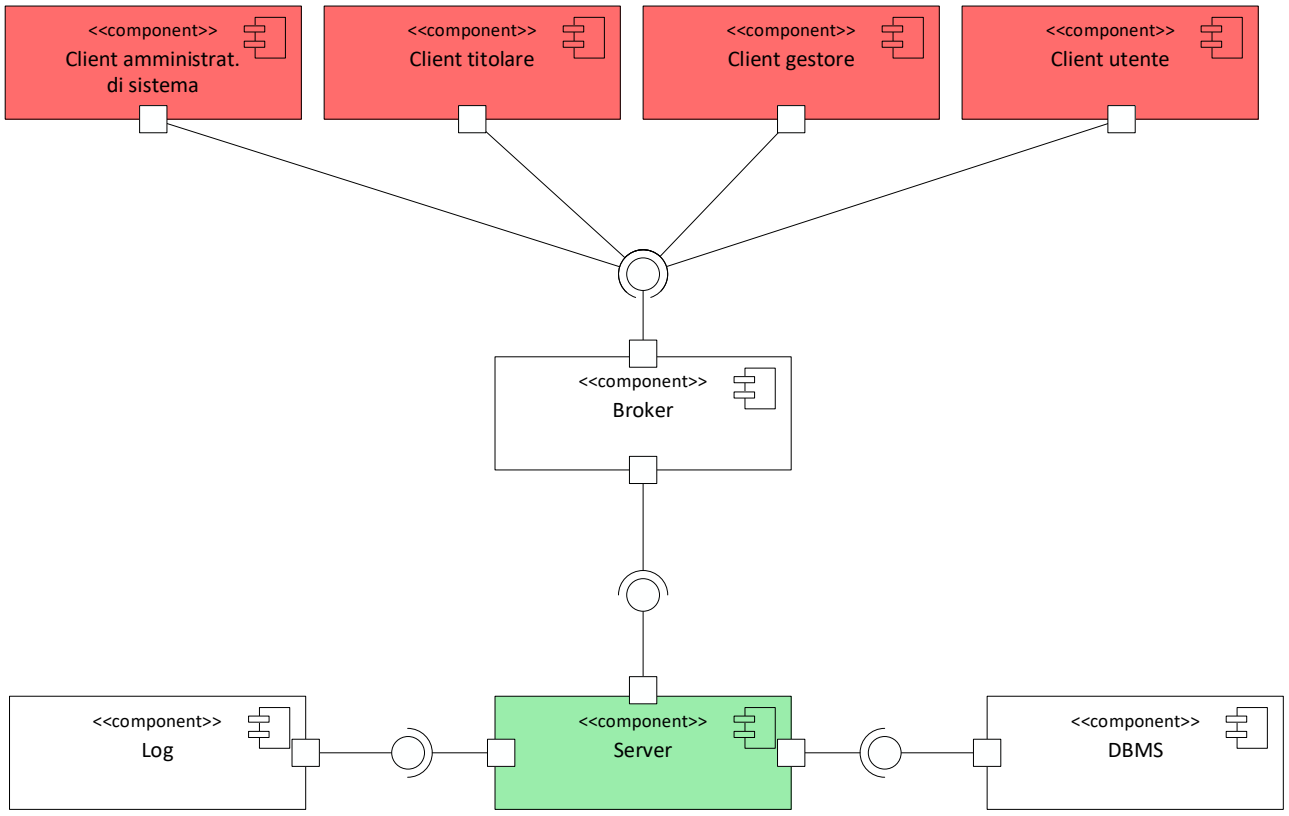


Figura 21 - progettazione/architettura/diagramma_componenti.vsdx

Strutturando le componenti software in questo modo si avrà la possibilità di installare dei client specifici in base al tipo di utente finale che li dovrà utilizzare. Per quanto riguarda la componente middleware ho pensato di definirne solo una da installare, in quanto le funzionalità sono ridotte e non è necessario separare più server in base alle funzionalità.

Scelte tecnologiche

Ho deciso di utilizzare come protocollo di sicurezza e trasmissione dei dati il **protocollo HTTPS**: la sicurezza è garantita dal layer TLS, che assicura una comunicazione sicura end-to-end. Inoltre, data la grande diffusione di questa tecnologia, nel futuro un possibile servizio esterno potrebbe interfacciarsi con l'applicativo middleware, senza riscontrare troppe difficoltà. Per il formato dei dati ho invece optato per **JSON**, dato il grande supporto e la sua facilità di utilizzo.

Il pattern broker sarà implementato non come una componente a sé stante, ma sotto forma di package sia nel component server sia nei componenti client. Ogni client avrà quindi il suo *GestoreInterrogazioni* che, in base alle configurazioni di deployment dove viene specificato un URL (o più), si connetterà al package broker del livello di middleware, che a sua volta smisterà le richieste al controller adeguato.

Per quanto riguarda l'applicativo front-end userò dei framework basati su JavaScript, come ad esempio ReactJS o Angular. Questa scelta è stata influenzata anche dalla grande presenza di librerie per il disegno di figura geometriche che si basano appunto su JavaScript. Io le userò per la definizione dello schema dei tavoli, semplicemente inserendo dei rettangoli numerati all'interno di una pagina vuota.

Progettazione di dettaglio

Struttura

Nonostante le funzioni siano più o meno le stesse viste in fase di analisi, la struttura del software è cambiata considerevolmente per rispettare i design pattern e per dividere i due livelli, server e client.

Struttura middleware

La struttura di questo livello segue il modello MVP:

- **Model**: definizione del modello dei dati. Qui viene anche implementato il pattern DAO, con un'interfaccia e una classe concreta per ogni DTO del modello del dominio.
- **View**: vista passiva, che ha il solo scopo di inviare la risposta al lato client.
- **Presenter**: gestione della logica applicativa con elaborazione del modello dei dati e conseguente trasmissione alla View.

Package application.middleware.model.dao

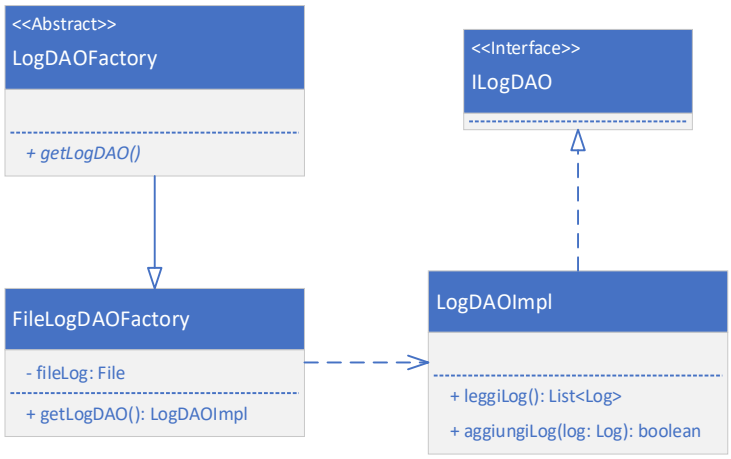


Figura 22 - progettazione/struttura/middleware/model_dao.vsdx

Anche per la persistenza dei log è stato applicato il pattern DAO. A differenza del resto delle entità il pattern viene applicato al **filesystem**, ma, grazie alla classe astratta LogDAOFactory, risulterebbe semplice un'eventuale implementazione per un data source differente.

Package application.middleware.model.dao

Figura 23 - progettazione/struttura/middleware/model_dao.vsdx

Nel diagramma sopra riportato sono presenti tutte le classi relative all'implementazione del pattern DAO. Per comodità di lettura non sono stati inseriti i metodi CRUD, presenti in ognuna delle implementazioni, ma è stata usata l'interfaccia **CRUDInterface** che viene estesa da ognuna delle interfacce DAO. In fase di implementazione dovrà essere effettuato un controllo sul tipo dell'oggetto passato. Ovviamente, pur non essendo riportata, ogni interfaccia ha al suo interno i metodi definiti all'interno delle relative implementazioni.

Package application.middleware.model.dto

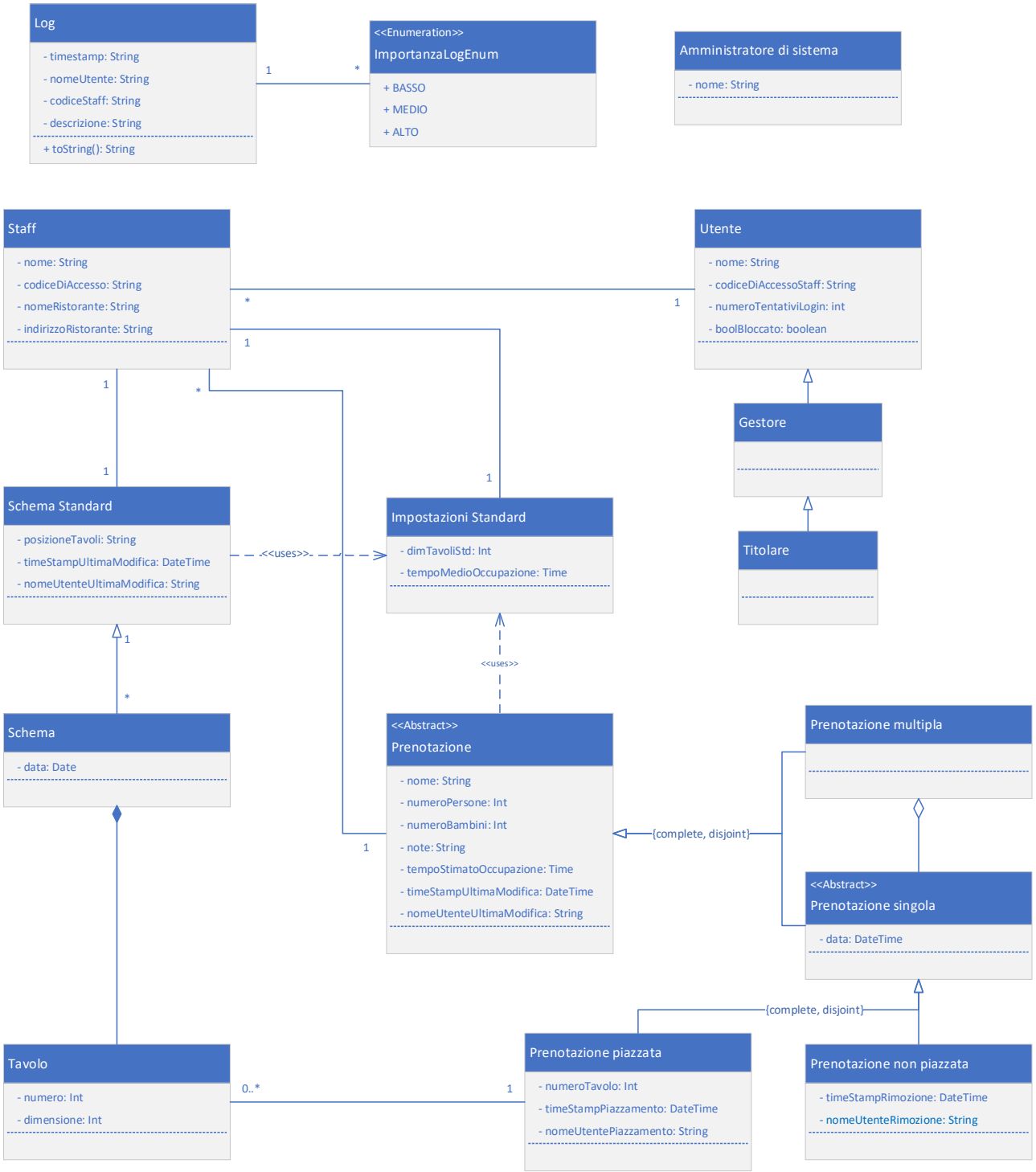


Figura 24 - progettazione/struttura/middleware/model_dto.vsdx

Nel pattern DAO questi oggetti sono chiamati **DTO** (data transfer object). Vediamo che per quanto riguarda Utente ed Amministratore di Sistema, oltre ad essere stato aggiunto l'attributo *nome*, sono stati rimossi username e password. La gestione delle credenziali viene fatta nel package DAO e quindi il modello di dominio non ha senso che contenga informazioni sulle credenziali. La proprietà *posizione_tavolo* si riferisce ad un file che definisce la rappresentazione dei tavoli.

!! Per quanto riguarda le credenziali di accesso dell'amministratore di sistema, queste vanno definite direttamente dallo stesso inserendo una tupla manualmente all'interno del database.

Package application.middleware.view

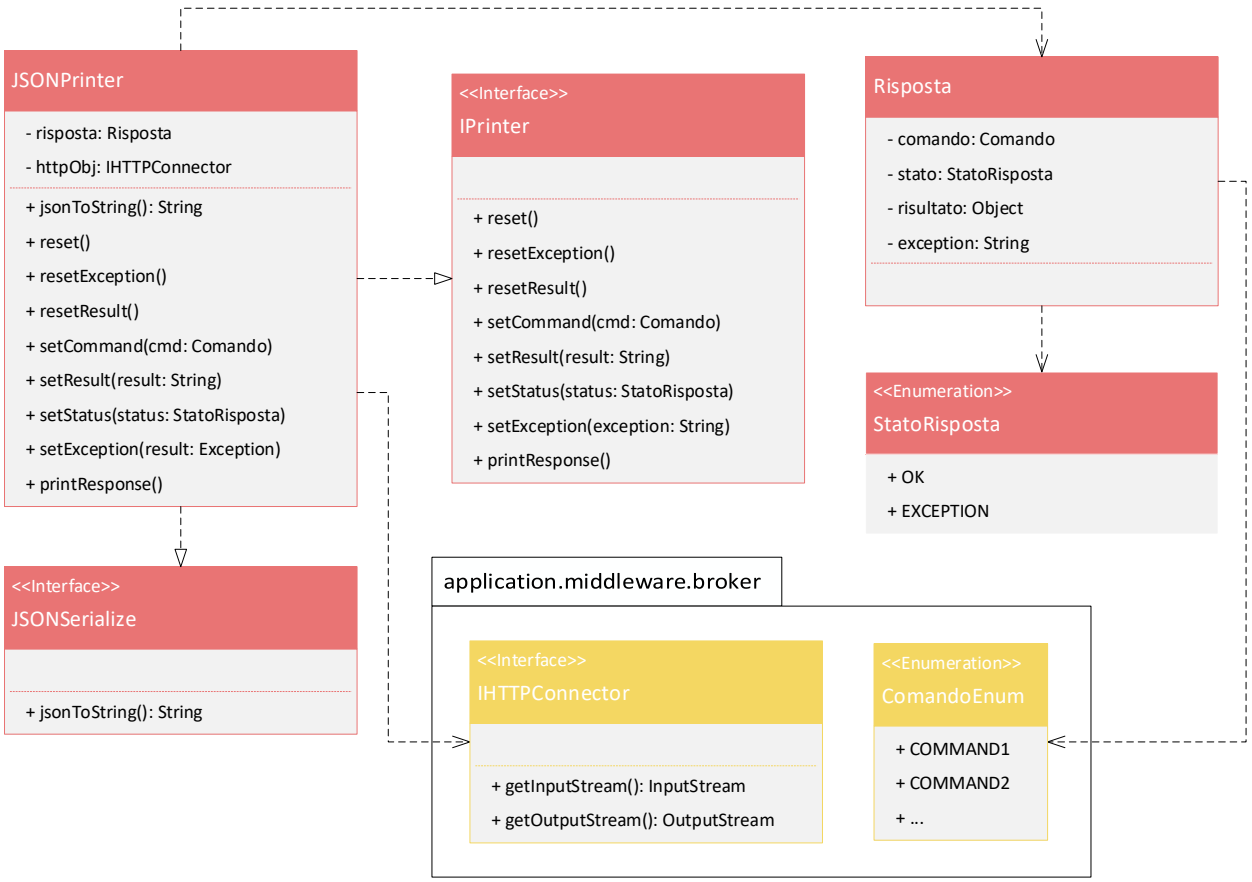


Figura 25 - progettazione/struttura/middleware/view.vsdx

Questo package si occupa di trasmettere i dati dal middleware fino all'applicativo front-end, passando ovviamente per il broker. Per questo motivo la view è stata precedentemente definita passiva.

Essendo che si è scelto di utilizzare JSON come formato dei dati è stata definita la classe **JSONPrinter** che implementa le interfacce **IPrinter** e **JSONSerialize**. In questo modo è sufficiente definire un'altra interfaccia per l'implementazione di una classe che usa un diverso formato dei dati.
La classe JSONPrinter si collega inoltre al broker per ricevere lo stream di output sul quale trasmettere la risposta.
Quest'ultima è una classe che a sua volta ricava dal package broker la lista di comandi disponibili, in modo da poter garantire tracciabilità tra la richiesta ricevuta dal client e la risposta fornita.
Risposta, oltre al risultato già serializzato sotto forma di stringa, usa l'enumeratore **StatoRisposta** per identificare l'esito e, in caso di errore, definisce anche un'eccezione, anch'essa già serializzata.

Package application.middleware.controller

Figura 26 - progettazione/struttura/middleware/controller.vsdx

AbstractController rappresenta una classe estesa da tutti i controller del livello middleware, evidenziati nei diagrammi successivi con lo stereotipo <<Controller>>. Oltre al collegamento alle DaoFactory per l'interazione con livello di persistenza, utilizza tre interfacce:

- *IAggiuntaLog*, necessaria poiché ogni controller ha la necessità di aggiungere una riga di log.
- *IPrinter*, per trasmettere i risultati al livello di front-end.
- *ISessione*, per salvare e successivamente reperire i dati dell'utente e del relativo staff (o dell'amministratore), anche in vista della scrittura del log.

Nei seguenti diagrammi verrà mostrato il livello Presenter del modello MVP. Per semplicità di notazione ho usato il nome Controller.

Package application.middleware.controller.log

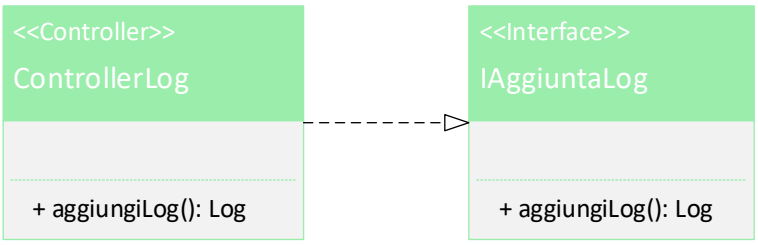


Figura 27 - progettazione/struttura/middleware/controller.vsdx

Package application.middleware.controller.autenticazione

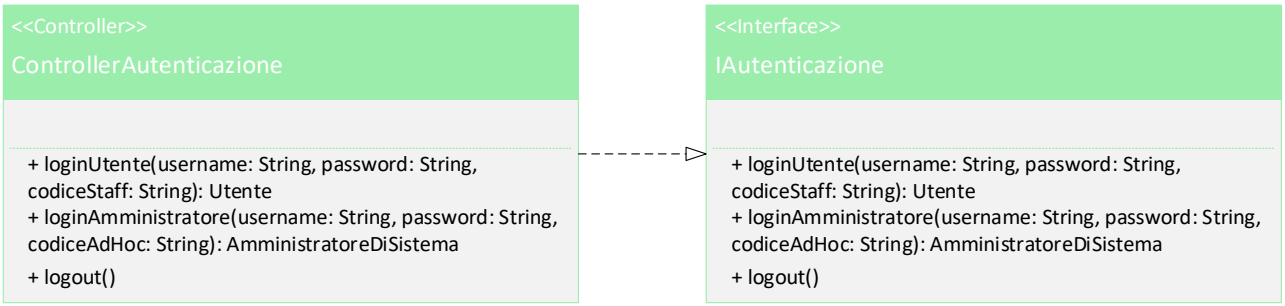


Figura 28 - progettazione/struttura/middleware/controller.vsdx

All'interno di *ContollerAutenticazione*, per garantire i requisiti R4NF e R5NF, viene riconosciuto se la password corrisponde a quella standard. In caso affermativo l'informazione va trasmessa all'interno della risposta all'applicativo front-end, che dovrà occuparsi di redirigere l'utente alla schermata di cambio password. Per riconoscere le password inserite a default dal titolare e dall' amministratore di sistema, queste devono contenere la stringa "STD", che non sarà poi possibile utilizzare per ridefinire password personali.

Oltre all'amministratore di sistema, sappiamo esserci nel sistema tre tipologie di utenti:

- Titolare
- Gestore
- Utente normale

Ho pensato di dividere i controller concreti per ruolo, i quali a loro volta implementano diverse interfacce, separate per ambito. Questa scelta è dovuta a motivi di estensibilità; supponendo infatti di dover aggiungere un nuovo tipo di utente oppure inserire qualche nuova funzionalità, basterà definire una nuova interfaccia e farla realizzare dal controller associato al ruolo che vogliamo. Inoltre, all'interno dei controller concreti relativi al titolare, al gestore e all'amministratore di sistema, viene effettuato un controllo dell'accesso per mezzo dell'interfaccia *ISessione* presente in *AbstractController*.

Package application.middleware.controller.amministratore

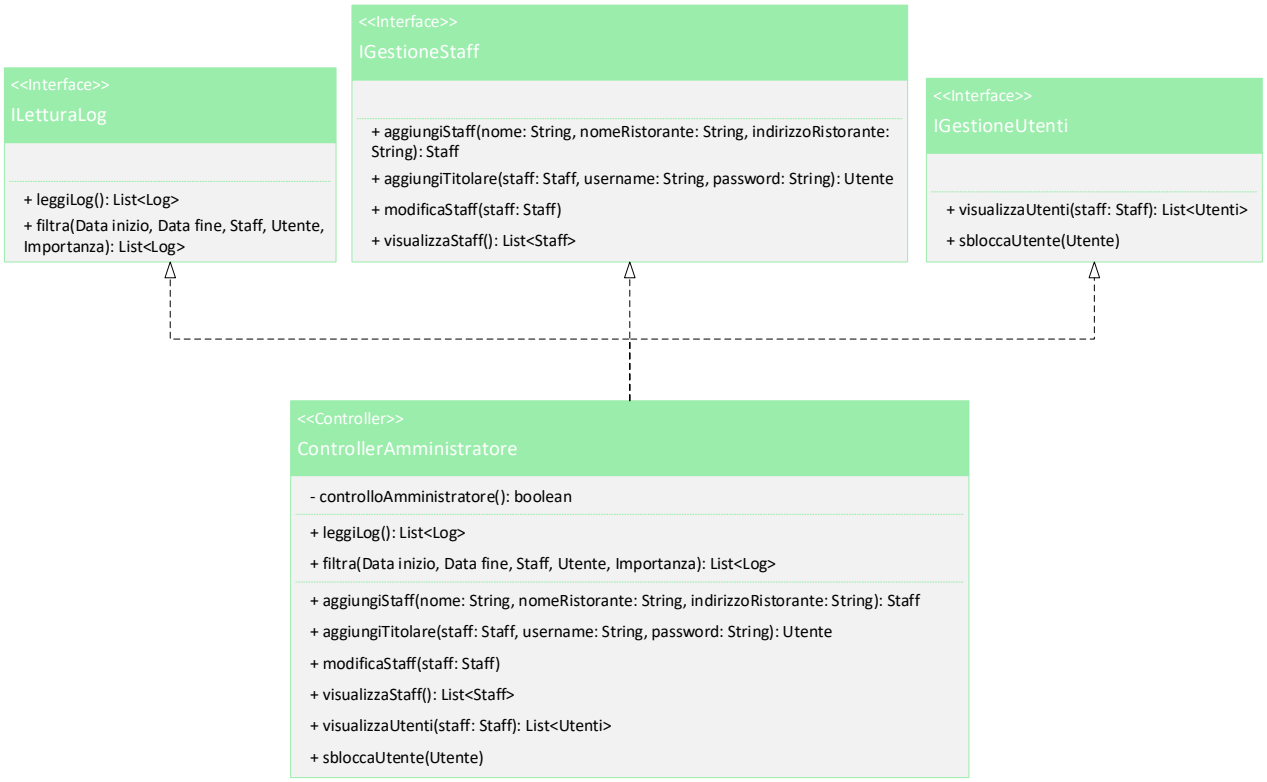


Figura 29 - progettazione/struttura/middleware/controller.vsdx

Package application.middleware.controller.titolare

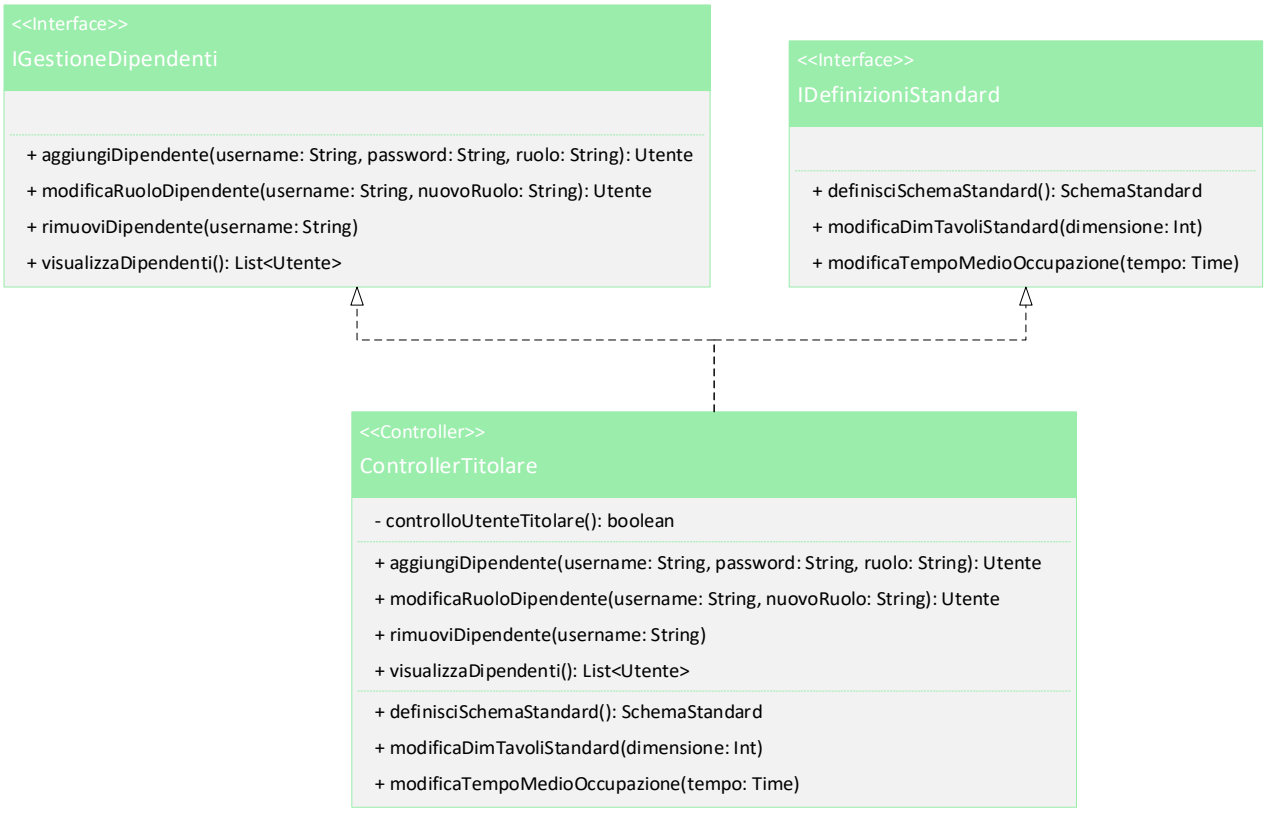


Figura 30 - progettazione/struttura/middleware/controller.vsdx

Package application.middleware.controller.gestore

Figura 31 - progettazione/struttura/middleware/controller.vsdx

Package application.middleware.controller.utente

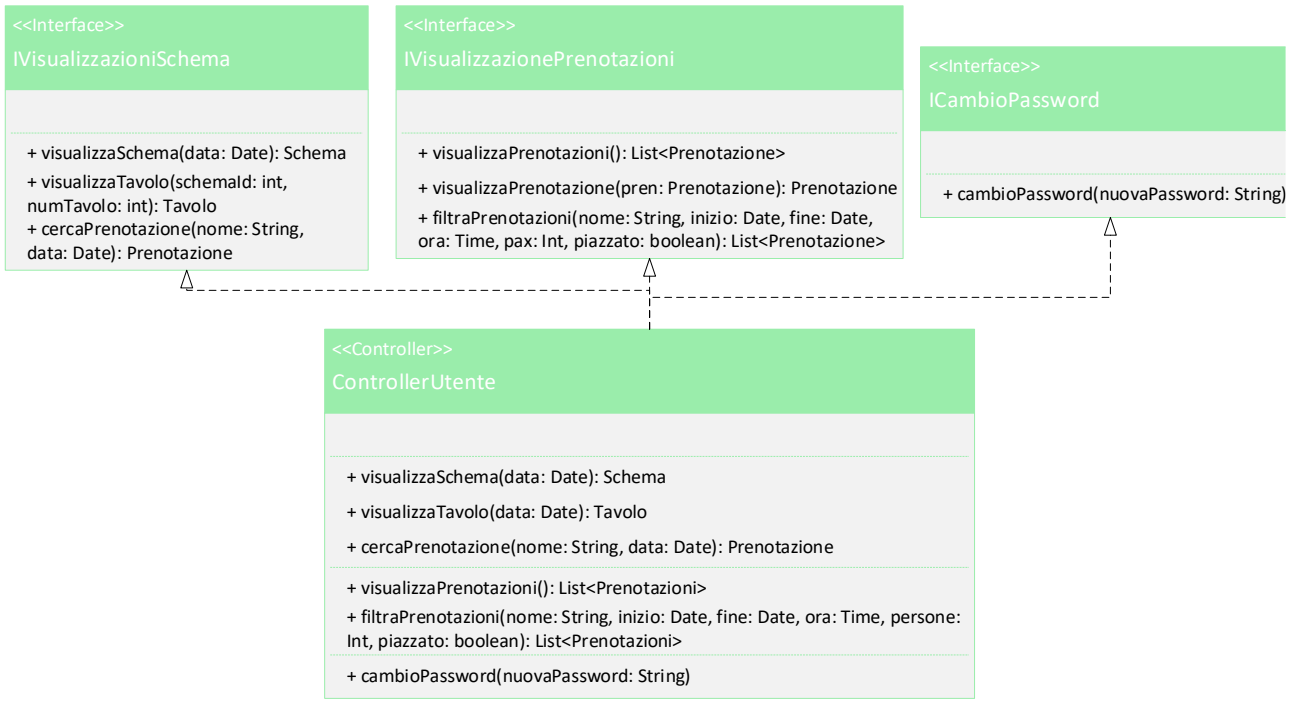


Figura 32 - progettazione/struttura/middleware/controller.vsdx

Package application.middleware.broker

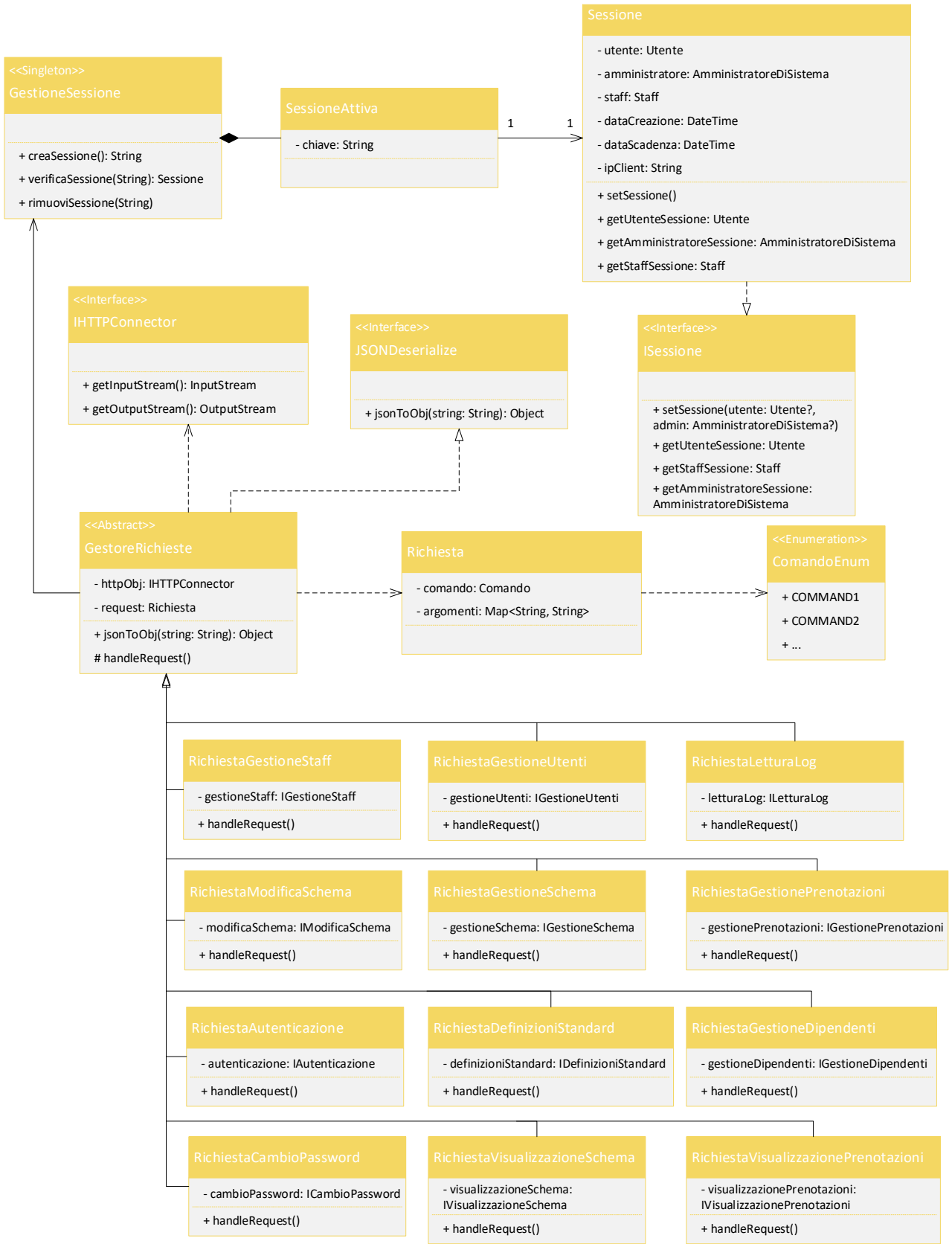


Figura 33 - progettazione/struttura/middleware/broker.vsdx

Il package broker all'interno della componente middleware si occupa di raccogliere le richieste http e di inoltrarle al controller opportuno. Tramite il **pattern State** il metodo *handleRequest()* della classe astratta **GestoreRichieste** viene ridefinito, differenziando i comandi richiesti per argomento.

All'interno del package è anche gestito il controllo della sessione; infatti, GestoreRichieste crea la sessione e verifica che sia attiva prima di smistarla. Una volta raggiunto il controller desiderato, avverrà il controllo dell'accesso sfruttando appunto la sessione.

Di seguito è riportata la lista di tutti i comandi disponibili nella classe ***ComandoEnum***.

Amministratore

- VISUALIZZA_LOG
- FILTRA_LOG
- VISUALIZZA_STAFF
- AGGIUNGI_STAFF
- MODIFICA_STAFF
- AGGIUNGI_TITOLARE
- VISUALIZZA_UTENTI

Gestore

- MODIFICA_SCHEMA
- AGGIUNGI_AL_TAVOLO
- RIMUOVI_DAL_TAVOLO
- TROVA_TAVOLO
- AGGIUNGI_PRENOTAZIONE
- MODIFICA_PRENOTAZIONE
- RIMUOVI_PRENOTAZIONE

Utente normale

- VISUALIZZA_SCHEMA
- VISUALIZZA_TAVOLO
- VISUALIZZA_PRENOTAZIONI
- VISUALIZZA_PRENOTAZIONE
- FILTRA_PRENOTAZIONI
- CAMBIO_PASSWORD

Titolare

- VISUALIZZA_DIPENDENTI
- AGGIUNGI_DIPENDENTE
- MODIFICA_RUOLO_DIPENDENTE
- RIMUOVI_DIPENDENTE
- DEFINISCI_SCHEMA_STANDARD
- MODIFICA_DIM_TAVOLI_STANDARD
- MODIFICA_TEMPO_MEDIO_OCCUPAZIONE

Altre

- LOGIN_UTENTE
- LOGIN_AMMINISTRATORE
- LOGOUT
- AGGIUNGI_LOG

Struttura front-end

La struttura di questo livello segue il model MVVM:

- **Model**: definizione del modello dei dati, utilizzato per la rappresentazione dei dati e per mappare le risposte ricevute dal lato server in oggetti.
- **View**: vista altamente specializzata che ha il solo scopo di presentazione, rappresentando i dati presenti nel corrispettivo View-Model.
- **View-Model**: gestione dei dati ricevuti dal lato middleware e binding con View tramite pattern Observer; è quindi il responsabile di quello che appare nella vista.

Package application.frontend.model

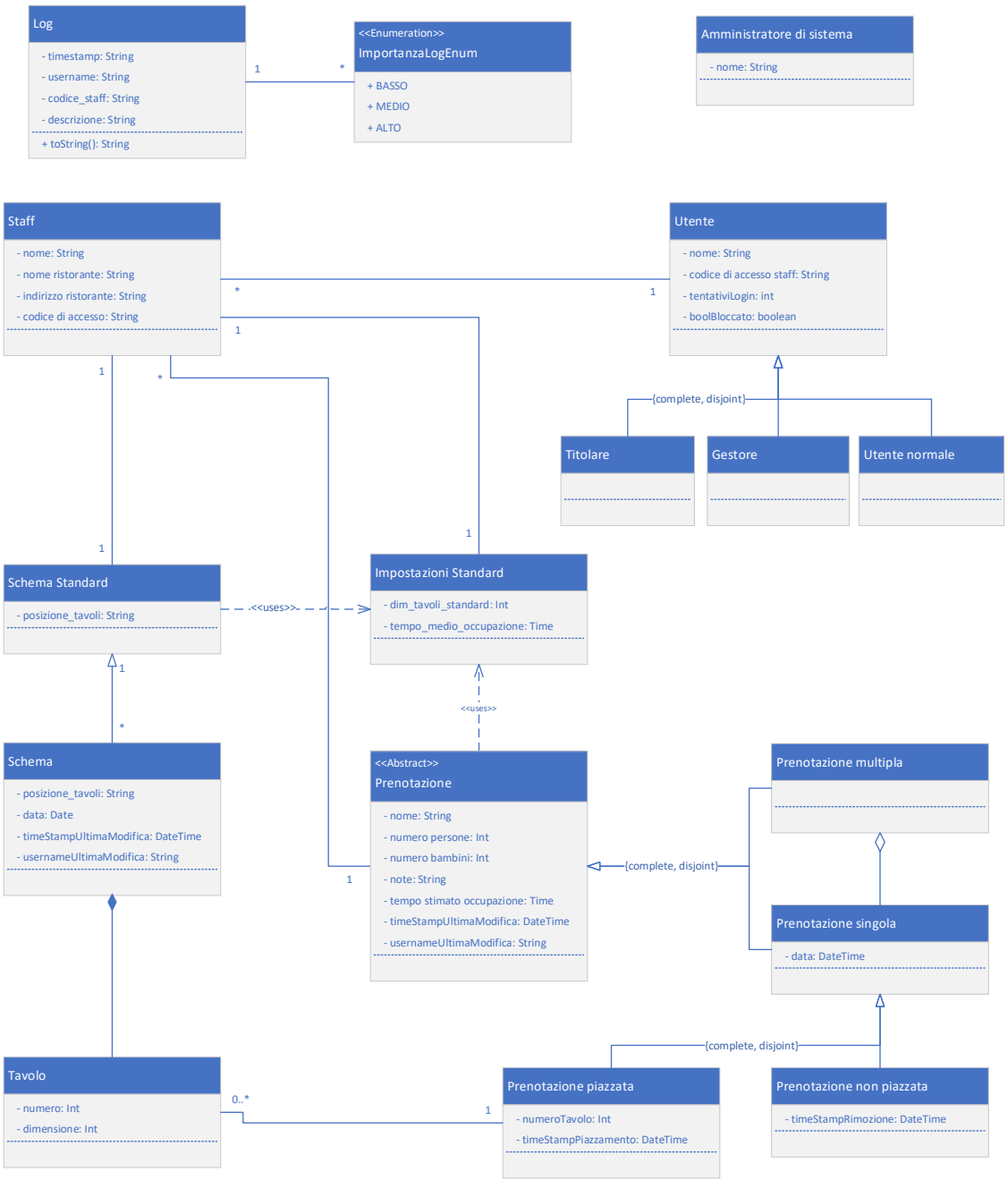


Figura 34 - progettazione/struttura/frontend/model.vsdx

Il modello del dominio è uguale a quello rappresentato nel livello middleware.

Package application.frontend.viewmodel

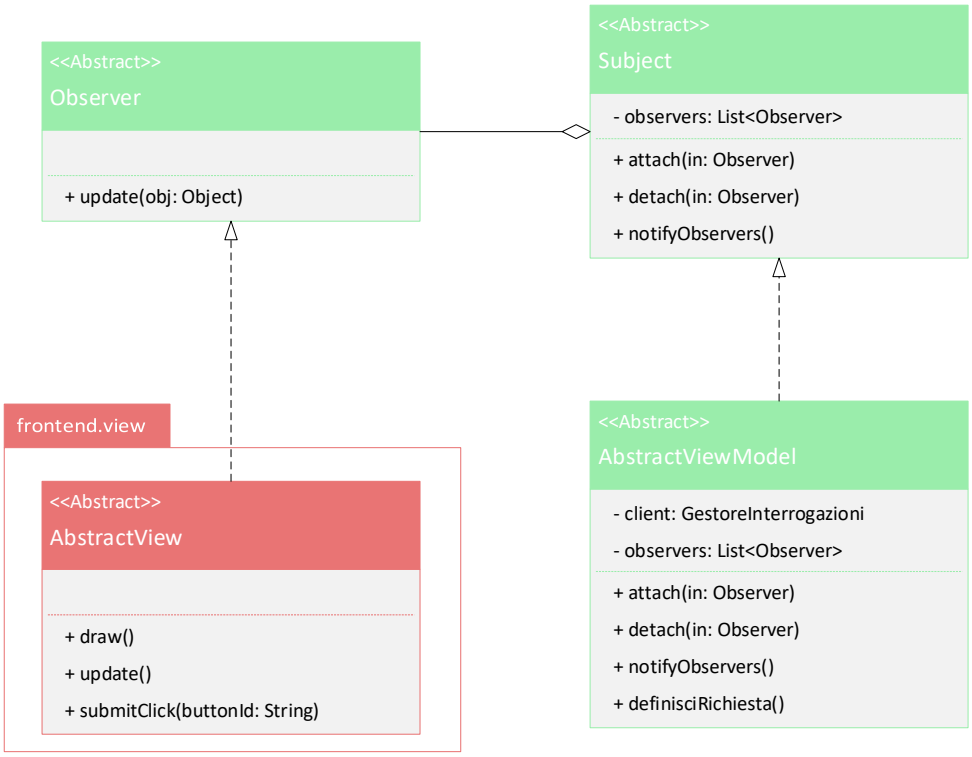


Figura 35 - progettazione/struttura/frontend/view-model.vsdx

Viene qui mostrato come avviene il binding dei dati tra View e View-Model, ovvero tramite il pattern Observer. Ogni view, che erediterà da **AbstractView**, conterrà il metodo *update(),* chiamato dal relativo view-model ogni volta che ci sarà una modifica dei dati da rappresentare tramite *notifyObservers()*.
Il metodo *submitClick()* serve per interagire con la view e fargli compiere l'azione voluta, che sia il reindirizzamento ad un'altra view oppure la chiamata ad un metodo del relativo view-model.

!! Non sono state riportate le interfacce relativa ad ogni View-Model per semplicità di presentazione

Package application.frontend.viewmodel.autenticazione

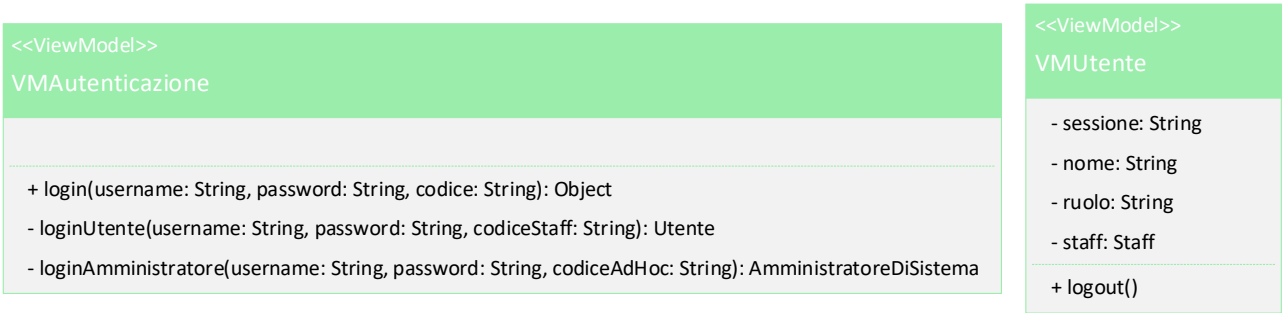


Figura 36 - progettazione/struttura/frontend/view-model.vsdx

Ho scelto di creare una nuova view, e quindi un ulteriore view-model, per la rappresentazione dei dati dell'utente, con la possibilità di effettuare il logout.

Package application.frontend.viewmodel.amministratore

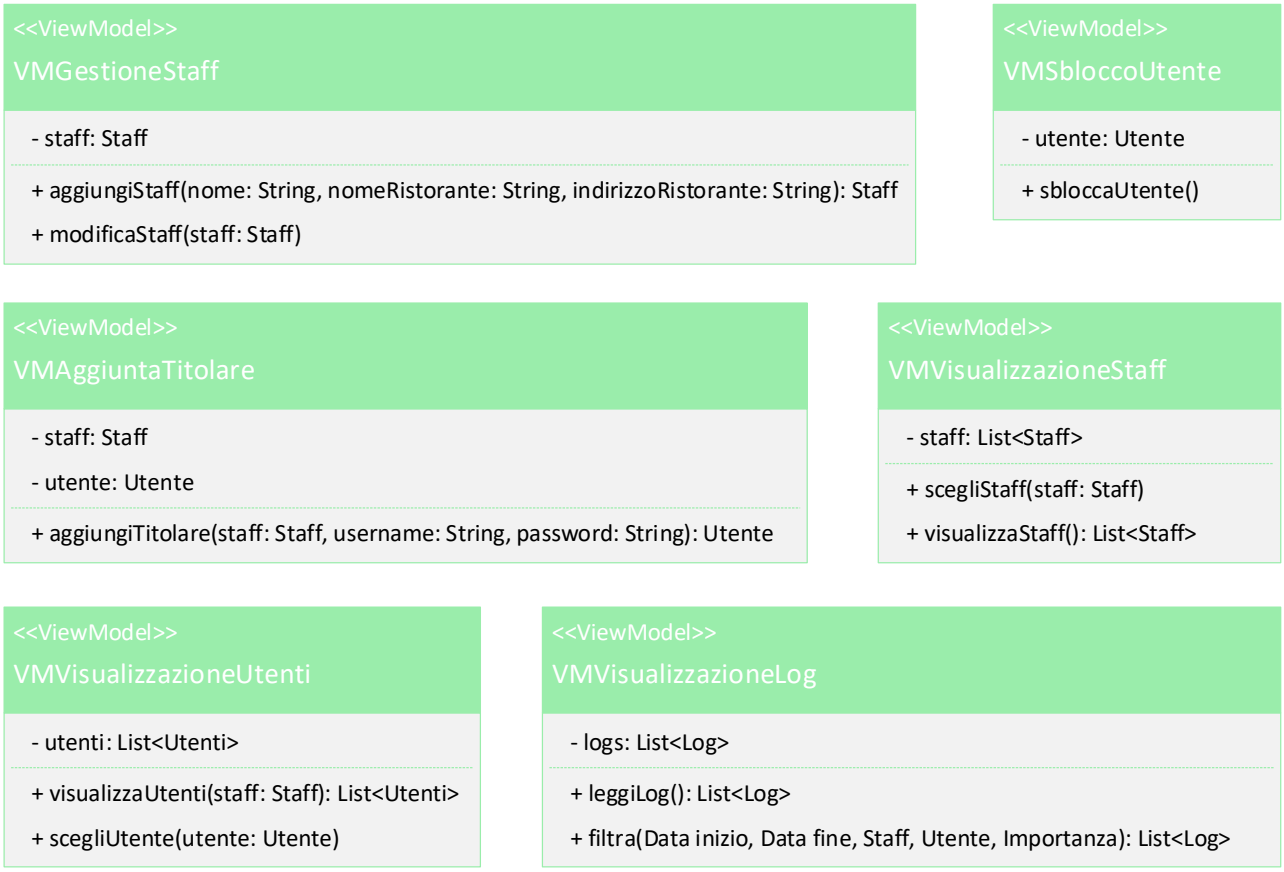


Figura 37 - progettazione/struttura/frontend/view-model.vsdx

Package application.frontend.viewmodel.titolare

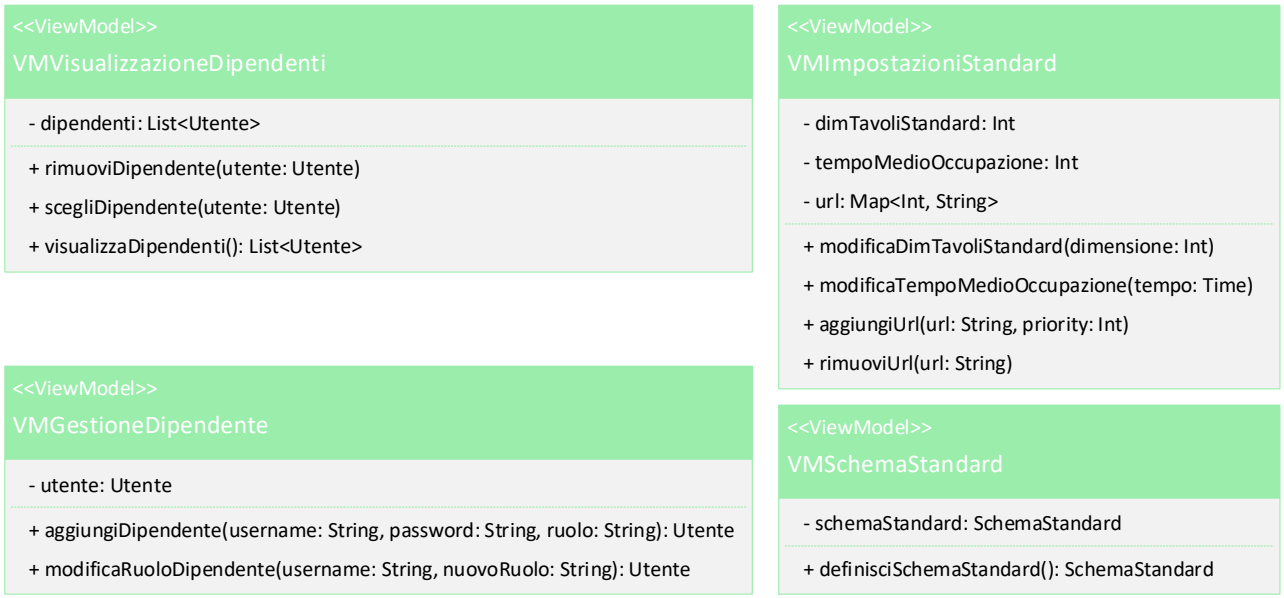


Figura 38 - progettazione/struttura/frontend/view-model.vsdx

Package application.frontend.viewmodel.gestore

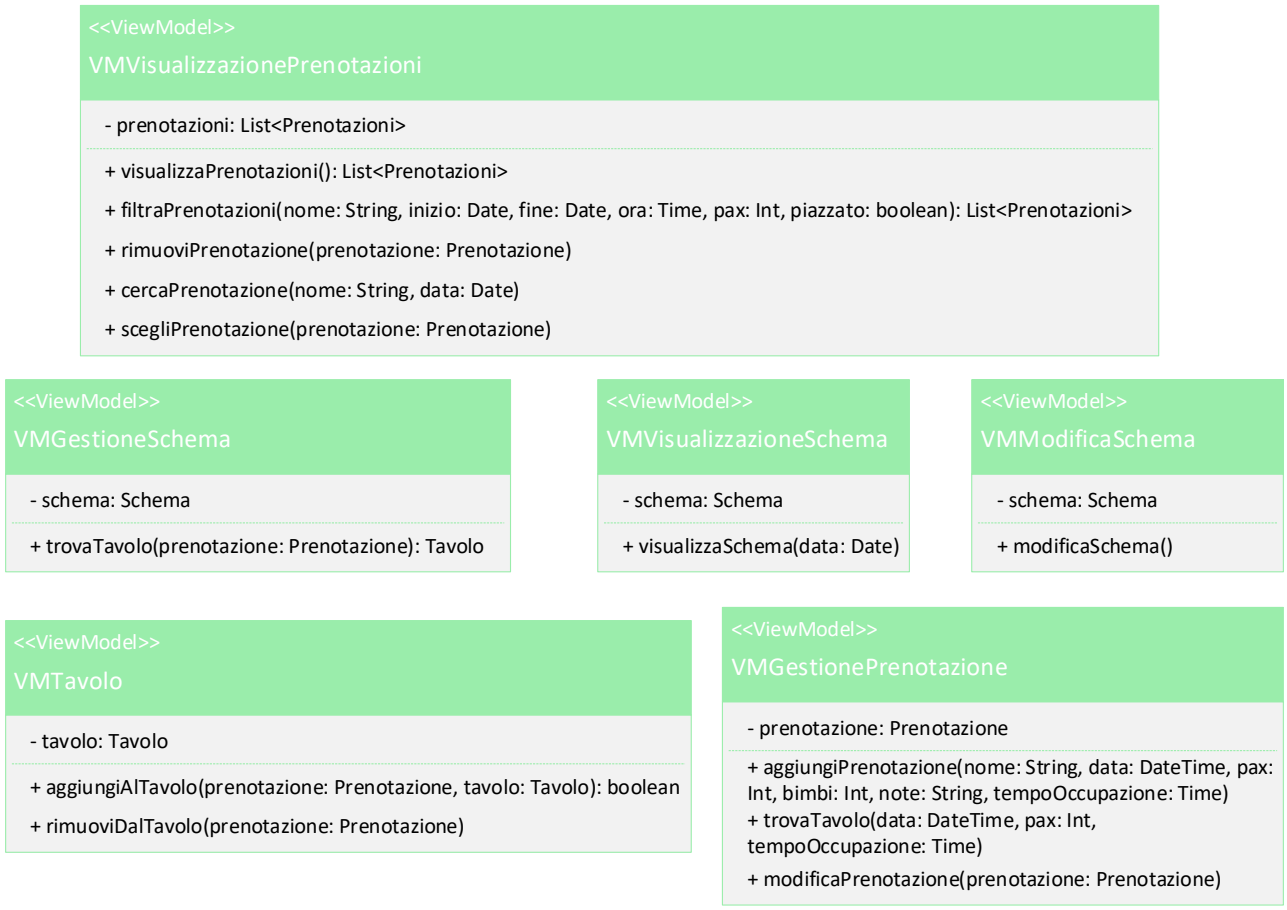


Figura 39 - progettazione/struttura/frontend/view-model.vsdx

Package application.frontend.viewmodel.utente

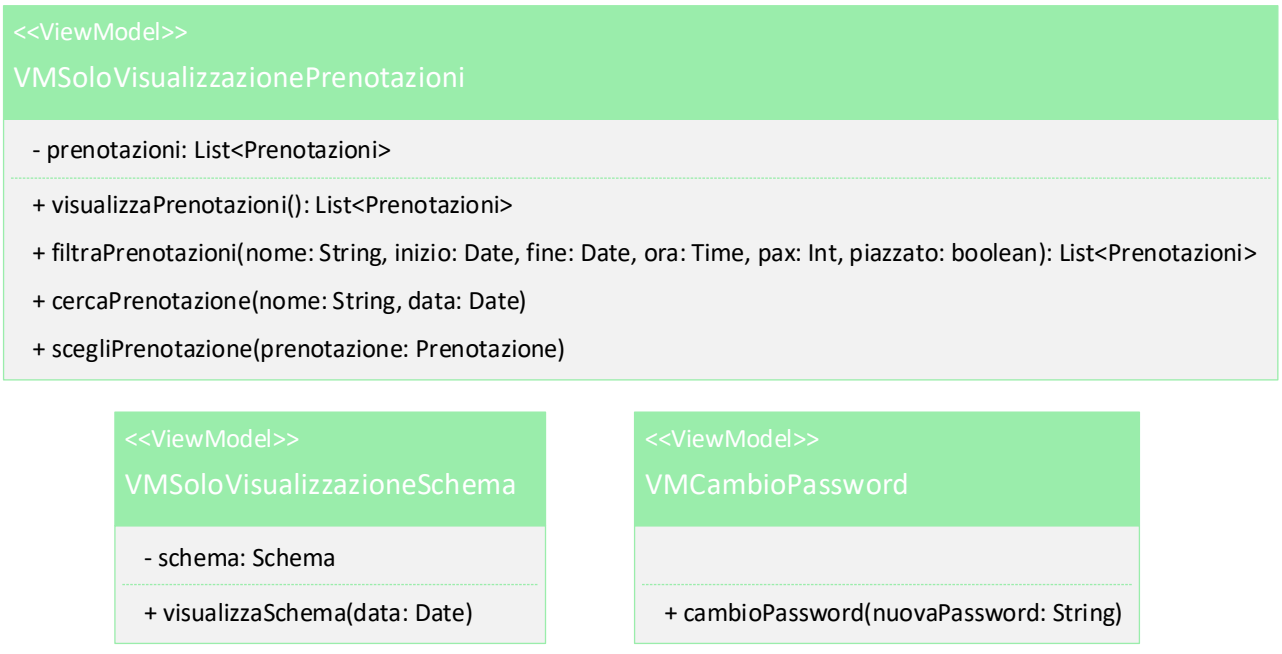


Figura 40 - progettazione/struttura/frontend/view-model.vsdx

Package application.frontend.view

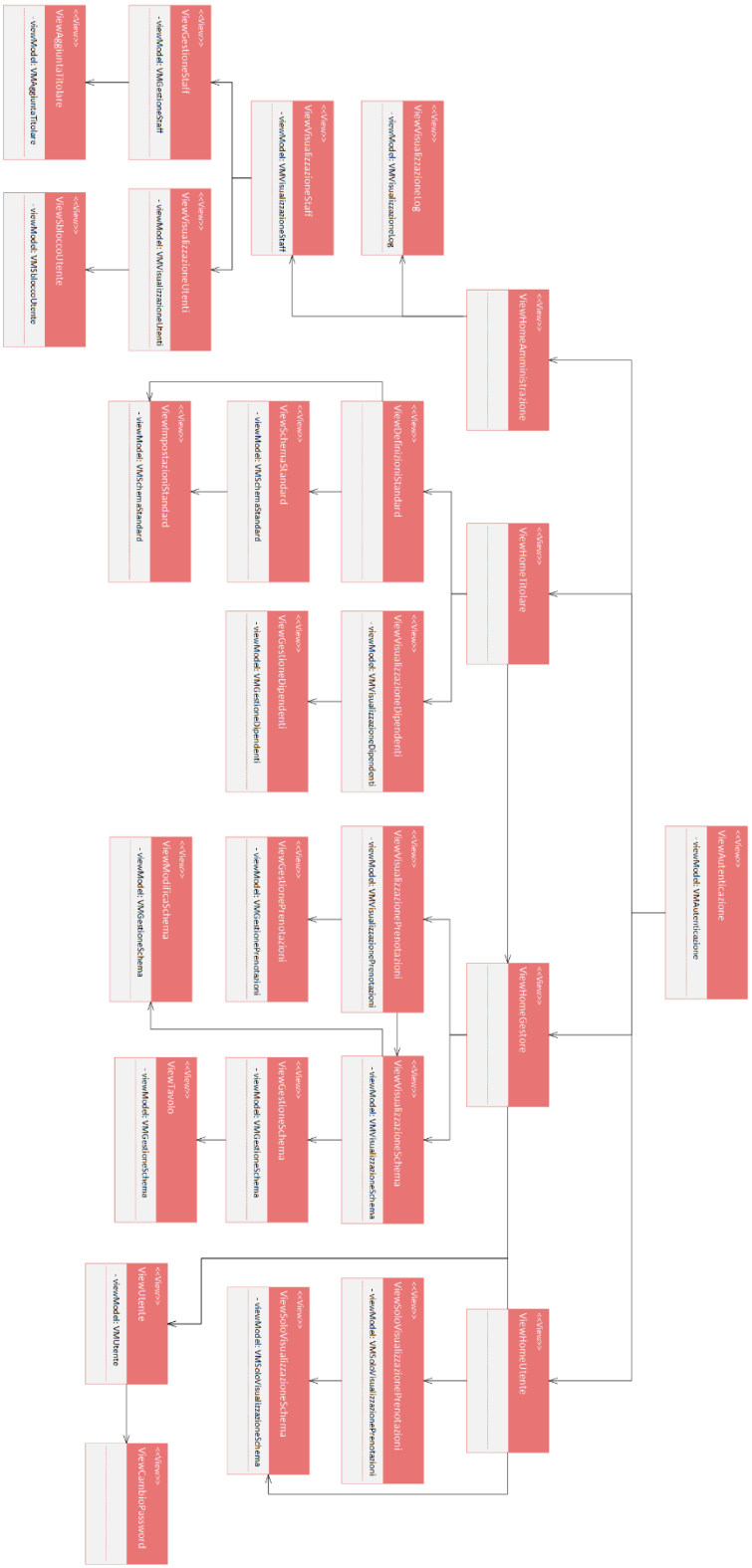


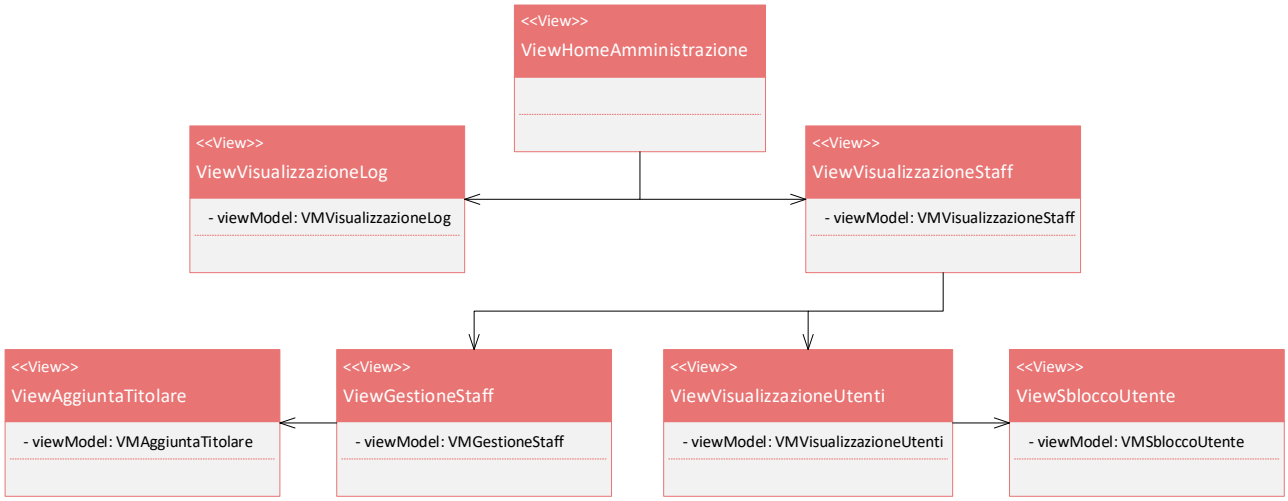
Figura 41 - progettazione/struttura/frontend/view.vsdx

L'immagine a fianco mostra tutte le view presenti all'interno del software frontend. Ognuna estende **AbstractView** e non sono stati riportati i relativi metodi per semplicità di rappresentazione. Le view che non hanno un View-Model associato hanno il solo scopo di reindirizzare ad altre view.

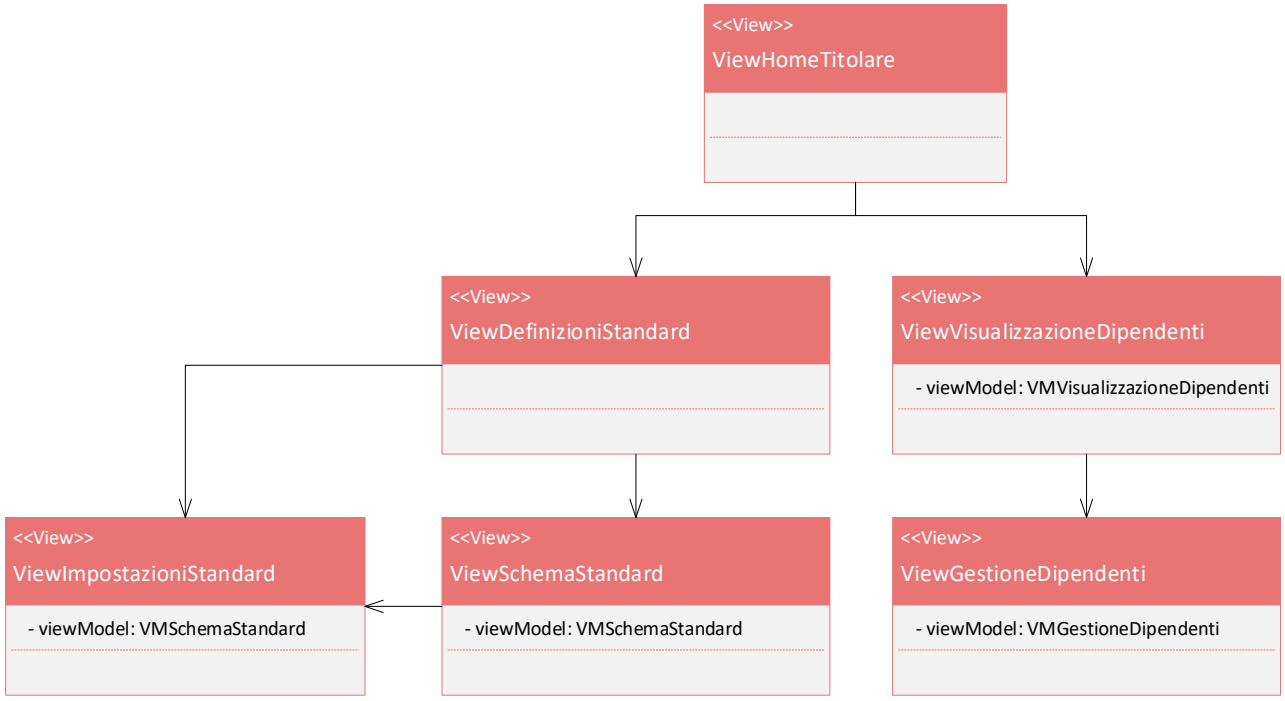
Vediamo che dalla view per il login avviene lo smistamento alla Home view relativa alla tipologia di utente.

Le frecce tra una classe e l'altra mostrano un possibile workflow, ma non sono le uniche possibili interazioni. Infatti, ogni freccia è intesa come bidirezionale, permettendo all'utente di andare a ritroso.

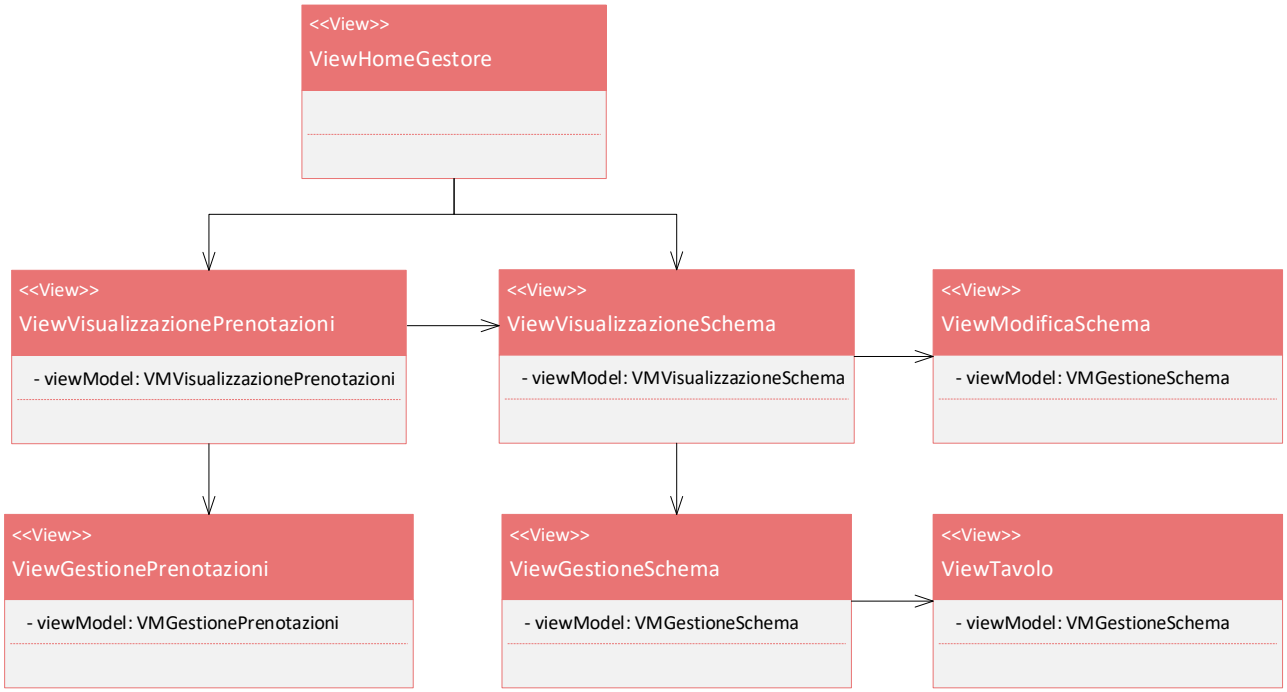
Package application.frontend.view.amministratore



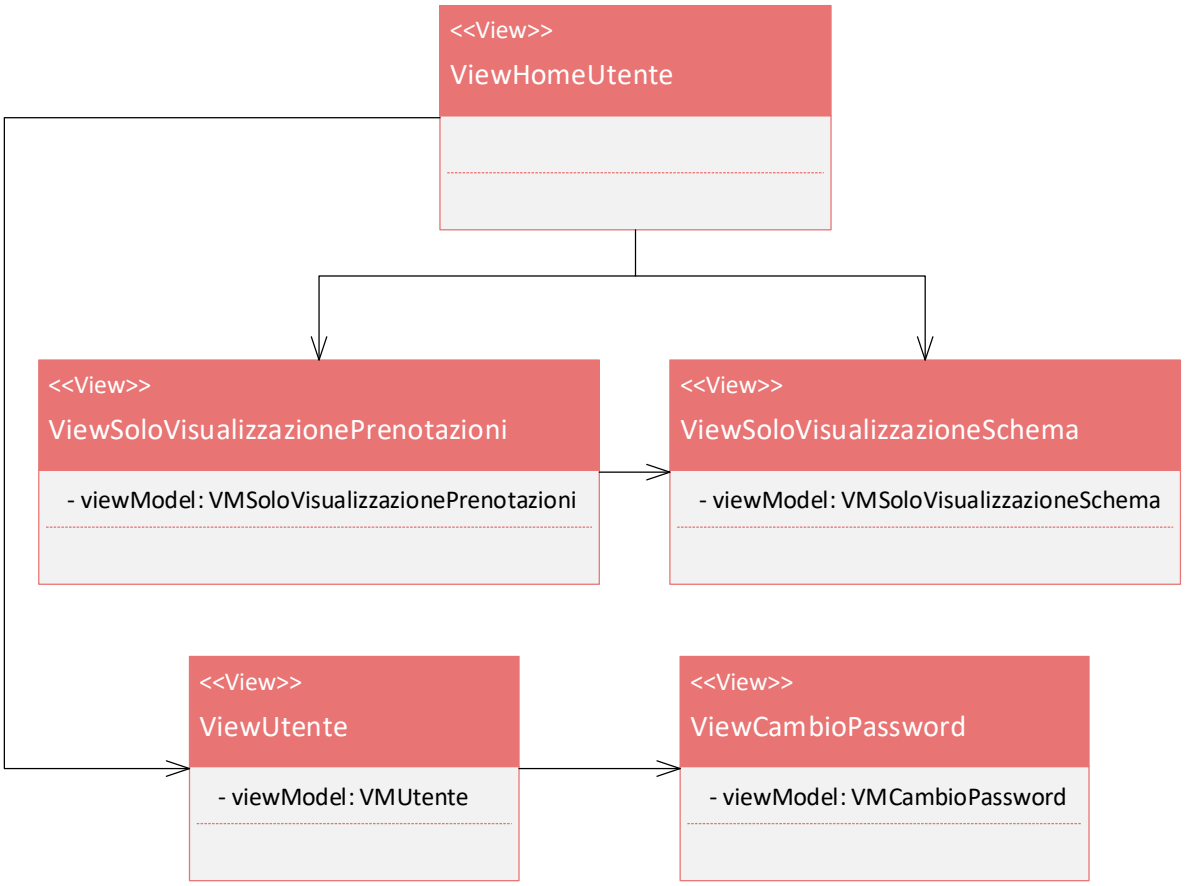
Package application.frontend.view.titolare



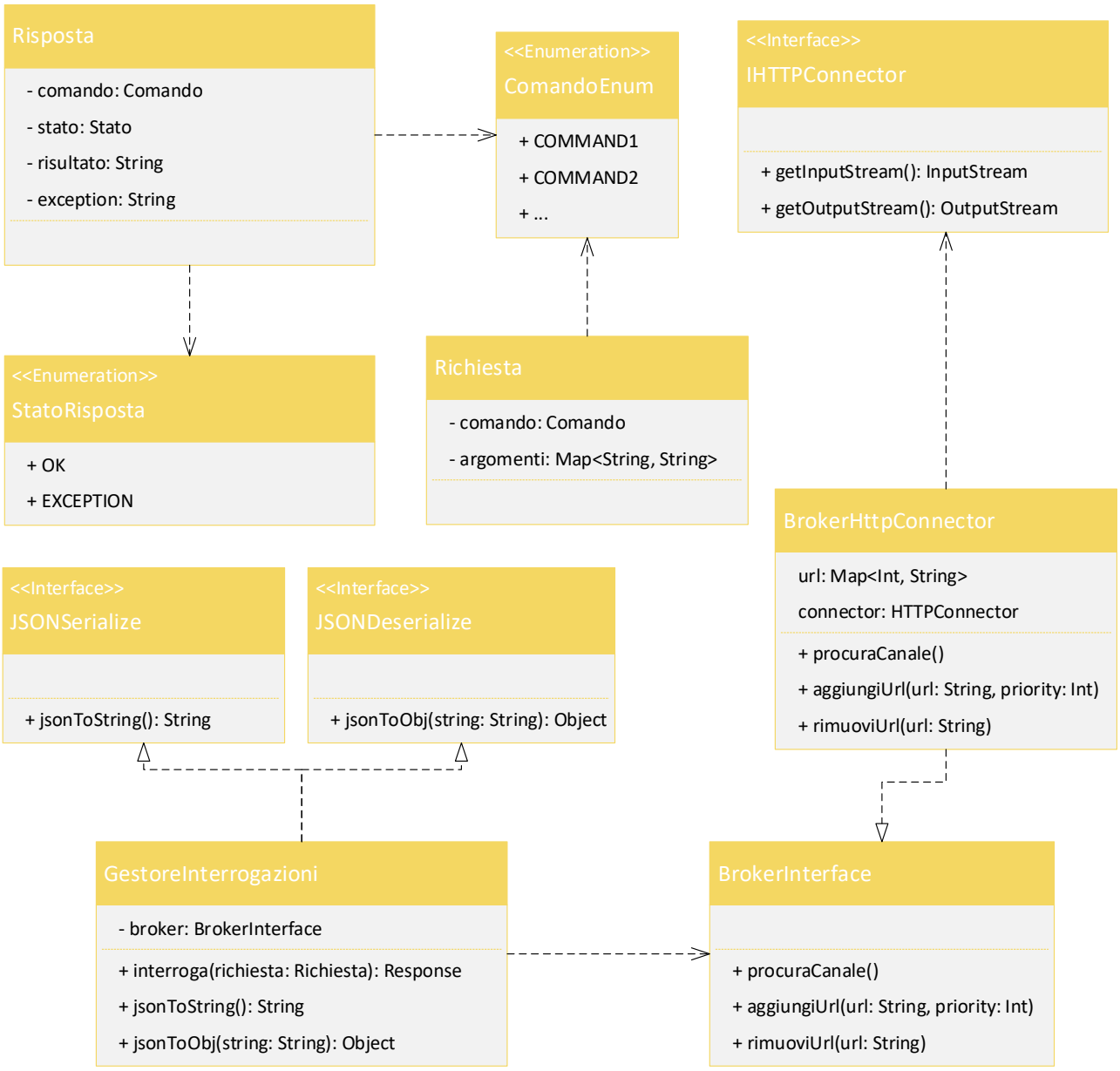
Package application.frontend.view.gestore



Package application.frontend.view.utente



Package application.frontend.broker



Molte classi ed interfacce sono in comune con il livello middleware.

Il fulcro di questo package è sicuramente **GestoreInterrogazioni** che si occupa di inoltrare le richieste al broker del livello middleware utilizzando l'interfaccia **BrokerInterface** per reperire il canale di comunicazione. Essendo che ho scelto HTTPS come protocollo di comunicazione, la classe concreta che implementa BrokerInterface è **BrokerHttpConnector**. Quest'ultima possiede una serie di URL con priorità numerica che possono essere aggiunti o rimossi dall'utente su indicazione dell'amministratore di sistema.

Interazione

Diagramma di sequenza – Autenticazione utente

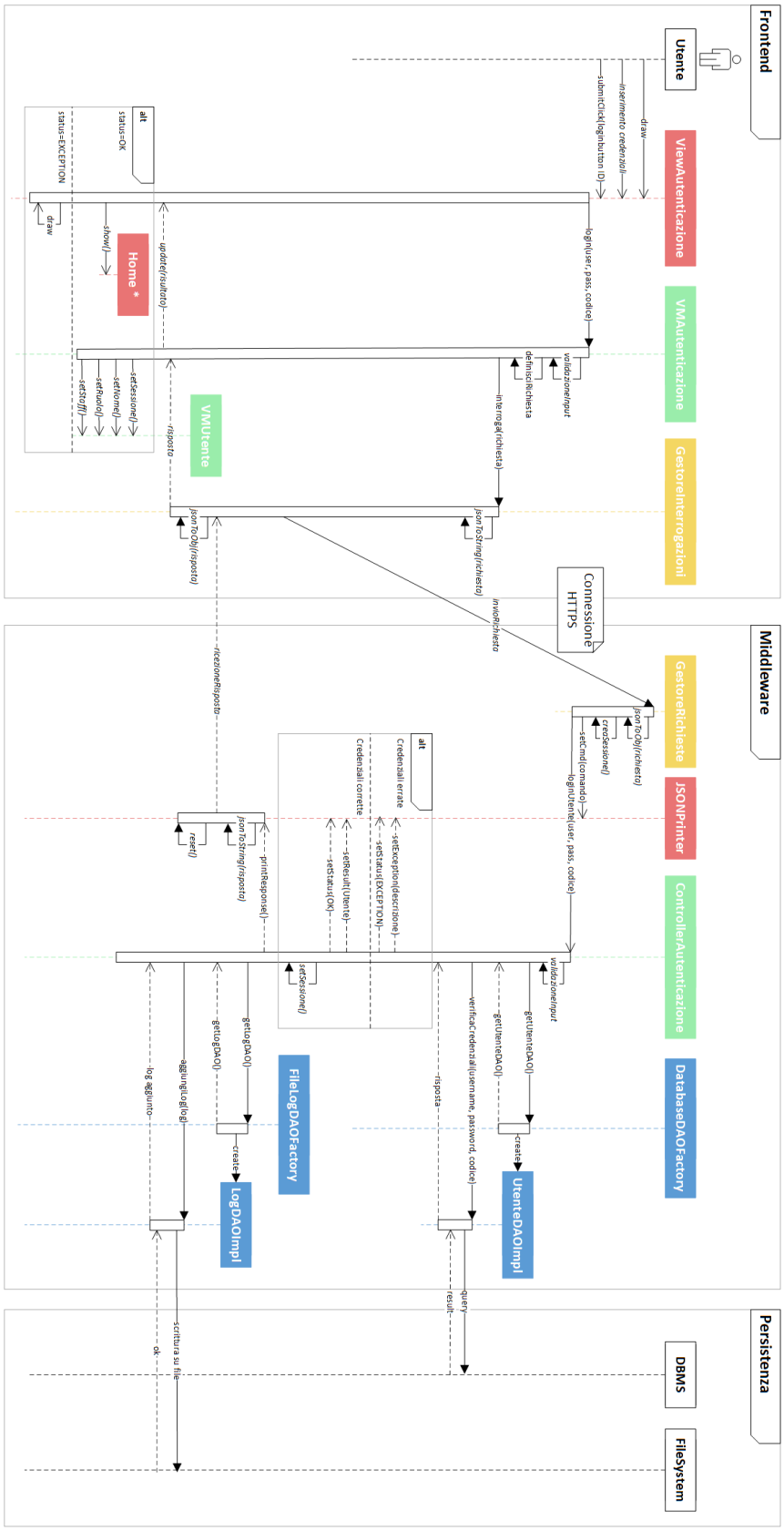


Figura 42 - progettazione/interazione/autenticazione_utente.vsdx

***Home** ** simboleggia la Home View specifica di ciascun tipo di utente, in base al ruolo.

Diagramma di sequenza – Assegnamento di un tavolo ad una prenotazione

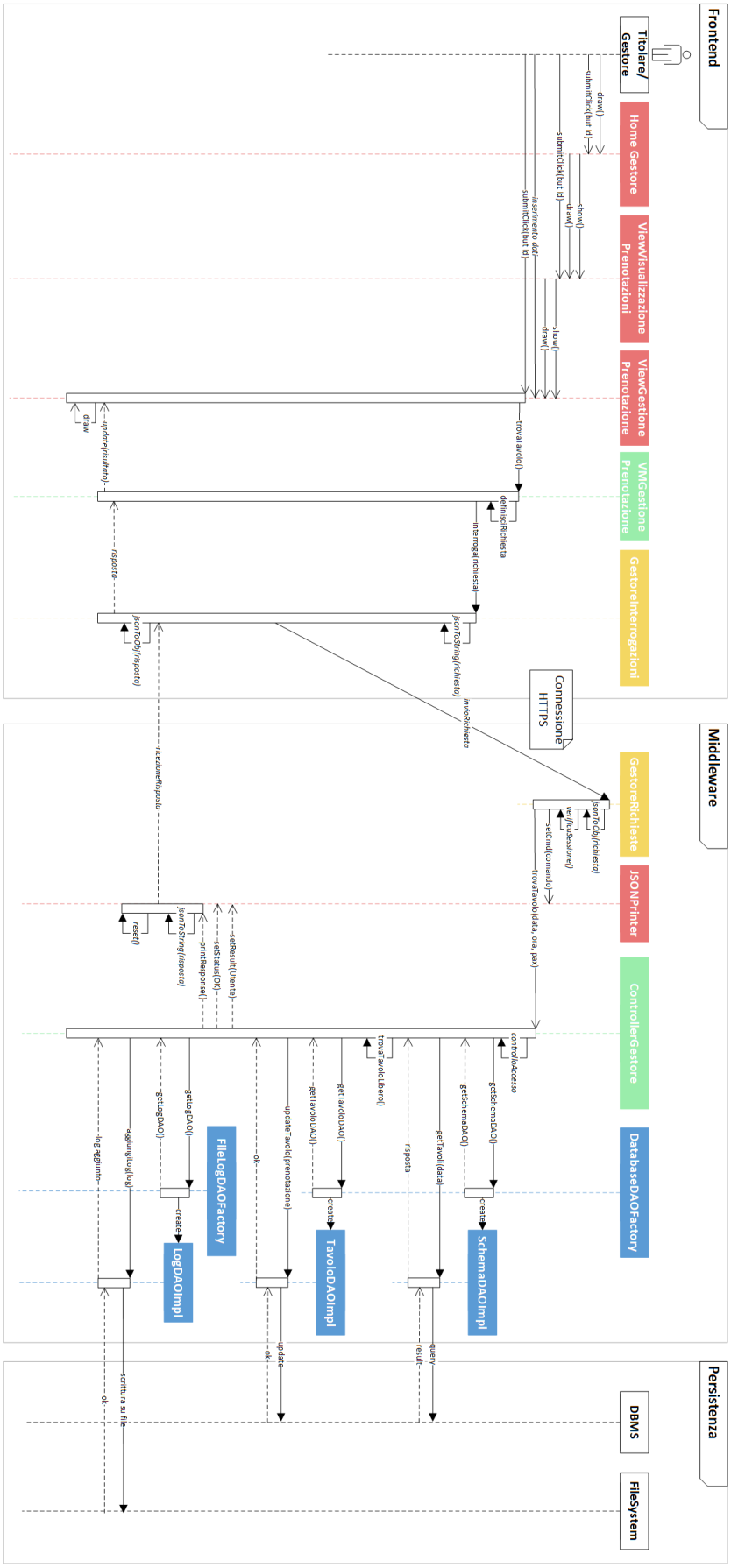


Figura 43 - progettazione/interazione/assegnamento_tavolo.vsdx

Comportamento

Ho scelto di riportare solo due diagrammi delle attività relativi all'assegnamento ed al cambio della password standard degli utenti. Infatti, l'amministratore definisce il titolare che a sua volta aggiunge dei dipendenti. Sia il titolare che i dipendenti al primo accesso dovranno cambiare la password.

Diagramma di attività – Gestione password standard

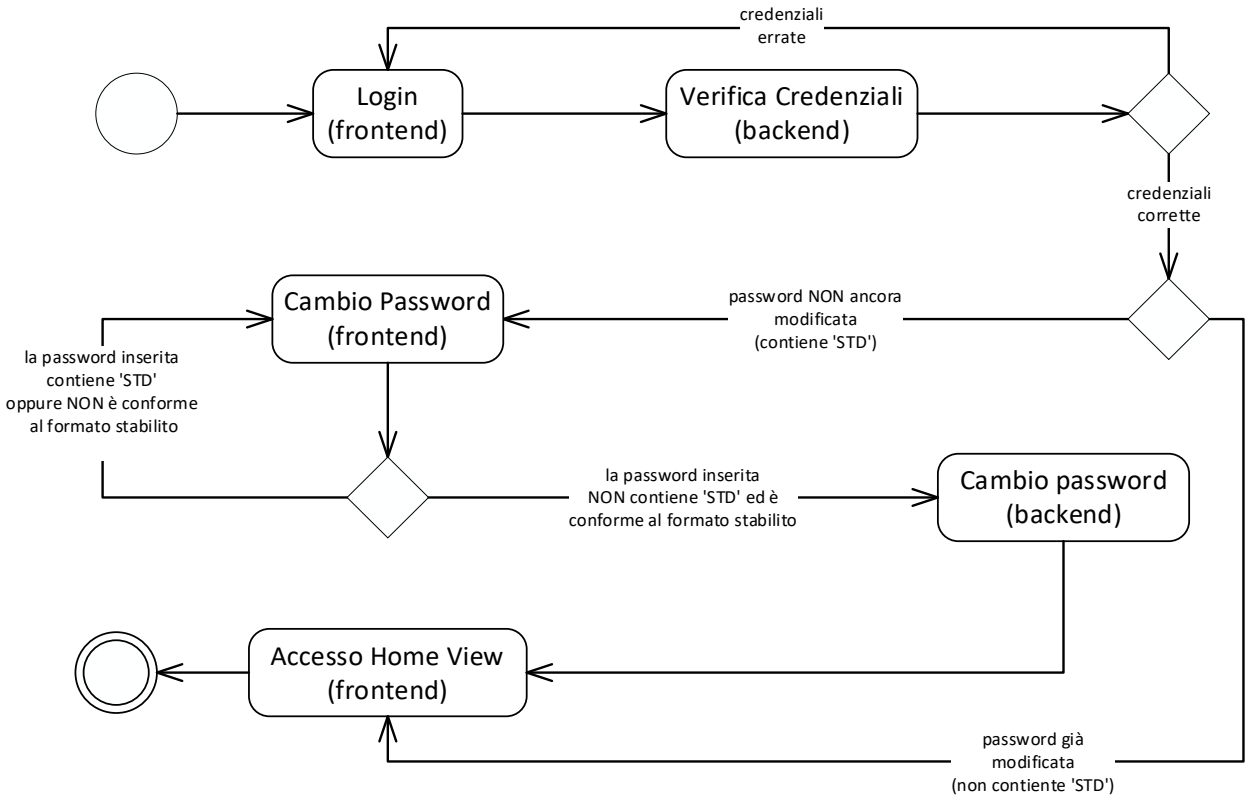


Figura 44 - progettazione/comportamento/gestione_password_standard.vsdx

Di seguito, un ulteriore diagramma che riguarda l'aggiunta del titolare o di un dipendente da parte rispettivamente dell'amministratore di sistema e del titolare.

Diagramma di attività – Aggiunta di un utente

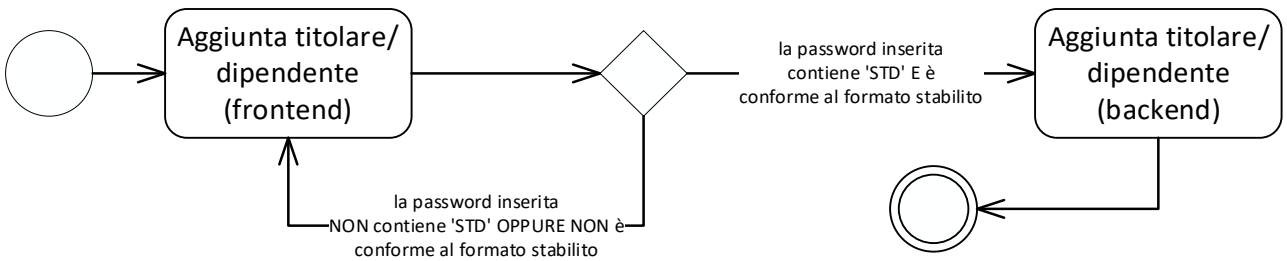


Figura 45 - progettazione/comportamento/aggiunta_utente.vsdx

Progettazione della persistenza

Schema E-R

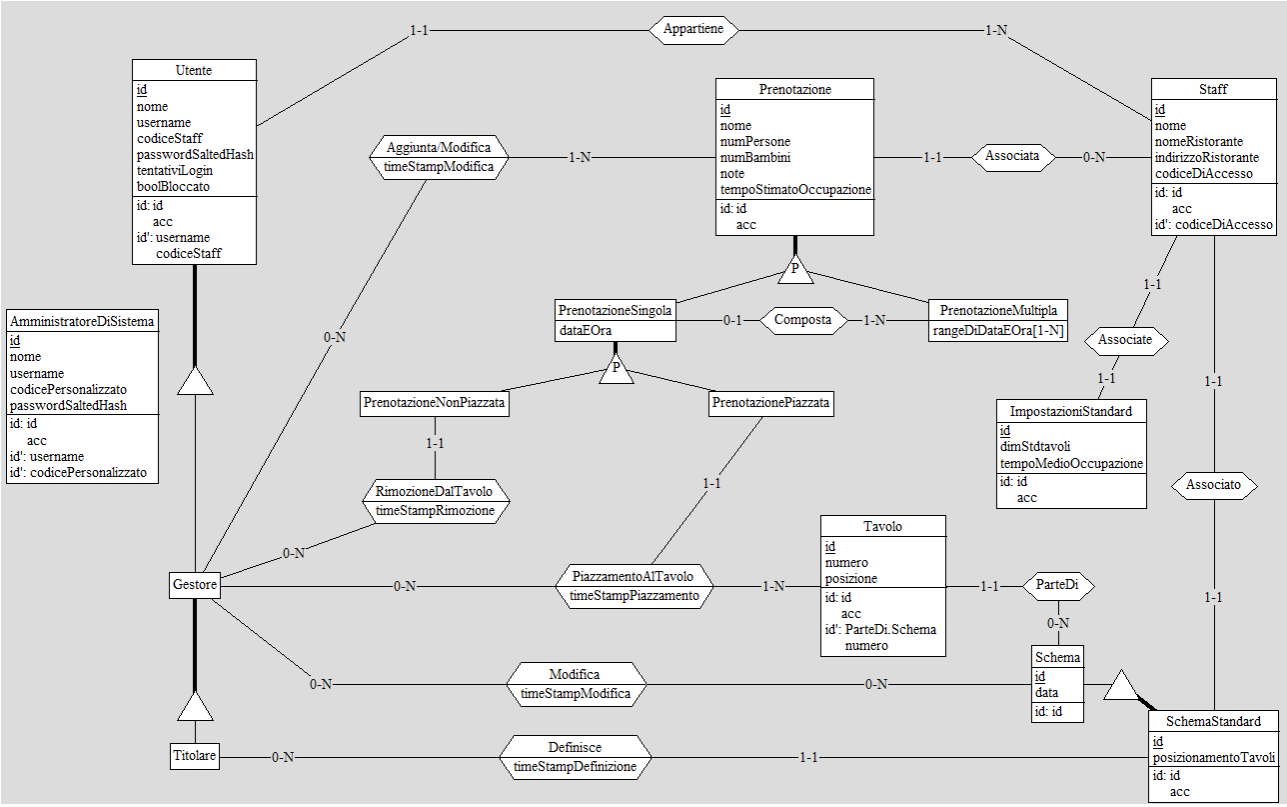


Figura 46 - progettazione/persistenza.lun

Tutte le entità possiedono degli **id** interi auto incrementali. Oltre ad essi che vengono utilizzati tramite access key, sono presenti anche **vincoli UNIQUE**:

- Coppia username dell'utente e codice del relativo staff
- Codice di accesso dello staff
- Numero del tavolo all'interno dello schema di un determinato giorno
- Data dello schema (non possono esserci due schemi dei tavoli per la stessa data)
- Coppia data-nome relativa ad una prenotazione

Deve essere anche realizzato un **Trigger** per quanto riguarda data e ora della prenotazione, che devono essere successive al momento nel quale la si sta prendendo.

Formato Log

Essendo salvati all'interno di un file, l'entità Log non compare all'interno dello schema del database. Il formato del file è il seguente:

<timestamp> <codice_staff - username> <descrizione> <livello>

Progettazione del collaudo

Di seguito sono riportati alcuni dei test unitari presenti nel sistema, realizzati in NUnit.

```
5   namespace GestionalePrenotazioni.Tests.Model
6   {
7       public class TestUtente
8       {
9           private Staff staff1, staff2;
10          private Utente utente1,utente2,utente3,utente4;
11          private StaffDaoImpl controllerStaff = new StaffDaoImpl();
12          private UtenteDaoImpl controllerUtente = new UtenteDaoImpl();
13
14          [SetUp]
15          public void setup()
16          {
17              // Creazione degli staff
18              staff1 = new Staff("staff1","ST11_", "risto1", "indirizzoRisto1");
19              staff2 = new Staff("staff2", "ST22_", "risto2", "indirizzoRisto2");
20
21              staff1.Id = controllerStaff.create(staff1);
22              staff2.Id = controllerStaff.create(staff2);
23
24              // Creazione degli utenti
25              utente1 = new Utente("utente1", "username1", "password1", staff1.codiceDiAccesso, staff1.Id);
26              utente2 = new Titolare("utente2", "username2", "password2", staff1.codiceDiAccesso, staff1.Id);
27              utente3 = new Gestore("utente3", "username3", "password3", staff2.codiceDiAccesso, staff2.Id);
28              utente4 = new Titolare("utente4", "username4", "password4", staff2.codiceDiAccesso, staff2.Id);
29
30              utente1.Id = controllerUtente.create(utente1);
31              utente2.Id = controllerUtente.create(utente2);
32              utente3.Id = controllerUtente.create(utente3);
33              utente4.Id = controllerUtente.create(utente4);
34          }
36          [Test]
37          public void gettersStaff()
38          {
39              Staff staff = (Staff)controllerStaff.retrieve(staff1.Id);
40
41              // Associazione Staff - Utenti
42              staff.utenti = controllerUtente.leggiUtenti(staff1.Id);
43
44              Assert.AreEqual("staff1", staff.nome);
45              Assert.AreEqual("risto1", staff.nomeRistorante);
46              Assert.AreEqual("indirizzoRisto1", staff.indirizzoRistorante);
47              Assert.AreEqual("ST11_", staff.codiceDiAccesso);
48              Assert.AreEqual(staff.utenti[0].Id, utente1.Id);
49              Assert.AreEqual(staff.utenti[1].Id, utente2.Id);
50          }
51
52          [Test]
53          public void gettersUtente()
54          {
55              Utente utente = (Utente)controllerUtente.retrieve(utente2.Id);
56
57              Assert.AreEqual("utente2", utente.nome);
58              Assert.AreEqual("username2", utente.username);
59              Assert.AreEqual("password2", utente.password);
60              Assert.AreEqual(false, utente.boolBloccato);
61              Assert.AreEqual(0, utente.numeroTentativiLogin);
62              Assert.AreEqual(staff1.codiceDiAccesso, utente.codiceDiAccessoStaff);
63              Assert.AreEqual(staff1.Id, utente.staffId);
64              Assert.AreEqual(utente.GetType(), typeof(Titolare));
65          }
```

Figura 47 - progetttazione/collaudo/testUtente.cs

Progettazione del deployment

Deployment per la sicurezza

Il servizio middleware deve essere installato su un nodo sicuro all'interno di una rete privata. Non vi può essere una comunicazione diretta con i controller, ma l'unico punto di accesso dall'esterno deve essere fornito dal package broker. Lo stesso vale per l'uscita di informazioni.

Configurazioni di deployment

Lato server, la configurazione è gestita interamente dall'amministratore di sistema, il quale dovrà delineare un file di configurazione per l'accesso al database, specificandone:

- URL
- tipo
- nome
- credenziali per l'accesso

Lato client, verrà solo richiesto l'inserimento dell'URL del servizio middleware

Artefatti

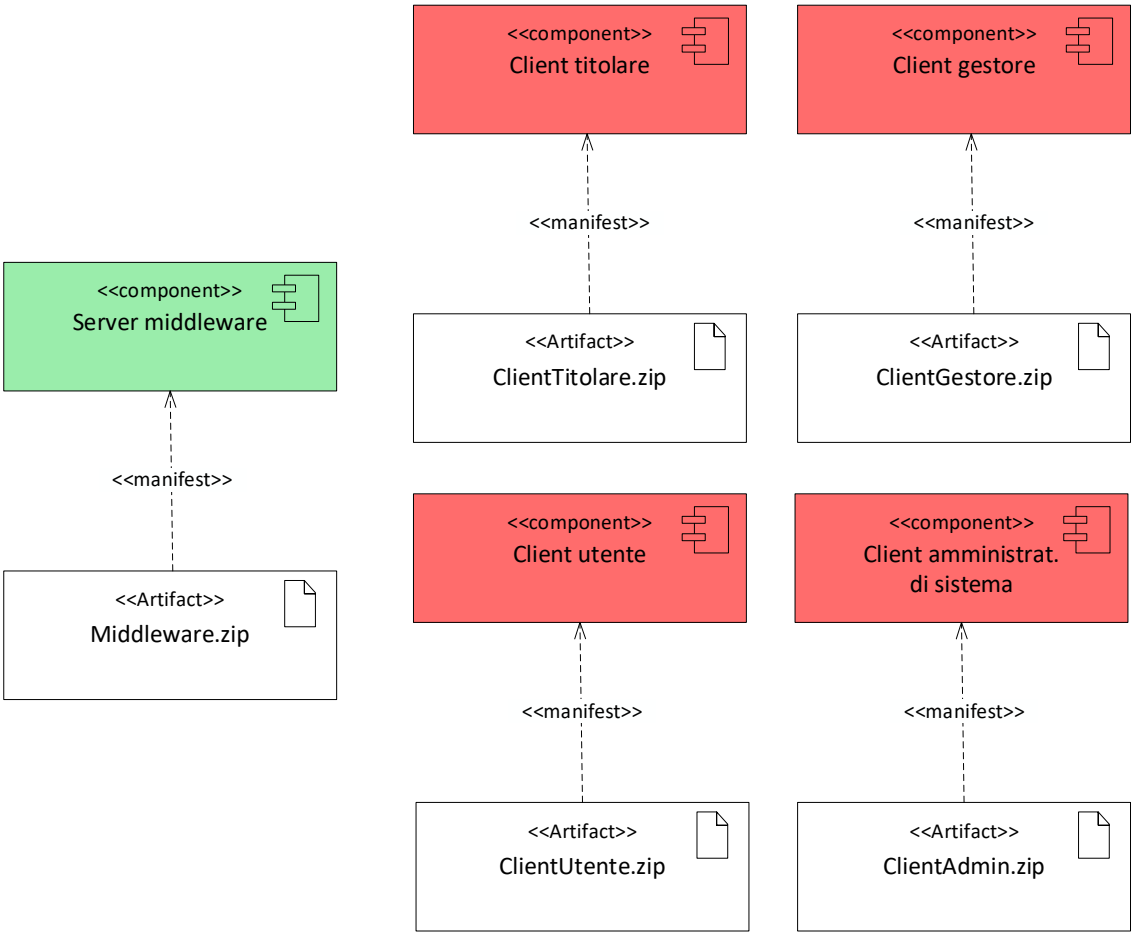


Figura 48 - deployment/artifact.vsdx

Come mostrato nel diagramma dei package della fase di progettazione, ognuna delle componenti client possiede le funzionalità disponibili non solo per l'utente specifico (e.g. il titolare), ma anche per i suoi sottoposti (in questo caso il gestore e l'utente normale). *ClientAdmin* fa eccezione, in quanto verrà installato solo ed esclusivamente sul dispositivo dell'amministratore di sistema e solo da lui usato.

Type-Level

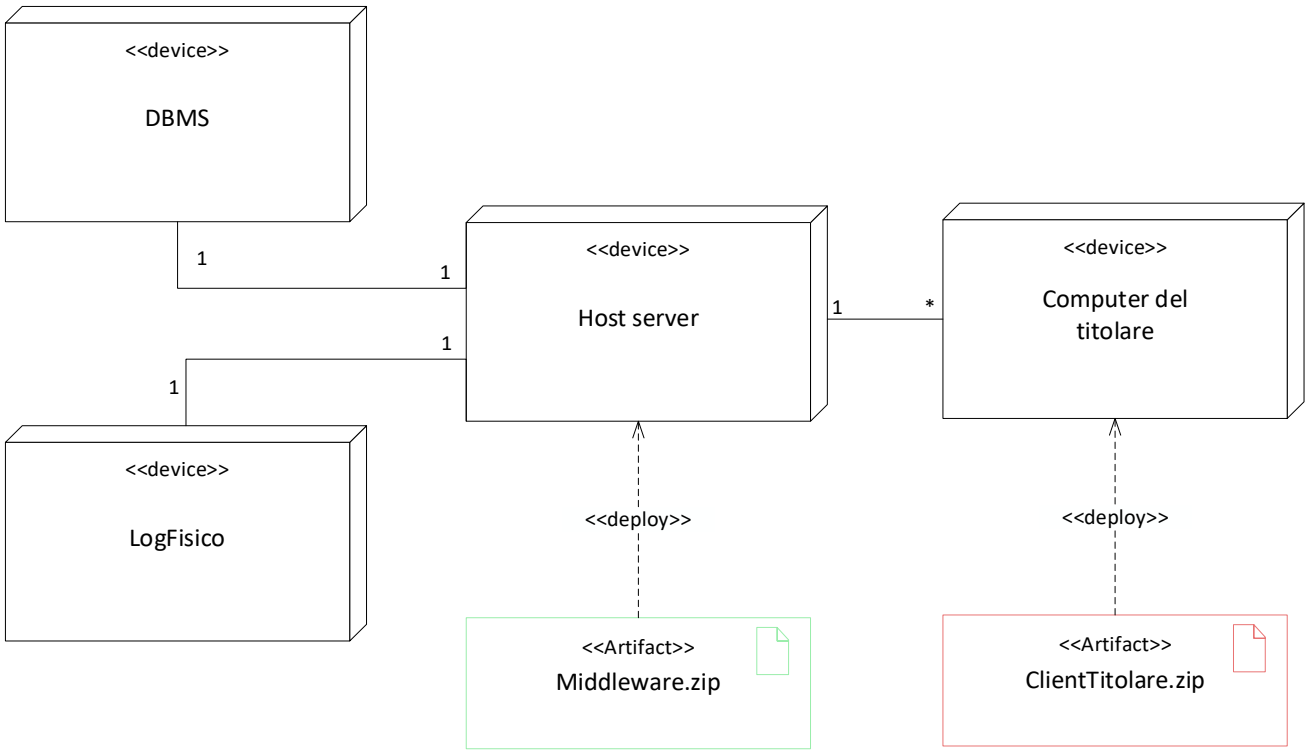


Figura 49 - deployment/type-level.vsdx

Nel diagramma sopra è mostrata l'installazione del solo *ClientTitolare*, ma lo schema è analogo per qualsiasi altro tipo di client.

Implementazione

Scelte tecnologiche

Per quanto riguarda lo strato middleware ho scelto di realizzarlo in C# all'interno di .NET Framework, mentre per la persistenza ho utilizzato Microsoft SQLServer. Grazie a Visual Studio 2022 e ad Entity Framework ho potuto creare le tabelle del database rapidamente e in modo automatico.
All'interno della soluzione, oltre al progetto principale, è presente un secondo progetto di test basato su NUnit.

Lo strato front-end è stato invece implementato in Angular13, usando quindi una soluzione web-based. All'interno di ogni componente le view sono rappresentate dai file HTML e CSS, mentre i view-model in TypeScript.

Per la pubblicazione delle API da parte dello strato middleware ho optato per Swagger, di uso comune in .NET Framework.

Scelte implementative

- Per ogni implementazione DAO ho definito delle stringhe parametriche per la definizione delle query.
- Vengono usati id auto incrementali all'interno del database, perciò dopo ogni inserimento di una tupla viene restituito l'id da associare all'oggetto all'interno del modello.
- Le password vengono salvate nel database secondo algoritmo di hashing SHA256.
- La parte View dell'applicativo front-end interagisce con View-Model tramite direttive NgModel.