



UNIVERSITÀ FEDERICO II DI NAPOLI

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

CORSO DI SOFTWARE ARCHITECTURE DESIGN LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA

ANNO ACCADEMICO 2023-2024

DOCUMENTAZIONE SOFTWARE ARCHITECTURE DESIGN

FavolApp

Insieme per un futuro migliore, un passo alla volta.

Prof.ssa

Ing. Anna Rita Fasolino

Autori:

Francesco Scognamiglio - M63001364

Felice Micillo - M63001377

Indice

1 Introduzione	5
1.1 Contesto dell'applicazione	5
1.2 Organizzazioni Non Lucrativo di Utilità Sociale (ONLUS)	5
1.2.1 Cooperative di tipo A.....	5
1.2.2 Cooperative di tipo B.....	5
2 Avvio della progettazione.....	6
2.1 Attori e obiettivi generali	6
2.2 Funzionalità generali del sistema software.....	6
2.3 Assunzioni e dipendenze.....	7
2.4 Storie utente – Epic (Requisiti funzionali).....	7
2.4.1 Elementi chiave	7
2.5 Requisiti funzionali delle interfacce esterne.....	8
2.5.1 Interfaccia utente.....	8
2.5.2 Interfaccia software.....	15
2.5.3 Interfaccia hardware.....	15
2.5.4 Interfaccia di comunicazione.....	15
2.6 Requisiti di consegna	15
2.7 Requisiti di riservatezza.....	15
2.8 Requisiti non funzionali	16
2.8.1 Sicurezza	16
2.8.2 Scalabilità	18
2.8.3 Usabilità	18
2.8.4 Affidabilità.....	19
2.8.5 Manutenibilità	20
2.9 Stima dei costi.....	20
2.9.1 Unadjusted Use Case Weight (UUCW).....	21
2.9.2 Unadjusted Actor Weight (UAW).....	22
2.9.3 Technical Complexity Factor (TCF)	23
2.9.4 Environmental Complexity Factor (ECF)	24
2.9.5 Total Use Case Points (UCP).....	25
2.10 System Context Diagram	26
3 Analisi dei requisiti.....	27
3.1 Modello dei Casi d'uso	27
3.1.1 Casi d'uso - Autenticazione	27
3.1.2 Casi d'uso – Pagina del profilo personale.....	29
3.1.3 Casi d'uso – Pagina del profilo di un altro utente	32
3.1.4 Casi d'uso – Pagina per la gestione degli utenti	37
3.1.5 Casi d'uso – Pagina dei colleghi.....	41
3.1.6 Casi d'uso – Pagina dei report.....	44
3.1.7 Casi d'uso – Pagina dei pazienti.....	50
3.1.8 Casi d'uso – Pagina delle attività.....	55
3.2 System Sequence Diagram.....	59
3.2.1 System Sequence Diagram – Autenticazione	59
3.2.2 System Sequence Diagram – Pagina del profilo personale.....	60

3.2.3	System Sequence Diagram – Pagina del profilo di un altro utente	62
3.2.4	System Sequence Diagram – Pagina per la gestione degli utenti	64
3.2.5	System Sequence Diagram – Pagina dei colleghi	67
3.2.6	System Sequence Diagram – Pagina dei report	68
3.2.7	System Sequence Diagram – Pagina dei pazienti	72
3.2.8	System Sequence Diagram – Pagina dei report	75
3.2.9	System Domain Model	77
3.3	Web App Map	78
3.4	Mockup	78
4	Architettura e progettazione del software	83
4.1	Decisioni architettoniche e progettuali	83
4.1.1	Web Application	83
4.2	Pattern architettonico – Model View Controller (MVC)	84
4.3	Stile architettonico – Microservizi	85
4.4	GraphQL	85
4.4.1	Funzionamento	86
4.4.2	Vantaggi	87
4.5	REST	87
4.6	Vista componenti e connettori	88
4.7	Sequence diagram di progettazione	89
4.7.1	Sequence diagram di progettazione – Autenticazione	90
4.7.2	Sequence diagram di progettazione – Utenti	91
4.7.3	Sequence diagram di progettazione – Pazienti	97
4.7.4	Sequence diagram di progettazione – Report	100
4.7.5	Sequence diagram di progettazione – Attività	104
4.7.6	Sequence diagram di progettazione – Storage	105
4.8	Activity Diagram	107
4.8.1	Activity Diagram – Autenticazione	107
4.8.2	Activity Diagram – Utenti	109
4.8.3	Activity Diagram – Pazienti	114
4.8.4	Activity Diagram – Report	117
4.8.5	Activity Diagram – Attività	120
4.8.6	Activity Diagram – Storage	122
4.9	Communication Diagram	123
4.9.1	Communication Diagram – Autenticazione	123
4.9.2	Communication Diagram – Utenti	124
4.9.3	Communication Diagram – Pazienti	127
4.9.4	Communication Diagram – Report	129
4.9.5	Communication Diagram – Attività	131
4.9.6	Communication Diagram – Storage	132
4.10	Class Diagram MVC	133
5	Processo di sviluppo	135
5.1	Framework di sviluppo	135
5.1.1	SCRUM Framework	135
5.1.2	Suddivisione dei ruoli	135
5.1.3	Evento SCRUM	136
5.2	Descrizione degli Sprint	136
5.2.1	Sprint 1	136
5.2.2	Sprint 2	137

5.2.3	Sprint 3.....	138
5.3	Software utilizzati durante il processo di sviluppo.....	139
5.3.1	GitLab.....	140
5.3.2	Microsoft Teams	141
5.3.3	Visual Studio Code.....	141
5.3.4	Word.....	142
5.3.5	Visual Paradigm.....	142
6	Tecnologie utilizzate.....	144
6.1	Introduzione alle tecnologie utilizzate.....	144
6.2	Frontend - ReactJS	144
6.2.1	Componenti	144
6.2.2	Virtual DOM.....	144
6.2.3	Single Page Application (SPA).....	145
6.3	Backend – Amazon Web Services.....	145
6.3.1	Configurazione dei servizi e delle risorse	145
6.4	Servizi utilizzati AWS	146
6.4.1	AWS Cognito	146
6.4.2	AWS AppSync.....	146
6.4.3	AWS Lambda	147
6.4.4	AWS DynamoDB.....	148
6.4.5	AWS CloudWatch	149
6.4.6	AWS S3 (Simple Storage Service)	150
6.4.7	CloudFront.....	151
6.5	Architettura AWS	151
6.5.1	Ramo di autenticazione.....	152
6.5.2	Ramo di orchestrazione delle richieste	154
6.5.3	Ramo per lo storage delle immagini.....	156
6.5.4	Ramo per la distribuzione dell'applicazione	157
6.5.5	AWS Deployment Diagram.....	159
6.6	Comunicazione tra ReactJS e AWS	160
7	Specifiche di implementazione	161
7.1	Package Diagram	161
7.1.1	Package Diagram – FavolApp	161
7.2	Design Pattern.....	162
7.2.1	Pattern Grasp.....	163
7.2.2	Pattern Proxy	164
7.2.3	Pattern Observer	164
7.3	Sequence diagram di dettaglio.....	164
7.3.1	Sequence diagram di dettaglio - Autenticazione.....	165
7.3.2	Sequence diagram di dettaglio - Utenti	166
7.3.3	Sequence diagram di dettaglio - Pazienti.....	170
7.3.4	Sequence diagram di dettaglio - Report.....	172
7.3.5	Sequence diagram di dettaglio - Attività	173
7.3.6	Sequence diagram di dettaglio - Storage	174
7.4	Deployment Diagram	175
8	Testing	177
8.1	Test funzionali e non funzionali	177
8.2	Testing dei servizi	177
8.2.1	Testing microservizio AuthenticationService	178

8.2.2 Testing microservizio UsersService	179
8.2.3 Testing microservizio PatientsService	183
8.2.4 Testing microservizio ReportsService	185
8.2.5 Testing microservizio Attività	188
8.3 Test funzionali.....	189
8.3.1 Testing operazione di login.....	189
8.3.2 Testing operazione di creazione utente.....	190
8.3.3 Testing operazione di ottenimento utente.....	192
8.3.4 Testing operazione di eliminazione utente.....	192
8.3.5 Testing operazione di creazione paziente	193
8.3.6 Testing operazione di modifica paziente.....	193
8.3.7 Testing operazione di creazione report.....	194
8.3.8 Testing operazione di modifica report.....	195
8.4 Test non funzionali.....	196
8.4.1 Sicurezza.....	196
8.4.2 Scalabilità	198
8.4.3 Usabilità	200
8.4.4 Affidabilità.....	201
8.4.5 Manutenibilità	201
9 Presentazione del prodotto	202
9.1 Accesso all'applicazione	202
9.1.1 Presentazione	202
9.1.2 Autenticazione	202
9.2 Dashboard	203
9.3 Profilo personale.....	204
9.4 Utenti	205
9.5 Pazienti	206
9.5.1 Scheda del paziente	207
9.6 Report.....	208
9.7 Colleghi	210
9.8 Attività	211

1 Introduzione

Contestualizzazione del progetto e illustrazione dell'idea di partenza

1.1 Contesto dell'applicazione

FavolApp è concepita come un'innovativa piattaforma digitale per la gestione dei pazienti con autismo e disturbi del comportamento e dell'apprendimento. L'obiettivo è creare uno strumento che permetta ai supervisori di monitorare e gestire il lavoro dei tutor in modo efficiente e dettagliato. I tutor potranno utilizzare l'applicazione per creare schede interattive personalizzate, progettate per stimolare l'apprendimento dei pazienti, e per redigere report dettagliati sui loro progressi. Inoltre, il sistema prevede un pannello di amministrazione centralizzato che faciliterà il monitoraggio delle attività e l'analisi dei dati, consentendo interventi tempestivi e mirati. L'applicazione mira a migliorare la comunicazione e la coesione tra supervisori e tutor, ottimizzando così l'efficacia degli interventi terapeutici.

1.2 Organizzazioni Non Lucratивe di Utilità Sociale (ONLUS)

Le cooperative sociali ONLUS (Organizzazioni Non Lucratивe di Utilità Sociale) sono entità cruciali per la promozione del benessere comunitario e l'inclusione sociale in Italia. Regolate dalla legge 381/1991, queste cooperative si distinguono in due tipologie principali, ciascuna con specifici obiettivi e ambiti di intervento.

1.2.1 Cooperative di tipo A

Le cooperative di tipo A sono focalizzate sulla fornitura di servizi socio-sanitari, educativi, assistenziali e formativi. Questi servizi mirano a supportare varie categorie di persone in difficoltà, tra cui anziani, minori, disabili e individui con problemi di salute o a rischio di esclusione sociale. Il loro obiettivo è quello di migliorare la qualità della vita di queste persone, offrendo assistenza e opportunità di crescita personale e sociale attraverso programmi dedicati e interventi mirati.

1.2.2 Cooperative di tipo B

Le cooperative di tipo B, invece, si concentrano sull'inserimento lavorativo di persone svantaggiate. Tra questi rientrano disabili, ex detenuti, tossicodipendenti e individui a rischio di esclusione dal mercato del lavoro. L'obiettivo è di creare opportunità di lavoro e di integrazione sociale, favorendo l'inclusione attiva di queste persone nel tessuto economico e sociale. Le cooperative di tipo B svolgono un ruolo fondamentale nel fornire formazione professionale e opportunità lavorative, contribuendo così a ridurre la marginalizzazione e a promuovere l'autonomia delle persone coinvolte.

2 Avvio della progettazione

Gli obiettivi del progetto vengono definiti chiaramente, includendo la descrizione degli attori coinvolti e le specifiche funzioni che ciascuno di essi dovrà svolgere

2.1 Attori e obiettivi generali

Si descrivono, ad alto livello, gli attori del sistema e i relativi obiettivi generali:

Supervisore

- È in grado di registrare **Tutor** e **Supervisori** sulla piattaforma, inserendo le loro informazioni personali e un'email valida per l'invio delle credenziali d'accesso.
- Può modificare le informazioni personali dei **Supervisori** e dei **Tutor** già registrati.
- Ha la possibilità di aggiornare e visualizzare le proprie informazioni tramite la pagina del profilo.
- Può visualizzare di tutti gli utenti, le loro informazioni personali, le loro attività e i loro report.
- Può aggiungere e gestire i pazienti sulla piattaforma in maniera interattiva.
- È in grado di visualizzare e modificare i report creati da sé stesso, nonché di visualizzare i report realizzati dai colleghi e da tutti gli utenti.
- Attraverso un pannello dedicato, può supervisionare tutte le attività effettuate sulla piattaforma dagli altri **Supervisori** e dai **Tutor**.

Tutor

- Ha la possibilità di aggiornare e visualizzare le proprie informazioni tramite la pagina del profilo.
- Può visualizzare le informazioni personali di tutti i suoi colleghi.
- È in grado di creare report dettagliati sui propri pazienti.
- Ha la possibilità di visualizzare e modificare i propri report, nonché di visualizzare i report dei colleghi riguardanti pazienti in comune.

2.2 Funzionalità generali del sistema software

Il sistema software “Favolapp” deve poter:

- Consentire la registrazione e l'autenticazione degli utenti tramite un sistema di gestione delle identità sicuro.
- Permettere agli utenti di accedere e visualizzare le proprie informazioni personali nella pagina dedicata al profilo.

- Offrire la possibilità di consultare elenchi di utenti, pazienti e colleghi, con funzionalità di ricerca e filtri basati su nome, cognome o codice fiscale.
- Fornire una scheda specifica per visualizzare dettagliatamente le informazioni di ciascun paziente.
- Generare automaticamente log dettagliati che tracciano tutte le operazioni eseguite sulla piattaforma, facilitando la revisione delle attività svolte.

2.3 Assunzioni e dipendenze

Il primo utente della piattaforma, con il ruolo di Supervisore, dovrà essere creato da uno sviluppatore. Dopo questo passaggio, il Supervisore avrà la possibilità di aggiungere nuovi utenti alla piattaforma, assegnando loro i ruoli di Tutor o altri Supervisori, in base alle necessità operative.

La relazione tra "colleghi" dovrà essere automatizzata, permettendo al sistema di creare un'associazione tra Supervisori e Tutor che condividono i medesimi pazienti, senza la necessità di interventi manuali.

Essendo una web app, l'applicazione dovrà essere accessibile da qualsiasi browser, sia su desktop che su dispositivi mobili, richiedendo quindi una connessione a Internet per il suo utilizzo. Inoltre, l'interfaccia grafica dovrà essere progettata in modo responsive, per garantire un'esperienza utente ottimale su schermi di tutte le dimensioni, dai computer ai dispositivi mobili come smartphone e tablet.

2.4 Storie utente – Epic (Requisiti funzionali)

Le user stories sono una tecnica utilizzata per descrivere i requisiti funzionali di un sistema dal punto di vista dell'utente finale. Rappresentano piccoli incrementi di funzionalità che contribuiscono alla realizzazione di un prodotto e sono espresse in un linguaggio semplice e comprensibile. Ogni user story è formulata seguendo uno schema standard:

Come [ruolo], io voglio [obiettivo], in modo da [beneficio].

Questa struttura permette di focalizzarsi sugli obiettivi degli utenti e sui benefici ottenibili dall'implementazione di una determinata funzionalità, mantenendo la descrizione a un livello chiaro e accessibile a tutti i membri del team.

2.4.1 Elementi chiave

Ogni user story è solitamente arricchita da alcune informazioni chiave che ne facilitano la gestione e la pianificazione:

- **ID (Identificativo):**

L'ID è un identificatore univoco associato a ciascuna user story. Viene utilizzato per tracciare la storia nel sistema di gestione dei progetti e per fare riferimento alla stessa in modo rapido e preciso. L'ID consente una facile ricerca e organizzazione delle storie utente, soprattutto in progetti di grandi dimensioni.

- **Priorità:**
La priorità indica l'importanza relativa di una user story rispetto alle altre. Questo valore aiuta il team di sviluppo a decidere quali funzionalità devono essere implementate per prime. La priorità viene solitamente stabilita dal Product Owner o da chi si occupa della gestione del progetto e varia a seconda delle necessità del business o degli utenti finali.
- **Story Points:**
Gli story points sono una misura che esprime la complessità, l'incertezza e lo sforzo richiesto per completare una user story. Non misurano direttamente il tempo, ma piuttosto la difficoltà di sviluppo. I team di sviluppo assegnano un punteggio in base a vari fattori, come le risorse necessarie, le dipendenze e i potenziali rischi. Viene comunemente utilizzata la scala di Fibonacci (1, 2, 3, 5, 8, ecc.) per incoraggiare stime più accurate.
- **Epic Link:**
L'epic link è il collegamento che associa una user story a un epic, che rappresenta un macro-requisito o un insieme più ampio di funzionalità. Gli epics sono solitamente suddivisi in più user stories più piccole e gestibili, mantenendo una visione d'insieme sul progetto e facilitando la tracciabilità dei progressi verso obiettivi più ampi.
- **Etichetta (o Tag):**
Le etichette o tag sono parole chiave assegnate a una user story per facilitarne l'organizzazione e la ricerca. Possono essere utilizzate per categorizzare le storie in base a ruoli, componenti del sistema, attori coinvolti o altri aspetti rilevanti. Le etichette permettono di raggruppare storie simili o di identificare rapidamente quelle che appartengono a un determinato contesto.

2.5 Requisiti funzionali delle interfacce esterne

2.5.1 Interfaccia utente

L'interfaccia utente della piattaforma deve essere progettata per garantire un'esperienza d'uso semplice e intuitiva. Gli utenti devono poter navigare facilmente tra i vari menu, pagine e pulsanti, trovando le funzionalità di cui hanno bisogno in modo immediato e senza difficoltà. Di seguito sono riportate le user stories per le due principali tipologie di utenti che interagiranno con il software:

Supervisore (SU)

Storia Utente	SU01
Titolo	Registrare Tutor e Supervisori
Descrizione	Come Supervisore, voglio poter registrare Tutor e Supervisori, in modo da inserirli nella piattaforma con le loro informazioni personali e un'email valida per l'invio delle credenziali di accesso.

Priorità	Alta
Story Points	8
Epic Link	Gestione utenti
Etichette	Supervisore

Tabella 2.1: Storia Utente: Supervisore – Gestione utenti.

Storia Utente	SU02
Titolo	Modificare informazioni di Tutor e Supervisori
Descrizione	Come Supervisore, voglio poter modificare le informazioni personali di Tutor e Supervisori già registrati, in modo da mantenere i dati aggiornati.
Priorità	Media
Story Points	5
Epic Link	Gestione utenti
Etichette	Supervisore

Tabella 2.2: Storia Utente: Supervisore – Gestione utenti.

Storia Utente	SU03
Titolo	Aggiornare e visualizzare il proprio profilo
Descrizione	Come Supervisore, voglio aggiornare e visualizzare le mie informazioni personali tramite la pagina del profilo, in modo da gestire i miei dati sulla piattaforma.
Priorità	Bassa
Story Points	3

Epic Link	Profilo utente
Etichette	Supervisore

Tabella 2.3: Storia Utente: Supervisore – Profilo utente.

Storia Utente	SU04
Titolo	Visualizzare informazioni di tutti gli utenti
Descrizione	Come Supervisore, voglio poter visualizzare le informazioni personali, le attività e i report di tutti gli utenti della piattaforma, in modo da supervisionare e monitorare il loro operato.
Priorità	Alta
Story Points	8
Epic Link	Supervisione
Etichette	Supervisore

Tabella 2.4: Storia Utente: Supervisore – Supervisione.

Storia Utente	SU05
Titolo	Aggiungere e gestire i pazienti
Descrizione	Come Supervisore, voglio aggiungere e gestire pazienti sulla piattaforma in maniera interattiva, in modo da mantenere aggiornati i dati clinici e organizzativi.
Priorità	Media
Story Points	5
Epic Link	Gestione pazienti
Etichette	Supervisore

Tabella 2.5: Storia Utente: Supervisore – Gestione pazienti.

Storia Utente	SU06
Titolo	Modificare e visualizzare i propri report
Descrizione	Come Supervisore, voglio poter visualizzare e modificare i report che ho creato, in modo da aggiornare i dati e monitorare l'evoluzione dei pazienti.
Priorità	Alta
Story Points	5
Epic Link	Gestione report
Etichette	Supervisore

Tabella 2.6: Storia Utente: Supervisore – Gestione report.

Storia Utente	SU07
Titolo	Visualizzare i report degli altri utenti
Descrizione	Come Supervisore, voglio poter visualizzare i report creati dai colleghi e dagli altri utenti, in modo da avere una visione completa della gestione dei pazienti e delle attività.
Priorità	Media
Story Points	8
Epic Link	Supervisione
Etichette	Supervisore

Tabella 2.7: Storia Utente: Supervisore – Supervisione.

Storia Utente	SU08
Titolo	Supervisionare attività su piattaforma
Descrizione	Come Supervisore, voglio supervisionare tutte le attività effettuate dai Tutor e dagli altri Supervisori sulla piattaforma, in modo da garantire il corretto svolgimento delle operazioni.
Priorità	Alta
Story Points	13
Epic Link	Supervisione
Etichette	Supervisore

Tabella 2.8: Storia Utente: Supervisore – Supervisione.

Tutor (TU)

Storia Utente	TU01
Titolo	Aggiornare e visualizzare il proprio profilo
Descrizione	Come Tutor, voglio poter aggiornare e visualizzare le mie informazioni personali tramite la pagina del profilo, in modo da gestire i miei dati sulla piattaforma.
Priorità	Bassa
Story Points	3
Epic Link	Profilo utente
Etichette	Tutor

Tabella 2.9: Storia Utente: Tutor – Profilo utente.

Storia Utente	TU02
Titolo	Visualizzare le informazioni personali dei colleghi
Descrizione	Come Tutor, voglio poter visualizzare le informazioni personali dei miei colleghi, in modo da avere un quadro completo dei membri del team con cui collaboro.
Priorità	Media
Story Points	5
Epic Link	Collaborazione
Etichette	Tutor

Tabella 2.10: Storia Utente: Tutor – Collaborazione.

Storia Utente	TU03
Titolo	Creare report sui pazienti
Descrizione	Come Tutor, voglio poter creare report dettagliati sui miei pazienti, in modo da documentare e tenere traccia della loro situazione clinica e delle attività svolte.
Priorità	Alta
Story Points	8
Epic Link	Gestione report
Etichette	Tutor

Tabella 2.11: Storia Utente: Tutor – Gestione report.

Storia Utente	TU04
Titolo	Modificare e visualizzare i propri report
Descrizione	Come Tutor, voglio poter visualizzare e modificare i miei report sui pazienti, in modo da aggiornare i dati secondo le necessità.
Priorità	Alta
Story Points	5
Epic Link	Gestione report
Etichette	Tutor

Tabella 2.12: Storia Utente: Tutor – Gestione report.

Storia Utente	TU05
Titolo	Visualizzare i report dei colleghi
Descrizione	Come Tutor, voglio poter visualizzare i report dei miei colleghi riguardanti i pazienti in comune, in modo da collaborare in modo efficace e informato.
Priorità	Media
Story Points	8
Epic Link	Collaborazione
Etichette	Tutor

Tabella 2.13: Storia Utente: Tutor – Collaborazione.

2.5.2 Interfaccia software

L'interfaccia software riguarda il modo in cui il frontend (realizzato in React) comunica con il backend (realizzato interamente con i servizi AWS).

L'interfaccia software è costituita dalle API fornite dalla libreria AWS Amplify, che consente al frontend di comunicare con il backend e interagire con i vari servizi AWS (ad esempio, per inviare richieste HTTP, effettuare chiamate GraphQL e gestire l'autenticazione).

2.5.3 Interfaccia hardware

Non vi è una componente hardware specifica che richieda interazioni dirette nel contesto dell'applicazione. Tuttavia, l'infrastruttura cloud (AWS) ospita il backend e il frontend su server remoti, e si può considerare che l'applicazione si interfacci con hardware (server AWS) gestito da AWS stessa. Non ci sono dispositivi hardware specifici connessi direttamente al sistema tramite l'applicazione.

2.5.4 Interfaccia di comunicazione

L'interfaccia di comunicazione si riferisce al modo in cui i dati vengono scambiati tra le varie componenti del sistema.

La distribuzione dei file frontend avviene in modo sicuro tramite un sistema che utilizza un certificato per garantire la comunicazione protetta via HTTPS. Questo assicura che gli utenti finali possano interagire con il frontend attraverso un canale sicuro.

Le chiamate API tra frontend e backend, oppure verso servizi esterni, avvengono anch'esse su connessioni HTTPS. Questo meccanismo di comunicazione protegge i dati trasmessi, preservandone sicurezza e integrità.

2.6 Requisiti di consegna

L'organizzazione ONLUS "Favola", che ci ha commissionato questo progetto, ci ha dato una scadenza di un mese e mezzo, in vista di un aumento del numero di pazienti. In particolare, la consegna di una demo è stata richiesta entro un mese.

2.7 Requisiti di riservatezza

Le funzionalità riservate al Supervisore non sono accessibili ai Tutor. Ad esempio, il Supervisore ha la possibilità di consultare le informazioni personali di tutti gli utenti registrati sulla piattaforma, mentre i Tutor possono visualizzare solo i dati personali dei propri colleghi. Pertanto, i Tutor non hanno accesso a determinate informazioni che sono esclusivamente disponibili per i Supervisori.

2.8 Requisiti non funzionali

I requisiti non funzionali definiscono le caratteristiche di qualità di un sistema, come sicurezza, usabilità, scalabilità e manutenibilità. Si concentrano sul come il sistema deve operare, contribuendo a garantire un'esperienza d'uso affidabile ed efficiente. Per specificare e testare questi requisiti, si utilizzano gli scenari di qualità, che comprendono i seguenti elementi:

- **Stimulus:** L'evento che attiva una risposta del sistema, ad esempio una richiesta di accesso o un aumento del carico.
- **Stimulus Source:** L'entità che genera lo stimolo, come un utente, un processo o un sistema esterno.
- **Response:** L'azione che il sistema esegue in risposta allo stimolo, ad esempio elaborare una richiesta o bloccare un accesso non autorizzato.
- **Response Measure:** Un parametro che consente di misurare l'efficacia della risposta, come il tempo di risposta o il numero di errori.
- **Environment:** Il contesto operativo in cui si verifica lo scenario, come condizioni normali o situazioni di carico elevato.
- **Artifact:** La parte del sistema interessata dallo stimolo, come il database, il backend o l'interfaccia utente.

Questo approccio consente di rappresentare in modo chiaro e testabile i requisiti di qualità, supportando sia il confronto con gli stakeholder in fase di analisi, sia la validazione durante lo sviluppo.

Tra i principali requisiti non funzionali troviamo:

- **Sicurezza**, che include autenticazione, protezione dei dati e integrità delle comunicazioni.
- **Scalabilità**, per garantire che il sistema gestisca un aumento delle richieste o degli utenti.
- **Usabilità**, che riguarda la facilità di utilizzo, comprensione e gradimento da parte degli utenti.
- **Affidabilità**, per assicurare che il sistema funzioni in modo stabile anche in condizioni critiche.
- **Manutenibilità**, per agevolare aggiornamenti, correzioni e l'aggiunta di nuove funzionalità.

2.8.1 Sicurezza

I requisiti di sicurezza garantiscono che l'applicazione protegga i dati degli utenti e impedisca l'accesso non autorizzato. Questi requisiti definiscono le misure per salvaguardare la privacy e la confidenzialità delle informazioni.

RNF01	L'applicazione deve impedire l'accesso a utenti non autenticati e proteggere i dati trasmessi.
Stimulus	Un utente non autenticato tenta di accedere a un'API protetta.
Stimulus Source	Utente malintenzionato o client senza credenziali valide.
Response	Il sistema deve rifiutare la richiesta e restituire un errore.
Response Measure	Nessuna informazione deve essere restituita all'utente non autenticato.
Environment	Accesso tramite browser.
Artifact	Endpoint API protetti del backend.

Tabella 2.14: Requisito non funzionale per impedire l'accesso a utenti non autenticati.

RNF02	I dati trasmessi devono essere protetti da accessi non autorizzati.
Stimulus	Un attaccante tenta di intercettare i dati in transito.
Stimulus Source	Attaccante in grado di eseguire attacchi man-in-the-middle.
Response	I dati intercettati non devono essere leggibili.
Response Measure	I dati devono risultare cifrati tramite protocolli di sicurezza standard (ad esempio, TLS).
Environment	Comunicazione tra frontend e backend in modalità operativa normale.
Artifact	Traffico di rete tra client e server.

Tabella 2.15: Requisito non funzionale per la protezione dei dati trasmessi.

2.8.2 Scalabilità

I requisiti di scalabilità assicurano che l'applicazione possa gestire un aumento del traffico o della richiesta di risorse senza compromettere le prestazioni. Questi requisiti stabiliscono come il sistema si adatti a crescere in modo efficiente.

RNF03	Il sistema deve gestire un aumento delle richieste senza degradare le prestazioni.
Stimulus	Un aumento del carico pari al 100% delle richieste simultanee rispetto al normale utilizzo.
Stimulus Source	Utenti che accedono all'applicazione durante un picco di traffico.
Response	L'applicazione deve rispondere alle richieste con un tempo medio di risposta inferiore a 3 secondi.
Response Measure	Percentuale di richieste servite senza timeout superiore al 95%.
Environment	Ambiente di produzione durante un evento ad alto traffico.
Artifact	Server backend e risorse del sistema.

Tabella 2.16: Requisito non funzionale per la gestione del carico.

2.8.3 Usabilità

I requisiti di usabilità si concentrano sull'esperienza dell'utente, assicurandosi che l'interfaccia sia intuitiva, facile da usare e accessibile anche per gli utenti meno esperti. Questi requisiti stabiliscono standard per migliorare l'interazione e la soddisfazione dell'utente.

RNF04	L'applicazione deve essere facilmente comprensibile e navigabile per gli utenti.
Stimulus	Un nuovo utente accede per la prima volta all'applicazione.
Stimulus Source	Utente senza formazione specifica sull'uso del sistema.

Response	L'applicazione deve soddisfare un punteggio di accessibilità superiore o uguale a 90 nella valutazione automatica.
Response Measure	Il punteggio di accessibilità misurato deve essere almeno il 90%, considerando fattori come navigabilità, etichette accessibili, contrasto dei colori e struttura semantica.
Environment	Verifica eseguita tramite un tool integrato nel browser, in condizioni operative normali.
Artifact	Interfaccia utente (frontend).

Tabella 2.17: Requisito non funzionale per l'usabilità del sistema.

2.8.4 Affidabilità

I requisiti di affidabilità garantiscono che il sistema funzioni senza interruzioni significative e che possa recuperare rapidamente in caso di malfunzionamenti. Questi requisiti definiscono le aspettative di disponibilità e continuità del servizio.

RNF05	Il sistema deve garantire una disponibilità continua durante le ore di utilizzo principali.
Stimulus	Il server backend subisce un riavvio improvviso.
Stimulus Source	Problema tecnico o aggiornamento.
Response	L'applicazione deve recuperare il servizio entro 5 minuti senza perdita di dati.
Response Measure	Percentuale di uptime superiore al 99%.
Environment	Ambiente operativo normale durante l'orario lavorativo.
Artifact	Intero sistema backend.

Tabella 2.18: Requisito non funzionale per l'affidabilità del sistema.

2.8.5 Manutenibilità

I requisiti di manutenibilità assicurano che il sistema sia facile da aggiornare, correggere e migliorare nel tempo. Questi requisiti definiscono le modalità di gestione delle modifiche senza compromettere la stabilità e la qualità del sistema.

RNF06	Le modifiche al codice devono poter essere effettuate rapidamente e senza introdurre regressioni.
Stimulus	Uno sviluppatore implementa una nuova funzionalità.
Stimulus Source	Sviluppatore con accesso al repository del codice.
Response	La modifica deve essere completata, testata e distribuita in meno di 24 ore.
Response Measure	Percentuale di modifiche implementate senza errori superiori al 95%.
Environment	Ambiente di sviluppo e testing.
Artifact	Moduli software interessati dalla modifica.

Tabella 2.19: Requisito non funzionale per la manutenibilità del sistema.

2.9 Stima dei costi

Gli Use Case Points (UCP) rappresentano una tecnica di stima software utilizzata per prevedere le dimensioni e lo sforzo necessario per completare un progetto di sviluppo. Questo metodo si basa sull'uso di casi d'uso, una componente centrale dell'UML (Unified Modeling Language), comunemente impiegata per la progettazione e lo sviluppo del software. Gli UCP permettono di stimare in modo accurato il lavoro richiesto, partendo dai requisiti funzionali espressi attraverso i casi d'uso del sistema.

La stima UCP si basa su una combinazione di fattori, tra cui la complessità dei casi d'uso e degli attori coinvolti, che vengono classificati in categorie (semplici, medi o complessi). Inoltre, tiene conto di specifiche considerazioni tecniche ed ambientali, come l'integrità dei dati, la sicurezza e il livello di esperienza del team di sviluppo, per adattare la stima alle reali condizioni del progetto.

La formula per il calcolo dell'UCP coinvolge vari step:

- **Determinazione del Peso dei Casi d'Uso:** Ogni caso d'uso viene classificato in base alla sua complessità e associato a un punteggio.

- **Determinazione del Peso degli Attori:** Ogni attore, cioè qualsiasi entità che interagisce con il sistema, è valutato in base al tipo di interazione e assegnato un peso.
- **Calcolo del Fattore Tecnico (TF):** Attraverso una serie di indicatori tecnici, come la distribuzione del sistema, l'affidabilità e la sicurezza, si assegna un punteggio che influisce sulla stima complessiva.
- **Calcolo del Fattore Ambientale (EF):** Fattori come l'esperienza del team, la familiarità con il linguaggio di programmazione e gli strumenti di sviluppo sono considerati per rendere la stima più precisa.

Il prodotto finale, espresso in UCP, rappresenta una stima delle dimensioni del software che può essere convertita in stime di tempo e costo per lo sviluppo. Ad esempio, una volta calcolato il numero di UCP, viene applicato un coefficiente per stimare lo sforzo in termini di ore di lavoro, che varia a seconda della complessità del progetto e delle competenze del team.

$$UCP = (UUCW + UAW) * TCF * ECF$$

2.9.1 Unadjusted Use Case Weight (UUCW)

I valori di riferimento assegnati a ciascun caso d'uso, noti come Use Case Classification, sono suddivisi in tre categorie in base alla complessità del caso d'uso e al numero di transazioni coinvolte:

- **Semplice (Simple):** casi d'uso con 1-3 transazioni, ai quali viene assegnato un peso di 5.
- **Medio (Average):** casi d'uso con 4-7 transazioni, ai quali viene assegnato un peso di 10.
- **Complesso (Complex):** casi d'uso con 8 o più transazioni, ai quali viene assegnato un peso di 15.

Di seguito la tabella di classificazione dei casi d'uso dell'applicazione:

Macro categoria	Caso d'uso	Classificazione	Peso
Gestione informazioni personali	Visualizza, Modifica	Simple	5
Gestione degli utenti	Registra utente, Modifica utente, Visualizza utente, Elimina utente, Ricerca utente	Average	10
Gestione pazienti	Aggiungi paziente (1), Modifica paziente (2), Elimina paziente (1), Visualizzare pazienti (3), Ricerca pazienti (1)	Average	10
Gestione dei report	Crea (1), Modifica (1), Elimina (1), Visualizza (2), Ricerca (1)	Average	10

Gestione colleghi	Visualizza colleghi	Simple	5
Gestione attività	Visualizza (1), Applica filtro (2)	Average	10

Tabella 2.20: Classificazione dei casi d'uso.

Formula per il calcolo dell'UUCW:

$$UUCW = (TotalNo.\text{of}SimpleUC * 5) + \\ (TotalNo.\text{of}AverageUC * 10) + \\ (TotalNo.\text{of}ComplexUC * 15)$$

Complessità	Peso	No. UC	No.UC * Peso
Simple	5	2	10
Average	10	30	30
Complex	15	0	0
Total			40

Tabella 2.21: Calcolo UCCW.

2.9.2 Unadjusted Actor Weight (UAW)

L'Unadjusted Actor Weight (UAW) si basa sulla tipologia e sulla complessità degli attori che interagiscono con il sistema. Ogni attore è classificato in base al loro impatto sulle funzionalità del sistema e il loro peso viene determinato come segue:

- **Simple Actor:** Un attore che interagisce con il sistema in modo semplice e diretto. (Peso 1)
- **Average Actor:** Un attore che ha interazioni moderate e diverse con il sistema. (Peso 2)
- **Complex Actor:** Un attore che interagisce con il sistema in modo complesso e ha un ampio set di funzioni da gestire. (Peso 3)

Di seguito la tabella di classificazione degli attori:

Attore	Descrizione	Complessità
Supervisore	Ha la possibilità di gestire utenti, pazienti, report e attività. Interagisce con diverse funzionalità e ha molte responsabilità.	Complex
Tutor	Interagisce con il sistema per gestire i propri pazienti e report. Ha un set di funzioni più limitato rispetto al supervisore.	Average

Tabella 2.22: Classificazione degli attori.

Formula per il calcolo dell'UAW:

$$UAW = (TotalNo.\ of SimpleActors * 1) + (TotalNo.\ of AverageActors * 2) + (TotalNo.\ of ComplexActors * 3)$$

Complessità	Peso	No. Actors	No.Actors * Peso
Simple	1	0	0
Average	2	1	2
Complex	3	1	3
Total			5

Tabella 2.23: Calcolo UAW.

2.9.3 Technical Complexity Factor (TCF)

Il Technical Complexity Factor è calcolato mediante l'assegnazione di punteggi a 13 fattori tecnici, ognuno dei quali contribuisce a delineare la complessità del sistema in fase di sviluppo. Questi fattori comprendono aspetti come la distribuzione del sistema, le performance richieste, la manutenibilità e la sicurezza, tutti cruciali per garantire un'applicazione robusta e performante. Attraverso la valutazione attenta di questi elementi, è possibile ottenere una stima più precisa dello sforzo necessario per completare il progetto, tenendo conto delle sfide tecniche che potrebbero influenzare il percorso di sviluppo.

Di seguito la tabella dei fattori tecnici e il loro punteggio da noi assegnato:

Fattore	Descrizione	Punteggio (0 – 5)	Valore Ponderato
T1	Sistema distribuito	3	2.0
T2	Tempo di risposta/obiettivi prestazionali	2	1.0
T3	Efficienza dell'utente finale	5	1.0
T4	Complessità di elaborazione interna	2	1.0
T5	Riusabilità del codice	4	1.0
T6	Facile da installare	5	0.5
T7	Facile da usare	3	0.5
T8	Portabilità su altre piattaforme	5	2.0
T9	Manutenzione del sistema	3	1.0
T10	Elaborazione simultanea/parallela	4	1.0
T11	Funzioni di sicurezza	5	1.0
T12	Accesso per terze parti	2	1.0
T13	Formazione dell'utente finale	3	1.0

Tabella 2.24: Assegnazione del punteggio ai fattori.

Effettuiamo il calcolo del TF come somma dei punteggi assegnati valori ponderati:

$$\begin{aligned}
 TF = & 3 * 2 + 2 * 1 + 5 * 1 + 2 * 1 + 4 * 1 + \\
 & + 5 * 0.5 + 3 * 0.5 + 5 * 2 + 3 * 1 + \\
 & + 4 * 1 + 5 * 1 + 2 * 1 + 3 * 1 \\
 = & 50
 \end{aligned}$$

Effettuiamo il calcolo del TCF:

$$TCF = 0.6 + \left(\frac{TF}{100} \right) = 0.6 + \left(\frac{50}{100} \right) = 0.6 + 0.5 = 1.1$$

2.9.4 Environmental Complexity Factor (ECF)

L'Environmental Complexity Factor si basa su 8 fattori ambientali che riflettono il livello di esperienza, la familiarità con il processo di sviluppo, la motivazione del team e la stabilità dei requisiti. Ogni fattore viene valutato su una scala da 0 a 5, dove 0 indica nessuna esperienza o impatto, mentre 5 rappresenta una condizione ottimale o di massima esperienza. Questi punteggi vengono poi moltiplicati per pesi specifici, variabili a seconda dell'importanza del fattore.

Fattore	Descrizione	Punteggio (0 – 5)	Valore Ponderato
E1	Familiarità con il processo di sviluppo utilizzato	4	1.5
E2	Esperienza applicativa	4	0.5
E3	Esperienza di team orientata agli oggetti	2	1.0
E4	Capacità di lead analyst	2	0.5
E5	Motivazione del team	5	1.0
E6	Stabilità dei requisiti	4	2.0
E7	Personale part-time	1	-1.0
E8	Linguaggio di programmazione difficile	3	-1.0

Tabella 2.25: Assegnazione del punteggio ai fattori.

Effettuiamo il calcolo del EF come somma del punteggio assegnato per valori ponderati:

$$\begin{aligned}
 EF = & 4 * 1.5 + 4 * 0.5 + 2 * 1 + 2 * 0.5 + 5 * 1 + \\
 & + 4 * 2 + 1 * (-1) + 3 * (-1) \\
 = & 20
 \end{aligned}$$

Effettuiamo il calcolo del ECF:

$$ECF = 1.4 + (-0.03 * EF) = 1.4 + (-0.03 * 20) = 1.4 - 0.6 = 0.8$$

2.9.5 Total Use Case Points (UCP)

Dopo aver definito le dimensioni non aggiustate del progetto, ossia l'UUCW e l'UAW, e aver calcolato i fattori di complessità tecnica (TCF) e ambientale (ECF), si può calcolare l'UCP utilizzando la formula seguente:

$$UCP = (UUCW + UAW) * TCF * ECF$$

Sostituendo i valori calcolati nelle operazioni precedenti avremo:

$$UCP = (40 + 5) * 1.1 * 0.8 = 39.6$$

Una volta determinato l'UCP, si può stimare il numero di ore necessarie per lo sviluppo. Supponendo che ogni Use Case richieda in media 8 ore di lavoro, il totale delle ore di sviluppo sarà:

$$T_h = UCP * 8 = 39.6 * 8 = 316.8h$$

Considerando che ogni membro del team lavora circa 40 ore settimanali e che il team è composto da 2 persone, il totale delle ore di lavoro settimanali del gruppo sarà:

$$TW_h = 40 \times 2 = 80h/\text{settimana}$$

Infine, il numero di settimane necessarie per completare il progetto si calcola dividendo il totale delle ore di lavoro (T_h) per le ore lavorative settimanali del team (TW_h):

$$\frac{T_h}{TW_h} = \frac{316.8}{80} = 3.96 \approx 4$$

Pertanto, si stima che lo sviluppo del progetto richiederà circa 4 settimane di lavoro, suddivise in circa 2-3 iterazioni di sviluppo.

2.10 System Context Diagram

Il System Context Diagram (Figura 2.1), offre una panoramica ad alto livello delle interazioni tra il sistema e i suoi attori esterni, mostrando come il sistema comunica con le entità circostanti. Nel diagramma, viene rappresentato il sistema centrale, evidenziando i principali attori coinvolti e i flussi di dati che intercorrono tra loro. Il diagramma è utile per comprendere il perimetro del sistema, senza entrare nei dettagli tecnici dei collegamenti interni o delle implementazioni specifiche.

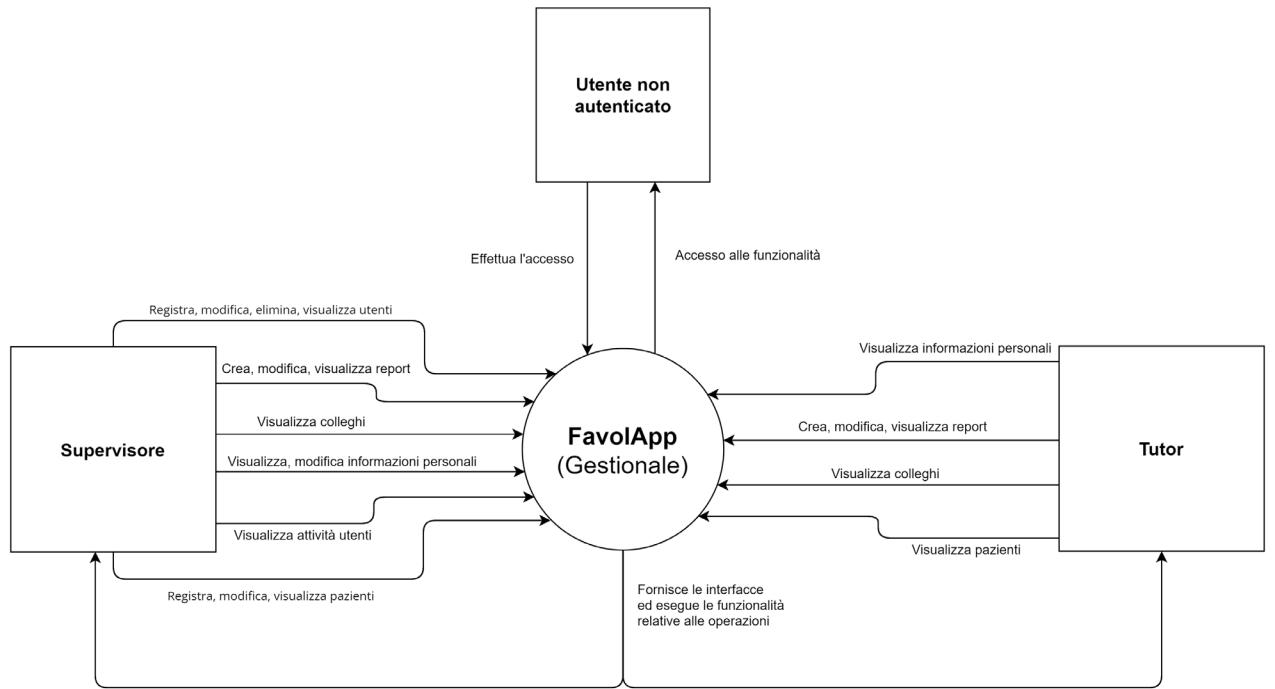


Figura 2.1: System Context Diagram di FavolApp.

3 Analisi dei requisiti

In questa fase si ha l'obiettivo di chiarire e documentare le funzioni e i servizi che vengono offerti dal prodotto software in esame

3.1 Modello dei Casi d'uso

I diagrammi di casi d'uso sono uno strumento essenziale per rappresentare graficamente le interazioni tra gli utenti e il sistema, offrendo una visione chiara e sintetica delle attività che possono essere svolte all'interno dell'applicazione. Ogni caso d'uso descrive un insieme specifico di funzionalità che soddisfano determinati requisiti degli attori, facilitando la comprensione delle dinamiche operative e dei processi fondamentali. Nei seguenti diagrammi, sono illustrate le azioni principali che i due attori, Supervisore e Tutor, possono eseguire all'interno dell'applicazione, evidenziando le loro responsabilità e interazioni con il sistema. Questi diagrammi aiutano a definire chiaramente i compiti di ciascun ruolo, supportando l'efficace organizzazione delle funzionalità.

3.1.1 Casi d'uso - Autenticazione

Di seguito è illustrato il diagramma di casi d'uso relativo al processo di autenticazione, in cui l'utente, che può assumere il ruolo di Tutor o Supervisore, interagisce con il sistema per accedere alle funzionalità riservate.

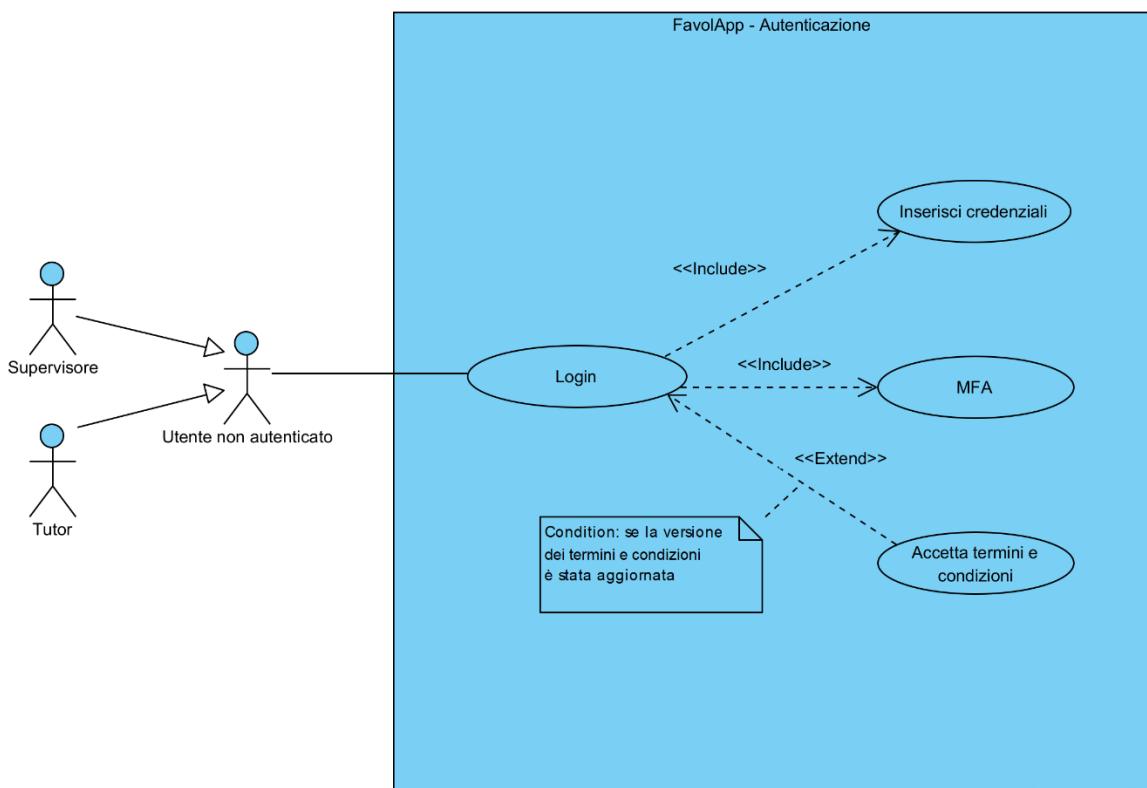


Figura 3.1: Casi d'uso inerenti alla fase di autenticazione.

Di seguito i casi d'uso in dettaglio che riguardano la fase di autenticazione:

Nome	Login
Obiettivo	Un utente vuole eseguire l'accesso alla piattaforma.
Attori	Utente non autenticato
Descrizione	L'utente non autenticato, che sia un Supervisore o un Tutor, deve eseguire l'accesso alla piattaforma completando delle challenge di sicurezza.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere stato registrato alla piattaforma da un Supervisore. • L'utente, al primo accesso, deve essere a conoscenza delle credenziali d'accesso ottenute tramite un email.
Post-condizioni	L'utente è autenticato e può accedere all'area riservata.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla piattaforma tramite il browser. 2. Inserisce le credenziali d'accesso nel form presentato. 3. Se le credenziali d'accesso sono corrette deve inserire il codice OTP ottenuto dall'app di autenticazione. 4. Accesso effettuato correttamente.
Estensioni o Flussi Alternativi	<p>Se l'utente accede per la prima volta deve accedere inserendo le credenziali d'accesso ottenute tramite email e deve configurare l'applicazione di autenticazione.</p> <p>Se i termini e condizioni vengono aggiornati, l'utente deve riaccettarli come step finale per effettuare l'accesso alla piattaforma.</p> <p>Se le credenziali d'accesso o l'otp non sono corretti allora verrà mostrato un alert che segnala l'errore.</p>
Requisiti speciali	Se non vengono accettati i termini e condizioni l'utente non può accedere alla piattaforma.
Frequenza di utilizzo	Media
Note Aggiuntive	Ogni operazione durante la fase di autenticazione, viene validata tramite il servizio di autenticazione.

Tabella 3.1: Dettagli caso d'uso Login.

3.1.2 Casi d'uso – Pagina del profilo personale

Il diagramma di casi d'uso illustrato di seguito descrive le funzionalità disponibili nella pagina del profilo personale di un utente. L'utente può visualizzare le proprie informazioni e caricare un avatar. Se l'utente ha il ruolo di Supervisore, può inoltre modificare le proprie informazioni personali.

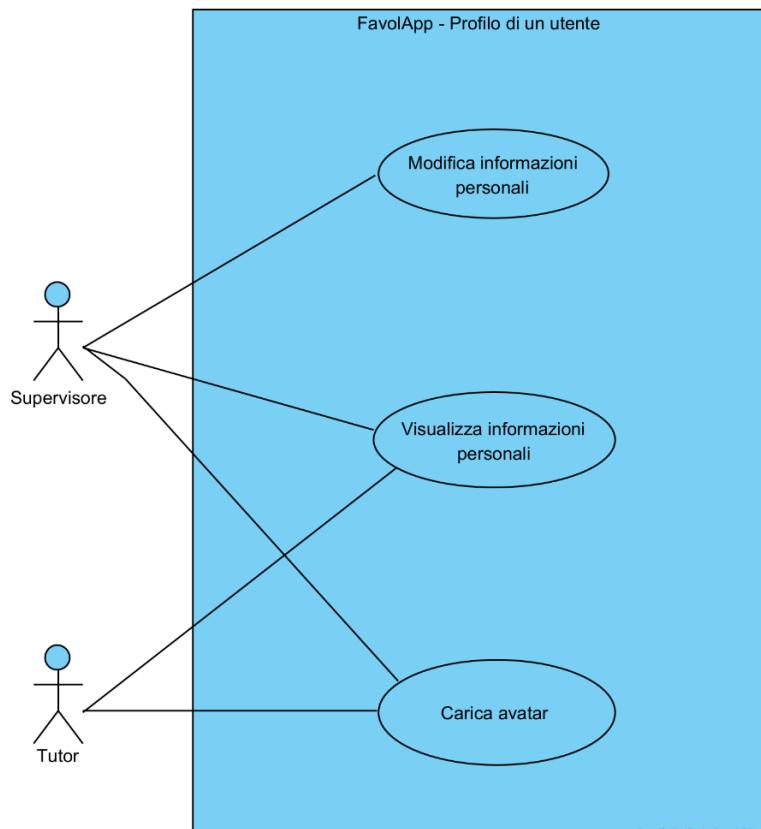


Figura 3.2: Casi d'uso inerenti alla pagina del profilo personale.

Di seguito i casi d'uso in dettaglio che riguardano la pagina del profilo personale:

Nome	Modifica informazioni personali
Obiettivo	Un Supervisore vuole modificare le proprie informazioni.
Attori	Supervisore
Descrizione	Il Supervisore nella pagina del profilo vuole modificare le proprie informazioni.
Pre-condizioni	<ul style="list-style-type: none">• L'utente deve essere autenticato.• L'utente deve avere il ruolo di Supervisore.

Post-condizioni	I dati dell’utente risulteranno modificati e visualizzabili nella pagina del profilo.
Scenario principale	<ol style="list-style-type: none"> 1. L’utente accede alla sezione Profilo dal menu di navigazione. 2. Clicca sul bottone “Modifica” in corrispondenza delle informazioni personali. 3. Viene mostrato un dialog dove può effettuare le modifiche delle informazioni.
Estensioni o Flussi Alternativi	<p>Nel caso in cui le informazioni dell’utente (nome, cognome, data di nascita, comune, provincia) non dovessero corrispondere al codice fiscale, verrà mostrato un errore.</p> <p>Se la modifica, lato backend, non avviene correttamente verrà mostrato un errore.</p> <p>L’utente può anche non effettuare alcuna modifica per cui ha la possibilità di chiudere il dialog.</p>
Requisiti speciali	Nessuno
Frequenza di utilizzo	Bassa
Note Aggiuntive	Nessuna

Tabella 3.2: Dettagli caso d’uso per la modifica delle informazioni personali.

Nome	Visualizza informazioni personali
Obiettivo	Un Supervisore o un Tutor vogliono visualizzare le proprie informazioni personali.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor nella pagina del profilo vogliono visualizzare le proprie informazioni.
Pre-condizioni	<ul style="list-style-type: none"> • L’utente deve essere autenticato.
Post-condizioni	I dati dell’utente saranno esposti sulla pagina del profilo.
Scenario principale	<ol style="list-style-type: none"> 1. L’utente accede alla sezione Profilo dal menu di navigazione. 2. Le informazioni personali verranno mostrate nella pagina del profilo.
Estensioni o Flussi Alternativi	Nessuno

Requisiti speciali	Nessuno
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.3: Dettagli caso d'uso per la visualizzazione delle informazioni personali.

Nome	Carica avatar
Obiettivo	Un Supervisore o un Tutor vogliono caricare un'immagine come avatar da mostrare sul proprio profilo.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor vogliono caricare e quindi modificare il proprio avatar che verrà mostrato sulla pagina del profilo.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato.
Post-condizioni	L'avatar dell'utente sarà esposto sulla pagina del profilo.
Scenario principale	<ol style="list-style-type: none"> L'utente accede alla sezione Profilo dal menu di navigazione. Clicca sull'avatar per caricare un'immagine. Seleziona l'immagine dal proprio PC tramite un dialog del sistema operativo. Selezionata l'immagine, viene visualizzata in dialog sull'applicazione. Cliccando su "Salva" l'immagine verrà caricata e visualizzata sul profilo.
Estensioni o Flussi Alternativi	<p>Se l'immagine selezionata non è valida, verrà visualizzato un errore.</p> <p>Se l'operazione di caricamento non va a buon fine, verrà visualizzato un errore.</p> <p>Se l'utente vuole annullare l'operazione può cliccare su "Annulla".</p>
Requisiti speciali	Nessuno
Frequenza di utilizzo	Bassa
Note Aggiuntive	Nessuna.

Tabella 3.4: Dettagli caso d'uso per il caricamento dell'avatar.

3.1.3 Casi d'uso – Pagina del profilo di un altro utente

Il diagramma di casi d'uso illustrato di seguito descrive le funzionalità disponibili nella pagina del profilo di un altro utente, visualizzabile da un utente autenticato. Quest'ultimo può consultare le informazioni personali dell'altro utente, accedere alla lista dei pazienti che hanno in comune e visualizzare i report relativi a tali pazienti. Inoltre, se l'utente autenticato ha il ruolo di Supervisore, può visualizzare tutti i pazienti assegnati all'altro utente.

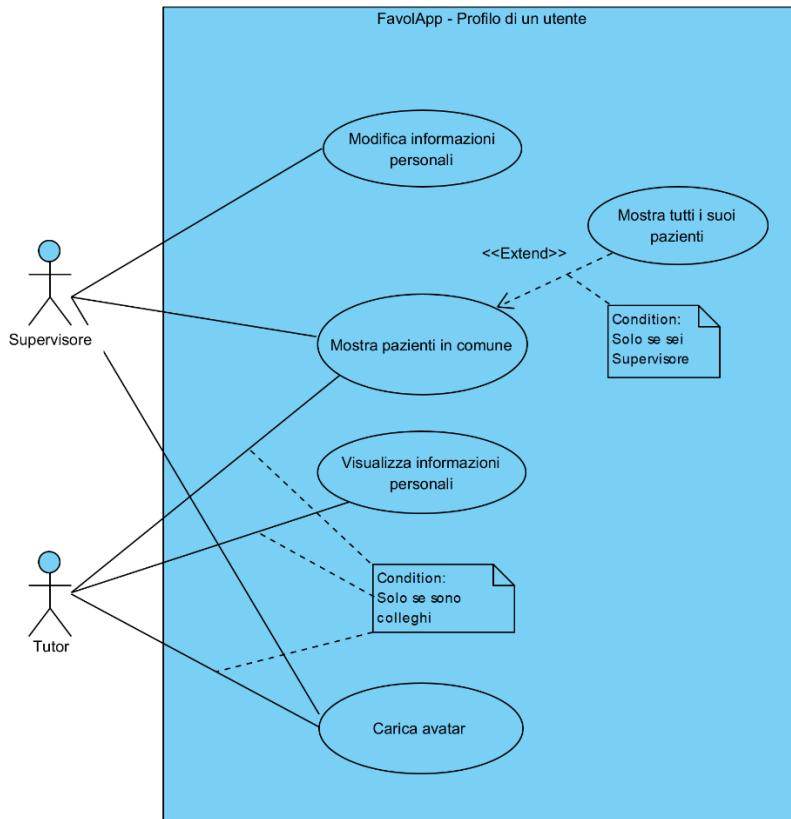


Figura 3.3: Casi d'uso inerenti alla pagina del profilo di un altro utente.

Di seguito i casi d'uso in dettaglio che riguardano la pagina del profilo di un altro utente:

Nome	Modifica informazioni utente
Obiettivo	Il Supervisore vuole modificare le informazioni dell'utente a cui appartiene la pagina del profilo.
Attori	Supervisore
Descrizione	Il Supervisore vuole modificare le informazioni dell'utente che sta visualizzando.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere un Supervisore.

Post-condizioni	Le informazioni modificate dal Supervisore saranno modificate sia sul database che mostrate sulla pagina del profilo utente.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina del profilo di un utente. 2. Clicca su "modifica" in corrispondenza delle informazioni dell'utente. 3. Tramite un form modifica le informazioni dell'utente. 4. Verifica che le informazioni inserite siano corrette. 5. Cliccando su "modifica" nel form, avvierà un processo sul backend che porterà alla modifica delle informazioni.
Estensioni o Flussi Alternativi	<p>Se le informazioni inserite sono scorrette allora verrà mostrato un messaggio di errore. Il controllo viene eseguito principalmente sul Codice Fiscale: se le informazioni utente non coincidono con il Codice Fiscale viene mostrato errore.</p> <p>Se l'operazione di modifica non va a buon fine, verrà visualizzato un errore.</p> <p>Se l'utente vuole annullare l'operazione può cliccare su "Annulla".</p>
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Bassa
Note Aggiuntive	Nessuna

Tabella 3.5: Dettagli caso d'uso per la modifica delle informazioni di un utente.

Nome	Mostra pazienti in comune
Obiettivo	Un Supervisore o un Tutor vogliono visualizzare i pazienti in comune.
Attori	Supervisore, Tutor
Descrizione	Un Supervisore o un Tutor tramite la pagina del profilo vogliono visualizzare i pazienti che hanno in comune.

Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato. L'utente Tutor che visualizza il profilo e l'utente visualizzato devono essere colleghi, cioè, devono avere dei pazienti in comune. L'utente Supervisore può visualizzare sempre i pazienti dell'utente proprietario della pagina del profilo.
Post-condizioni	Quando si clicca su “mostra pazienti” viene visualizzata una sezione per visualizzare i pazienti in comune.
Scenario principale	<ol style="list-style-type: none"> L'utente accede alla pagina del profilo di un altro utente. Clicca su “mostra pazienti”. Viene mostrata una sezione con una lista di pazienti in comune.
Estensioni o Flussi Alternativi	Se l'utente visualizzato non ha nessun paziente in comune con l'utente che sta visualizzando la pagina, e il suo ruolo è Tutor, allora il pulsante per visualizzare i pazienti non verrà mostrato. Altrimenti, se l'utente è un Supervisore verrà visualizzato.
Requisiti speciali	Solo se gli utenti hanno una relazione di colleghi, se hanno almeno un paziente in comune, possono accedere alla sezione.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.6: Dettagli caso d'uso per mostrare i pazienti in comune.

Nome	Mostra tutti i suoi pazienti
Obiettivo	Un Supervisore vuole visualizzare la lista di tutti i pazienti assegnati all'utente.
Attori	Supervisore
Descrizione	Un Supervisore tramite la sezione dedicata alla lista dei pazienti vuole visualizzare tutti i pazienti che sono stati assegnati all'utente.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato. L'utente deve avere il ruolo di Supervisore.

Post-condizioni	Quando si clicca su “visualizza tutti i pazienti”, viene mostrata in una tabella tutti i pazienti assegnati all’utente.
Scenario principale	<ol style="list-style-type: none"> 1. L’utente accede alla pagina del profilo di un altro utente. 2. Clicca su “mostra pazienti”. 3. Viene mostrata una sezione con una lista di pazienti in comune. 4. Clicca su “visualizza tutti i pazienti” 5. Viene mostrata una tabella con tutti i pazienti assegnati all’utente.
Estensioni o Flussi Alternativi	Nessuno
Requisiti speciali	Solo se l’utente ha il ruolo di Supervisore
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.7: Dettagli caso d’uso per visualizzare tutti i pazienti assegnati all’utente.

Nome	Visualizza i suoi report
Obiettivo	Un Supervisore o un Tutor vogliono visualizzare i report sui pazienti creati da un altro utente.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor vogliono ottenere una lista di tutti i report scritti dall’utente visualizzato tramite una sezione dedicata.
Pre-condizioni	<ul style="list-style-type: none"> • L’utente deve essere autenticato. • Se l’utente è un Tutor, per visualizzare i report dell’altro utente, devono essere colleghi. • Se l’utente è un Supervisore, può visualizzare sempre i report dell’altro utente.
Post-condizioni	Verrà visualizzata una lista dei report realizzati dall’altro utente sui pazienti in comune.

Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina del profilo di un altro utente. 2. Clicca sul bottone “visualizza report”. 3. Viene mostrata una sezione dedicata dove vi è una tabella che mostra i report scritti dall'altro utente.
Estensioni o Flussi Alternativi	Nessuno
Requisiti speciali	Nessuno
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.8: Dettagli caso d'uso per visualizzare i report scritti dall'altro utente sui pazienti.

Nome	Visualizza informazioni utente
Obiettivo	Il Supervisore o il Tutor vogliono visualizzare le informazioni personali dell'utente.
Attori	Supervisore, Tutor
Descrizione	Un Supervisore o un Tutor accedendo alla pagina del profilo di un altro utente, possono visualizzare le sue informazioni personali.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Se il ruolo dell'utente che visualizza è Tutor, per visualizzare le informazioni dell'altro utente, devono essere colleghi.
Post-condizioni	Le informazioni dell'altro utente verranno visualizzate sulla pagina del profilo.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina del profilo di un altro utente. 2. Visualizza le informazioni dell'altro utente.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.9: Dettagli caso d'uso per visualizzare le informazioni personali dell'altro utente.

3.1.4 Casi d'uso – Pagina per la gestione degli utenti

Il diagramma dei casi d'uso illustrato di seguito descrive le operazioni disponibili per un Supervisore nella sezione dedicata alla gestione degli utenti. Il Supervisore può visualizzare l'elenco completo degli utenti e, per ciascun utente, consultare i dettagli, apportare modifiche o procedere alla loro eliminazione. Inoltre, ha la possibilità di registrare nuovi utenti nel sistema.

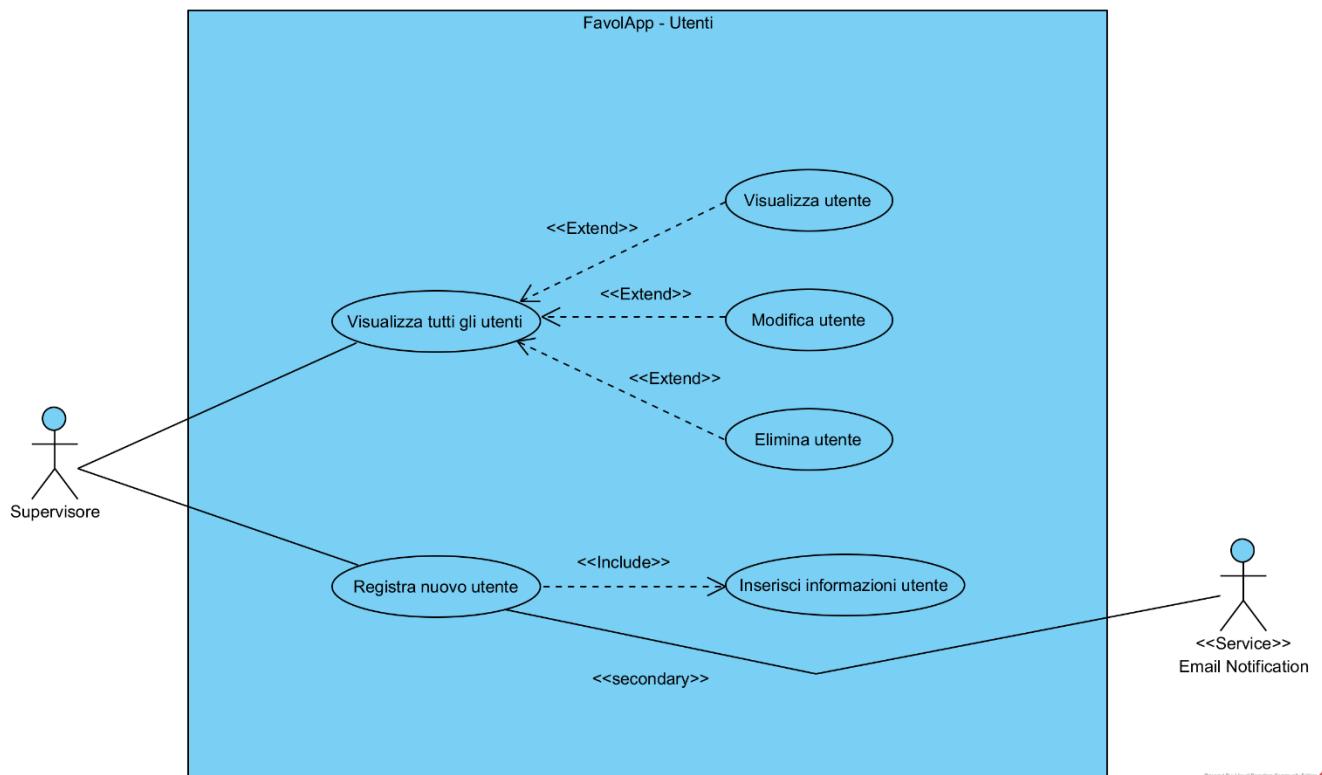


Figura 3.4: Casi d'uso inerenti alla pagina della gestione degli utenti.

Di seguito i casi d'uso in dettaglio che riguardano la sezione per la gestione degli utenti:

Nome	Visualizza tutti gli utenti
Obiettivo	Il Supervisore visualizza tutti gli utenti.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina degli utenti può visualizzare tutti gli utenti presenti sulla piattaforma.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato. Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	Verrà visualizzata la lista degli utenti registrati sulla piattaforma.

Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina degli utenti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti gli utenti in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.10: Dettagli caso d'uso per visualizzare la lista di tutti gli utenti.

Nome	Visualizza utente
Obiettivo	Il Supervisore visualizza i dettagli di un singolo utente.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina degli utenti può visualizzare i dettagli di un singolo utente.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	Verrà visualizzata la pagina del profilo dell'utente selezionato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina degli utenti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti gli utenti in una tabella. 3. Su un utente specifico clicca sul bottone "visualizza". 4. Verrà reindirizzato alla pagina del profilo dell'utente selezionato.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna.

Tabella 3.11: Dettagli caso d'uso per visualizzare i dettagli di un utente specifico.

Nome	Modifica utente
Obiettivo	Il Supervisore modifica un utente dalla lista.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina degli utenti può modificare un singolo utente.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	Verrà mostrato un form dove il Supervisore potrà modificare l'utente selezionato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina degli utenti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti gli utenti in una tabella. 3. Su un utente specifico clicca sul bottone "modifica". 4. Verrà visualizzato un form che permette la modifica dell'utente. 5. Conferma la modifica dei dati inseriti.
Estensioni o Flussi Alternativi	<p>Quando si clicca sul bottone modifica, il Supervisore può annullare la modifica tramite un bottone apposito.</p> <p>Se i dati inseriti sono scorretti, ad esempio il codice fiscale non coincide con i dati inseriti, allora verrà mostrato un messaggio di errore.</p>
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.12: Dettagli caso d'uso per modificare i dettagli di un utente specifico.

Nome	Elimina utente
Obiettivo	Il Supervisore vuole eliminare un utente.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina degli utenti può eliminare un utente specifico.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	L'utente selezionato verrà eliminato dalla piattaforma.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina degli utenti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti gli utenti in una tabella. 3. Su un utente specifico clicca sul bottone "elimina". 4. Verrà visualizzato un dialog dove si chiede la conferma di eliminazione, digitando la parola "Conferma".
Estensioni o Flussi Alternativi	Se non viene digitata la parola "Conferma" allora l'eliminazione dell'utente non può essere eseguita.
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.13: Dettagli caso d'uso per eliminare un utente specifico.

Nome	Registra nuovo utente
Obiettivo	Il Supervisore vuole registrare un nuovo utente.
Attori	Supervisore, Email Notification Service (secondario)
Descrizione	Il Supervisore tramite la pagina degli utenti può registrare un nuovo utente sulla piattaforma.

Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato. Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	L'utente selezionato verrà registrato sulla piattaforma.
Scenario principale	<ol style="list-style-type: none"> L'utente Supervisore accede alla pagina degli utenti tramite il menu di navigazione. Preme il bottone “Aggiungi utente”. Verrà mostrato un dialog dove il Supervisore inserirà tutti i dati dell'utente. Verranno mostrati i dati inseriti per verificare che siano stati inseriti correttamente. Il Supervisore preme il bottone “Registra” per avviare il processo di registrazione in backend.
Estensioni o Flussi Alternativi	<p>Se i dati inseriti sono scorretti, ad esempio il codice fiscale non coincide con i dati inseriti, allora verrà mostrato un messaggio di errore.</p> <p>Se l'email utilizzata per registrare l'utente è già presente nel database, verrà mostrato un messaggio di errore.</p> <p>Se il supervisore vuole annullare la registrazione può cliccare il tasto “Annulla”.</p>
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Dopo la registrazione di un nuovo utente, l'Email Notification Service provvederà ad inviare le credenziali d'accesso tramite l'email utilizzata in fase di registrazione.

Tabella 3.14: Dettagli caso d'uso per registrare un nuovo utente.

3.1.5 Casi d'uso – Pagina dei colleghi

Il diagramma dei casi d'uso illustrato di seguito riguarda la pagina dei colleghi. In questa sezione, sia il Tutor sia il Supervisore possono visualizzare l'elenco dei colleghi, ovvero gli utenti con cui condividono pazienti assegnati. Inoltre, il Supervisore ha la possibilità di modificare i dati di un collega, mentre entrambi, Supervisore e Tutor, possono accedere alla pagina del profilo di ciascun collega.

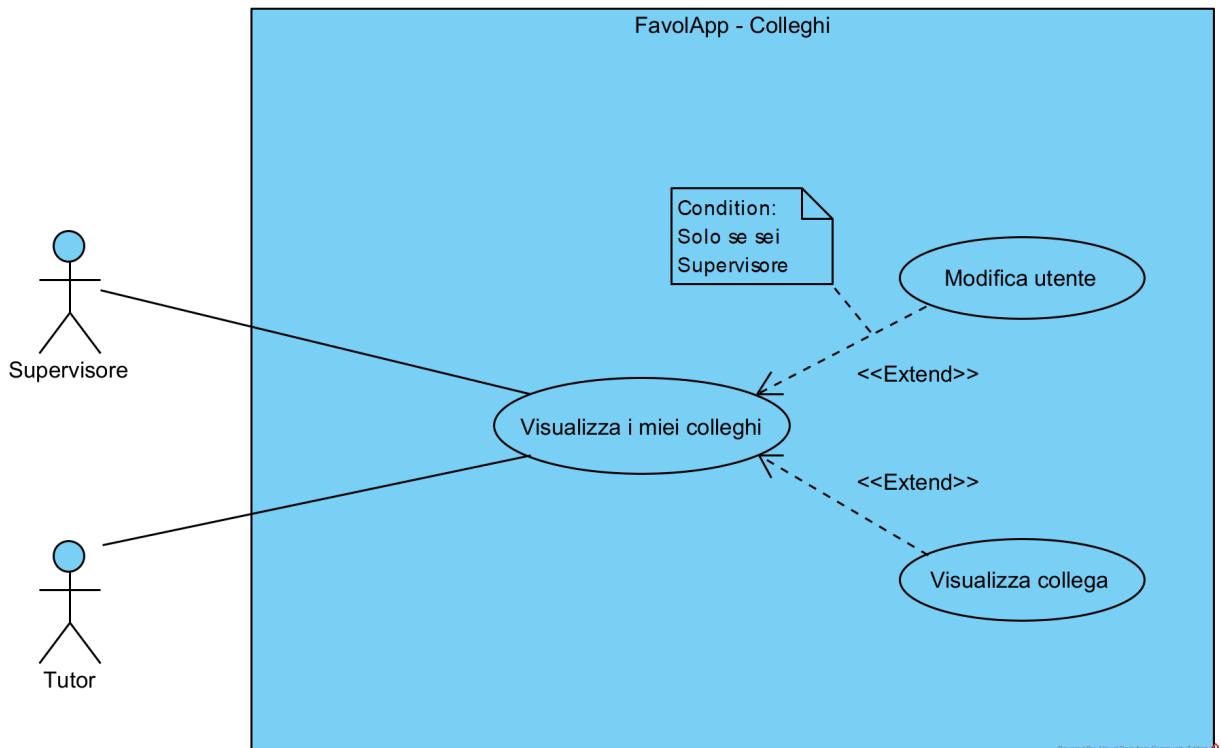


Figura 3.5: Casi d'uso inerenti alla pagina dei colleghi.

Di seguito i casi d'uso in dettaglio che riguardano la sezione per la gestione degli utenti:

Nome	Visualizza i miei colleghi
Obiettivo	Il Supervisore o il Tutor vogliono visualizzare i colleghi.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor, tramite la pagina dei colleghi visualizzate i loro colleghi, cioè gli utenti con cui condividono pazienti assegnati.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato.
Post-condizioni	Verrà mostrata una tabella con la lista dei propri colleghi.
Scenario principale	<ol style="list-style-type: none"> L'utente accede alla pagina dei colleghi tramite il menu di navigazione. Verrà mostrata la lista sei colleghi.
Estensioni o Flussi Alternativi	Nessuno

Requisiti speciali	Nessuno
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.15: Dettagli caso d'uso per visualizzare i colleghi.

Nome	Modifica utente
Obiettivo	Il Supervisore modifica un utente dalla lista.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina dei colleghi può modificare un singolo utente.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere il Supervisore.
Post-condizioni	Verrà mostrato un form dove il Supervisore potrà modificare l'utente selezionato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina dei colleghi tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i suoi colleghi. 3. Su un utente specifico clicca sul bottone "modifica". 4. Verrà visualizzato un form che permette la modifica dell'utente. 5. Conferma la modifica dei dati inseriti.
Estensioni o Flussi Alternativi	<p>Quando si clicca sul bottone modifica, il Supervisore può annullare la modifica tramite un bottone apposito.</p> <p>Se i dati inseriti sono scorretti, ad esempio il codice fiscale non coincide con i dati inseriti, allora verrà mostrato un messaggio di errore.</p>
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.16: Dettagli caso d'uso per modificare un collega.

Nome	Visualizza collega
Obiettivo	Il Supervisore o il Tutor visualizza i dettagli di un singolo collega.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore tramite la pagina dei colleghi può visualizzare i dettagli di un singolo utente.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato.
Post-condizioni	Verrà visualizzata la pagina del profilo dell'utente selezionato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei colleghi tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i colleghi in una tabella. 3. Su un utente specifico clicca sul bottone "visualizza". 4. Verrà reindirizzato alla pagina del profilo dell'utente selezionato.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.17: Dettagli caso d'uso per visualizzare un collega specifico.

3.1.6 Casi d'uso – Pagina dei report

Il diagramma dei casi d'uso illustrato di seguito descrive le funzionalità disponibili nella pagina dei report. Sia il Tutor sia il Supervisore possono visualizzare una lista di report, che include quelli scritti da loro stessi o dai loro colleghi. Il Supervisore, inoltre, ha accesso a una lista completa di tutti i report scritti da tutti gli utenti. Per ciascun report presente in lista, è possibile visualizzarne i dettagli e, se si è il proprietario del report, modificarlo. Durante la visualizzazione di un report, è inoltre possibile confermare la lettura dello stesso.

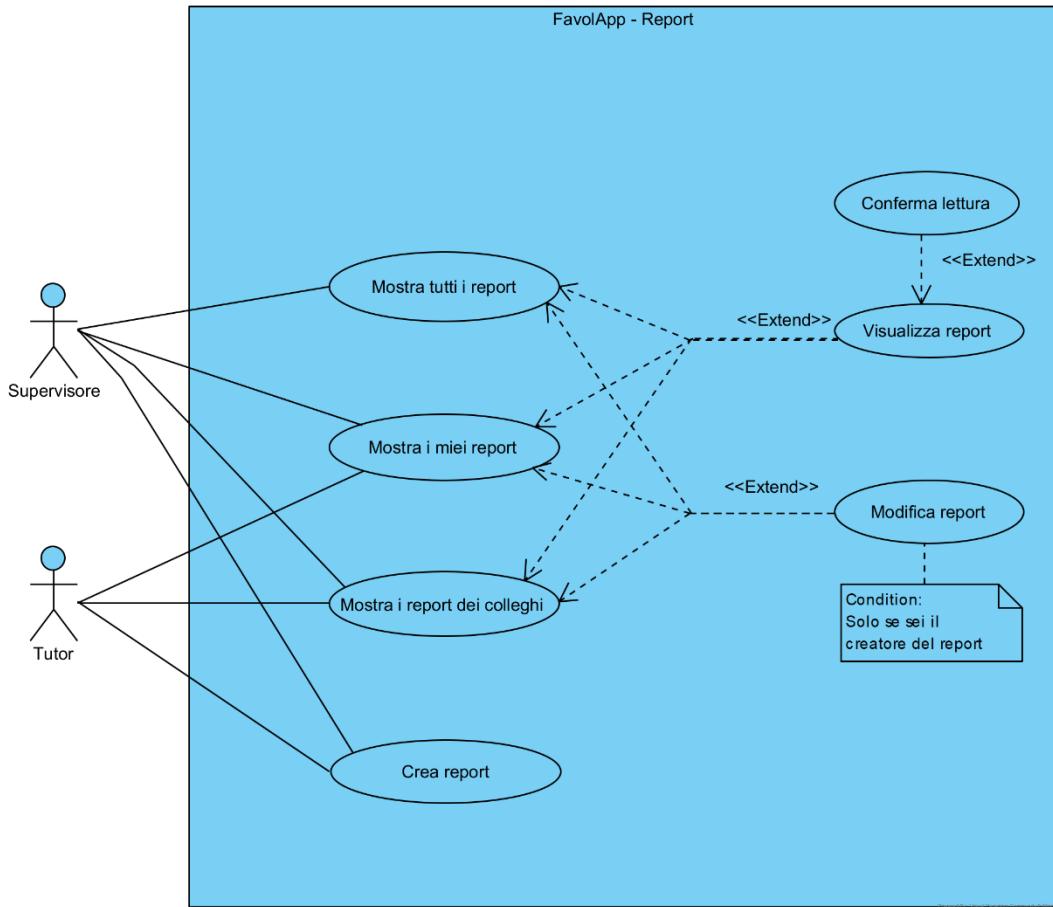


Figura 3.6: Casi d'uso inerenti alla pagina dei report.

Di seguito i casi d'uso in dettaglio che riguardano la sezione per la gestione dei report:

Nome	Mostra i miei report
Obiettivo	Il Supervisore o il Tutor visualizza i propri report.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può visualizzare una lista di tutti i report scritti da loro.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato.
Post-condizioni	Verrà visualizzata una lista con i report scritti dall'utente.
Scenario principale	<ol style="list-style-type: none"> L'utente accede alla pagina dei report tramite il menu di navigazione. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella.
Estensioni o Flussi Alternativi	Nessuna

Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.18: Dettagli caso d'uso per visualizzare la lista dei propri report.

Nome	Mostra i report dei colleghi
Obiettivo	Il Supervisore o il Tutor visualizza i report dei colleghi.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può visualizzare una lista dei report scritti dai loro colleghi.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato.
Post-condizioni	Verrà visualizzata una lista con i report scritti dai colleghi.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella. 3. Preme il bottone “Mostra i report dei colleghi”. 4. Verrà visualizzata una lista con tutti i report scritti dai colleghi.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.19: Dettagli caso d'uso per visualizzare la lista dei report dei colleghi.

Nome	Mostra tutti i report
Obiettivo	Il Supervisore visualizza i report di tutti gli utenti.
Attori	Supervisore

Descrizione	Il Supervisore tramite la pagina dei report può visualizzare una lista dei report scritti da tutti gli utenti della piattaforma.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • Il ruolo dell'utente deve essere Supervisore.
Post-condizioni	Verrà visualizzata una lista con i report scritti da tutti gli utenti.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella. 3. Preme il bottone “Mostra tutti i report”. 4. Verrà visualizzata una lista con tutti i report scritti dagli utenti della piattaforma.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.20: Dettagli caso d'uso per visualizzare la lista dei report di tutti gli utenti.

Nome	Visualizza report
Obiettivo	Il Supervisore o il Tutor visualizza un report.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può visualizzare i dettagli di un singolo report.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato.
Post-condizioni	Verranno visualizzati i dettagli di un report specifico.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella.

	<p>3. Preme sul bottone “Visualizza” in corrispondenza di un report specifico.</p> <p>4. Può visualizzare i dettagli del report selezionato.</p>
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.21: Dettagli caso d’uso per visualizzare i dettagli di un report.

Nome	Conferma lettura
Obiettivo	Il Supervisore o il Tutor conferma la lettura di un report.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può visualizzare i dettagli di un singolo report e confermarne la lettura.
Pre-condizioni	<ul style="list-style-type: none"> • L’utente deve essere autenticato.
Post-condizioni	Il report risulterà visualizzato dall’utente.
Scenario principale	<ol style="list-style-type: none"> 1. L’utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella. 3. Preme sul bottone “Visualizza” in corrispondenza di un report specifico. 4. Può confermare la lettura del report tramite un bottone “Conferma lettura”.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.22: Dettagli caso d’uso per rendere un report visualizzato.

Nome	Modifica report
Obiettivo	Il Supervisore o il Tutor modifica un report.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può modificare un report specifico.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve aver creato il report che vuole modificare.
Post-condizioni	Il report verrà modificato sul database e visualizzato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella. 3. Preme sul bottone “Modifica” in corrispondenza di un report specifico. 4. Tramite un dialog può modificare le informazioni del report.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente che vuole modificare il report deve essere il proprietario dello stesso.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.23: Dettagli caso d'uso per modificare un report.

Nome	Crea report
Obiettivo	Il Supervisore o il Tutor crea un report.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei report può creare un nuovo report.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato.

Post-condizioni	Il report verrà creato sul database e visualizzato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei report tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i report scritti da loro stessi in una tabella. 3. Preme sul bottone “Crea report”. 4. Tramite un dialog può inserire le informazioni del nuovo report. 5. Cliccando su “Crea report” nel dialog, procede alla creazione di un report
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna.

Tabella 3.24: Dettagli caso d'uso per creare un report.

3.1.7 Casi d'uso – Pagina dei pazienti

Il diagramma dei casi d'uso illustrato di seguito descrive le funzionalità disponibili nella pagina dei pazienti. Sia il Supervisore sia i Tutor possono accedere alla lista dei pazienti loro assegnati e consultare i dettagli relativi a ciascun paziente. Il Supervisore, inoltre, ha funzionalità aggiuntive: può aggiungere nuovi pazienti alla piattaforma e visualizzare l'elenco completo di tutti i pazienti presenti nel sistema.

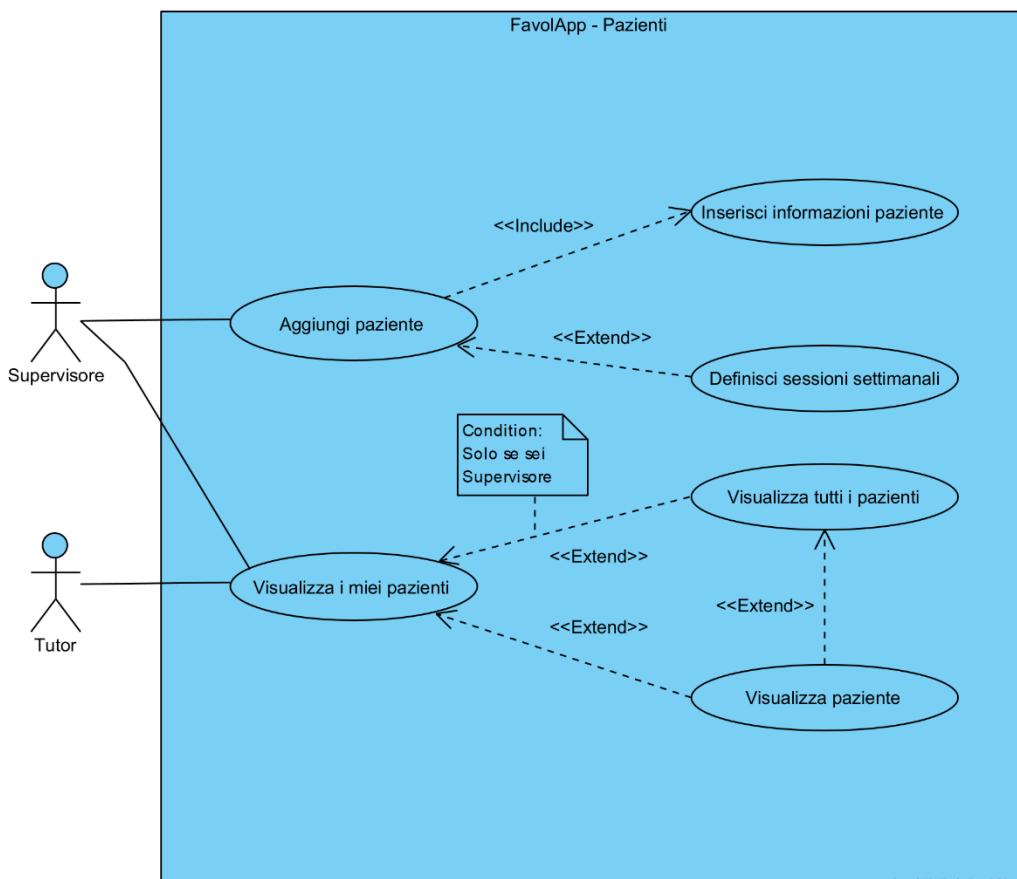


Figura 3.7: Casi d'uso inerenti alla pagina dei pazienti.

Di seguito i casi d'uso in dettaglio che riguardano la sezione per la gestione dei pazienti:

Nome	Visualizza i miei pazienti
Obiettivo	Il Supervisore o il Tutor visualizza i pazienti assegnati.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei pazienti possono visualizzare la lista dei pazienti a loro assegnati.
Pre-condizioni	<ul style="list-style-type: none"> L'utente deve essere autenticato.
Post-condizioni	Verrà mostrata una tabella con tutti i pazienti assegnati.
Scenario principale	<ol style="list-style-type: none"> L'utente accede alla pagina dei pazienti tramite il menu di navigazione. Viene mostrata la lista di tutti i pazienti a loro assegnati in una tabella.
Estensioni o Flussi Alternativi	Nessuna

Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.25: Dettagli caso d'uso per visualizzare i propri pazienti.

Nome	Visualizza paziente
Obiettivo	Il Supervisore o il Tutor visualizzano i dettagli di un paziente.
Attori	Supervisore, Tutor
Descrizione	Il Supervisore o il Tutor tramite la pagina dei pazienti possono visualizzare i dettagli di una paziente a loro assegnato.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato.
Post-condizioni	Verrà mostrata la scheda del paziente con tutte le sue informazioni.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina dei pazienti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i pazienti a loro assegnati in una tabella. 3. Preme sul pulsante “Visualizza” in corrispondenza del paziente da visualizzare. 4. Verrà mostrata la scheda del paziente.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	Nessuna
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.26: Dettagli caso d'uso per visualizzare i dettagli di un paziente.

Nome	Visualizza tutti i pazienti
Obiettivo	Il Supervisore visualizza la lista di tutti i pazienti.

Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina dei pazienti può visualizzare la lista completa di tutti i pazienti presenti nel sistema.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà mostrata la lista di tutti i pazienti presenti nel sistema.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina dei pazienti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i pazienti a lui assegnati in una tabella. 3. Preme sul pulsante “Visualizza tutti i pazienti”. 4. Verrà mostrata una lista di tutti i pazienti presenti nel sistema in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.27: Dettagli caso d'uso per visualizzare la lista di tutti i pazienti.

Nome	Aggiungi paziente
Obiettivo	Il Supervisore aggiunge un paziente nel sistema.
Attori	Supervisore,
Descrizione	Il Supervisore tramite la pagina dei pazienti può aggiungere un paziente al sistema inserendo le informazioni personali e definendo delle sessioni settimanali.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà aggiunto il paziente nel database e visualizzato.

Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina dei pazienti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i pazienti a lui assegnati in una tabella. 3. Preme sul pulsante “Aggiungi paziente”. 4. Verrà mostrato un dialog dove inserire le informazioni personali del paziente e le sessioni settimanali. 5. Si farà una revisione dei dati inseriti e delle sessioni definite e cliccando su “Aggiungi” il paziente verrà aggiunto al sistema.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.28: Dettagli caso d'uso per aggiungere un nuovo paziente.

Nome	Modifica paziente
Obiettivo	Il Supervisore modifica un paziente.
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina che mostra i dettagli del paziente può effettuare la modifica delle informazioni.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà modificato il paziente nel database e visualizzato.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina dei pazienti tramite il menu di navigazione. 2. Viene mostrata la lista di tutti i pazienti a lui assegnati in una tabella. 3. Preme sul pulsante “Visualizza paziente” su un paziente specifico. 4. Verrà mostrata una pagina con i dettagli del paziente.

	5. Clicca su “Modifica” e verrà mostrato un dialog per la modifica del paziente.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L’utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Nessuna

Tabella 3.29: Dettagli caso d’uso per modificare un paziente.

3.1.8 Casi d’uso – Pagina delle attività

Il diagramma dei casi d’uso illustrato di seguito descrive le funzionalità disponibili nella pagina delle attività, accessibile esclusivamente al Supervisore. In questa sezione, il Supervisore può visualizzare i log relativi a diverse tipologie di attività, come quelle di autenticazione, operazioni effettuate dagli utenti, attività legate ai pazienti e ai report. Inoltre, selezionando una specifica attività, è possibile consultare i dettagli completi ad essa associati.

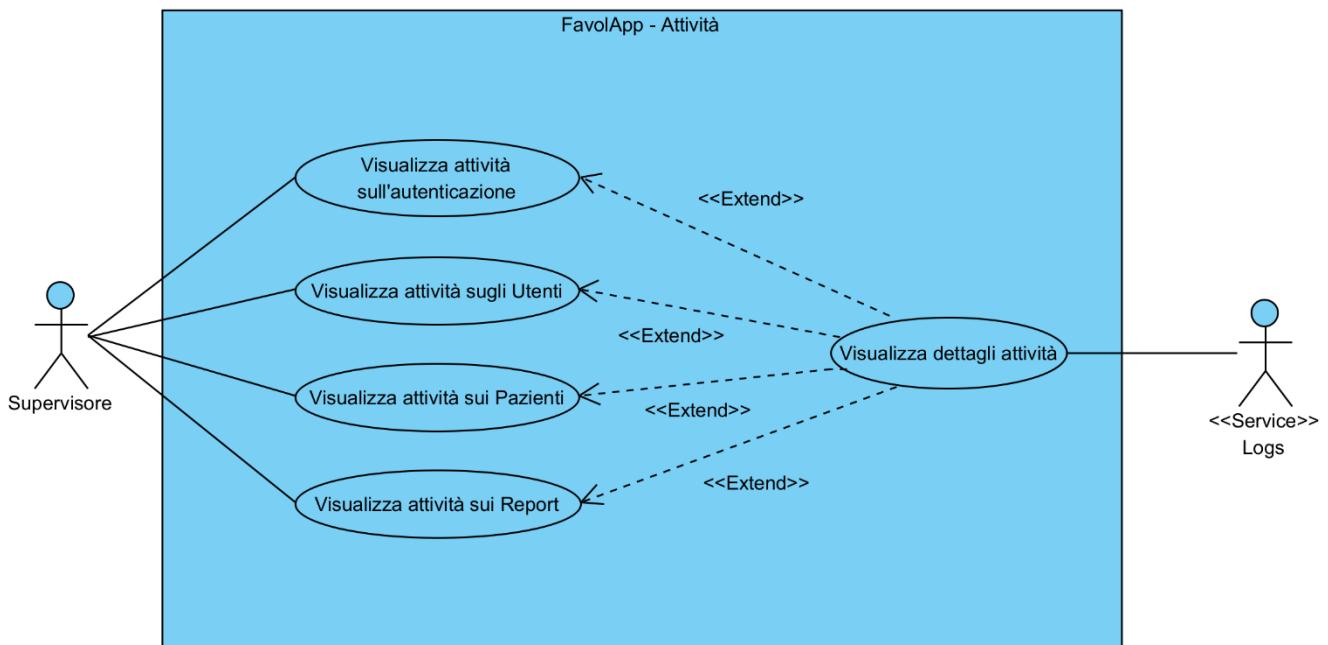


Figura 3.8: Casi d’uso inerenti alla pagina delle attività.

Di seguito i casi d’uso in dettaglio che riguardano la sezione per la pagina delle attività:

Nome	Visualizza attività sull'autenticazione
Obiettivo	Il Supervisore visualizza le attività sull'autenticazione
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina delle attività può visualizzare una lista di tutte le attività riguardo l'autenticazione.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà visualizzata una lista di attività legate all'autenticazione in una tabella.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina delle attività tramite il menu di navigazione. 2. Preme sul pulsante del tab "Auth". 3. Verrà mostrata la lista di tutte le attività legate all'autenticazione in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Un servizio di logs fornisce i log delle attività.

Tabella 3.30: Dettagli caso d'uso per visualizzare le attività sull'autenticazione.

Nome	Visualizza attività sugli utenti
Obiettivo	Il Supervisore visualizza le attività sugli utenti
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina delle attività può visualizzare una lista di tutte le attività riguardo gli utenti.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà visualizzata una lista di attività legate agli utenti in una tabella.

Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina delle attività tramite il menu di navigazione. 2. Preme sul pulsante del tab “Utenti”. 3. Verrà mostrata la lista di tutte le attività legate agli utenti in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Un servizio di logs fornisce i log delle attività.

Tabella 3.31: Dettagli caso d'uso per visualizzare le attività sugli utenti.

Nome	Visualizza attività sui pazienti
Obiettivo	Il Supervisore visualizza le attività sui pazienti
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina delle attività può visualizzare una lista di tutte le attività riguardo i pazienti.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà visualizzata una lista di attività legate ai pazienti in una tabella.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina delle attività tramite il menu di navigazione. 2. Preme sul pulsante del tab “Pazienti”. 3. Verrà mostrata la lista di tutte le attività legate ai pazienti in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Un servizio di logs fornisce i log delle attività.

Tabella 3.32: Dettagli caso d'uso per visualizzare le attività sui pazienti.

Nome	Visualizza attività sui report
Obiettivo	Il Supervisore visualizza le attività sui report
Attori	Supervisore
Descrizione	Il Supervisore tramite la pagina delle attività può visualizzare una lista di tutte le attività riguardo i report.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verrà visualizzata una lista di attività legate ai report in una tabella.
Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina delle attività tramite il menu di navigazione. 2. Preme sul pulsante del tab “Report”. 3. Verrà mostrata la lista di tutte le attività legate ai report in una tabella.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Un servizio di logs fornisce i log delle attività.

Tabella 3.33: Dettagli caso d'uso per visualizzare le attività sui pazienti.

Nome	Visualizza dettagli attività
Obiettivo	Il Supervisore visualizza i dettagli di un'attività
Attori	Supervisore, AWS AppSync (secondario)
Descrizione	Il Supervisore tramite la pagina delle attività può visualizzare i dettagli inerenti ad una specifica attività.
Pre-condizioni	<ul style="list-style-type: none"> • L'utente deve essere autenticato. • L'utente deve avere il ruolo di Supervisore.
Post-condizioni	Verranno visualizzati i dettagli inerenti all'attività selezionata.

Scenario principale	<ol style="list-style-type: none"> 1. L'utente Supervisore accede alla pagina delle attività tramite il menu di navigazione. 2. Preme su un tab per visualizzare le attività. 3. Verrà mostrata la lista di tutte le attività legate al tab selezionato in una tabella. 4. Preme su un'attività specifica e vengono mostrati tutti i dettagli su quell'attività.
Estensioni o Flussi Alternativi	Nessuna
Requisiti speciali	L'utente deve essere un Supervisore.
Frequenza di utilizzo	Media
Note Aggiuntive	Un servizio di logs fornisce i log delle attività.

Tabella 3.34: Dettagli caso d'uso per visualizzare i dettagli di un'attività.

3.2 System Sequence Diagram

Di Un System Sequence Diagram è uno strumento utilizzato per rappresentare le interazioni tra gli attori e il sistema in termini di scambio di messaggi. Gli attori, solitamente elementi esterni come utenti o altri sistemi, inviano messaggi al sistema, che risponde elaborando e restituendo informazioni.

I messaggi rappresentano le informazioni scambiate tra due entità. Questi possono essere:

- Sincroni, quando l'emittente attende una risposta prima di proseguire;
- Asincroni, quando l'emittente non attende immediatamente la risposta, che può arrivare successivamente.

Un messaggio di risposta viene generato come reazione a un messaggio ricevuto e contiene informazioni correlate al messaggio iniziale. Inoltre, è possibile che un'entità invii un messaggio a sé stessa, evento definito come messaggio ricorsivo.

Il System Sequence Diagram è particolarmente utile per visualizzare il flusso di interazioni a livello di sistema, evidenziando come il sistema risponda agli input ricevuti dagli attori in modo chiaro e strutturato.

3.2.1 System Sequence Diagram – Autenticazione

Il System Sequence Diagram riportato di seguito illustra l'interazione tra un utente non autenticato e il sistema durante il processo di autenticazione. Il diagramma include anche le possibili alternative che si verificano in caso di errori nella risposta del sistema.

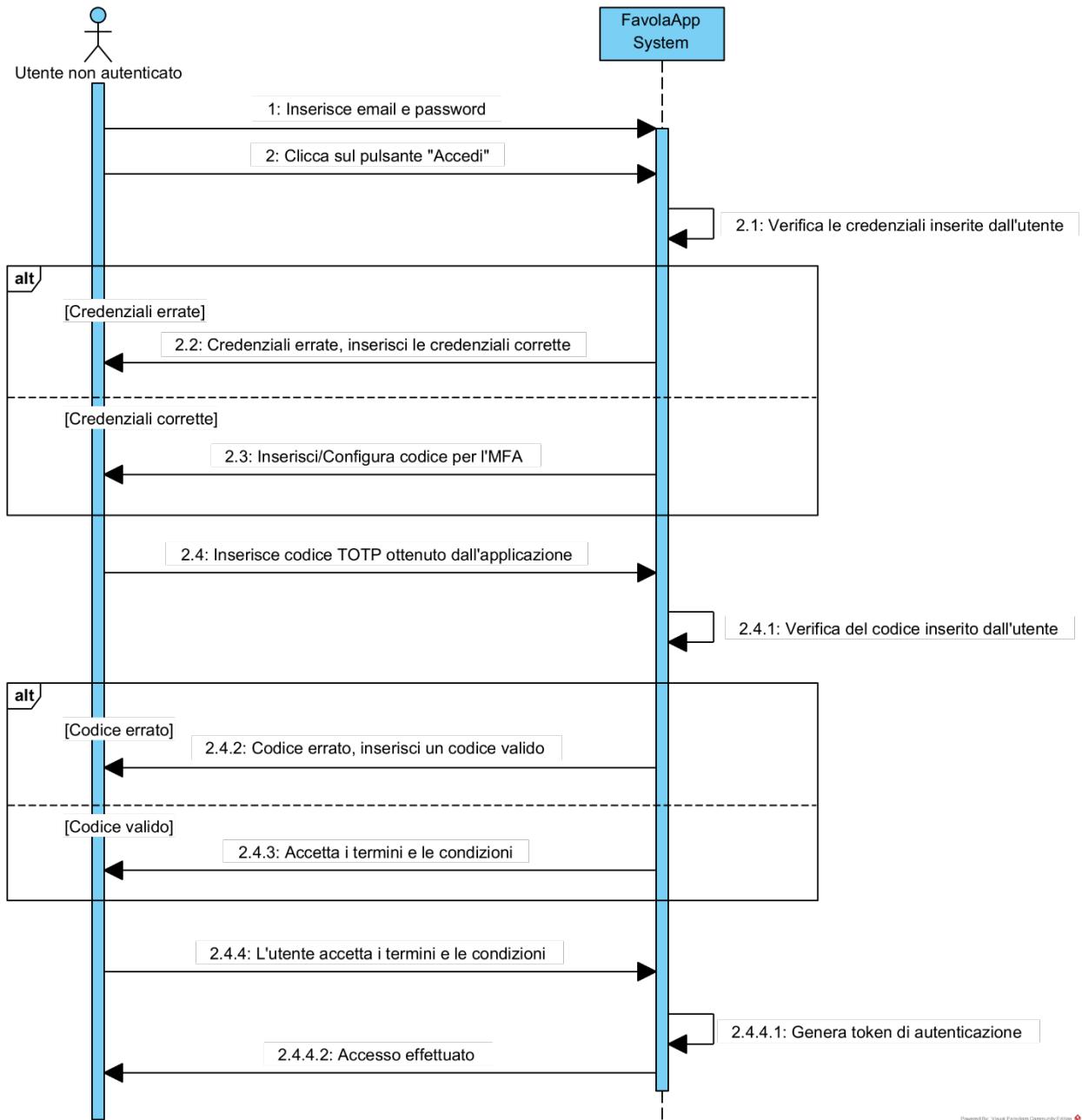


Figura 3.9: System Sequence Diagram della fase di autenticazione.

3.2.2 System Sequence Diagram – Pagina del profilo personale

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina del profilo personale di un utente. I diagrammi illustrano le interazioni principali tra l'utente e il sistema, evidenziando le operazioni disponibili, come la visualizzazione delle informazioni personali, il caricamento di un avatar o la modifica dei dati personali (se consentito). Ogni diagramma fornisce una rappresentazione dettagliata del flusso di messaggi scambiati, offrendo una chiara comprensione delle funzionalità offerte e delle condizioni in cui tali operazioni vengono eseguite.

3.2.2.1 Modifica informazioni personali

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema durante il processo di modifica delle informazioni personali.

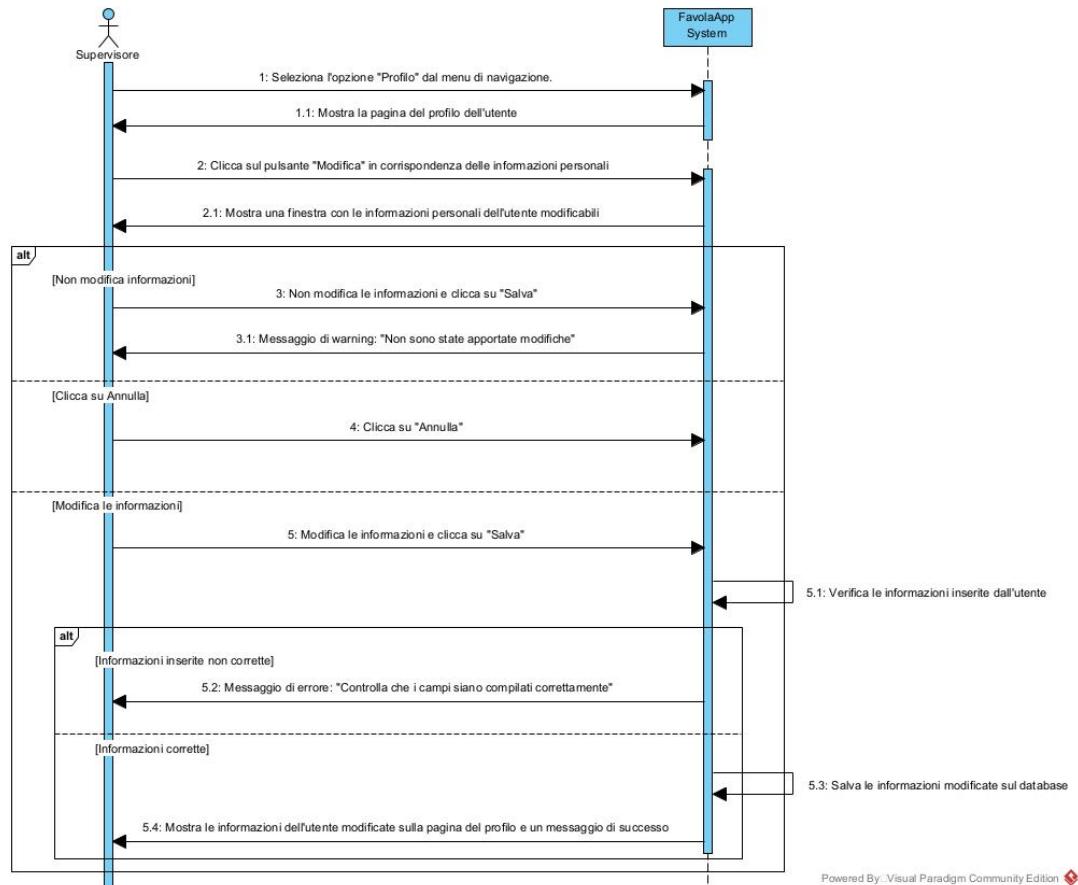


Figura 3.10: System Sequence Diagram sulla modifica delle informazioni personali.

3.2.2.2 Visualizza informazioni personali

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per visualizzare le informazioni personali nella pagina del profilo personale.

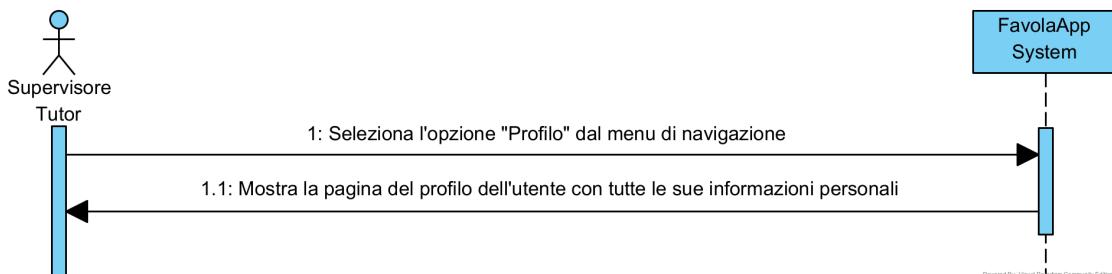


Figura 3.11: System Sequence Diagram per visualizzare le informazioni personali.

3.2.2.3 Carica avatar

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per caricare un avatar nella pagina del profilo personale.

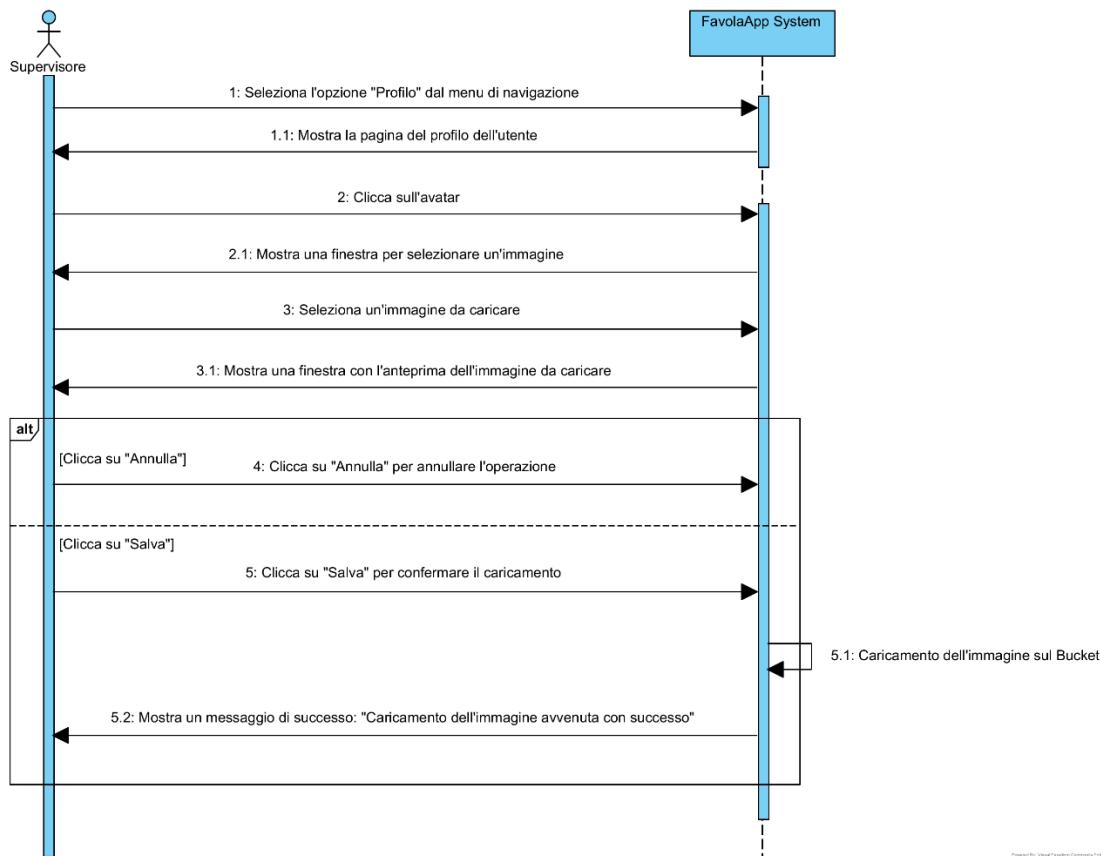


Figura 3.12: System Sequence Diagram per caricare l'avatar.

3.2.3 System Sequence Diagram – Pagina del profilo di un altro utente

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina del profilo di un altro utente. I diagrammi illustrano le principali interazioni tra l'utente autenticato e il sistema, mettendo in evidenza le operazioni disponibili, come la visualizzazione delle informazioni del profilo dell'altro utente, l'elenco dei pazienti in comune e i report condivisi. Ogni diagramma offre una rappresentazione dettagliata del flusso di messaggi scambiati, evidenziando le funzionalità offerte e le condizioni necessarie per eseguire ciascuna operazione.

3.2.3.1 Modifica informazioni utente

Il System Sequence Diagram che descrive l'interazione tra un Supervisore e il sistema per la modifica delle informazioni di un altro utente è identico a quello riportato nel *paragrafo 4.4.2.1*.

3.2.3.2 Mostra pazienti in comune

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare la lista dei pazienti in comune con l'utente visualizzato.

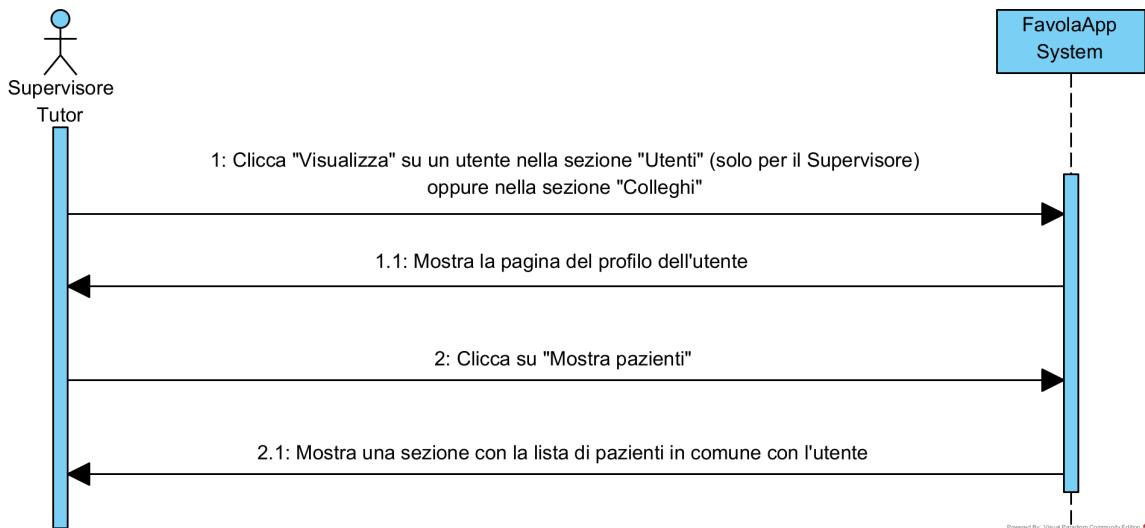


Figura 3.13: System Sequence Diagram per mostrare i pazienti in comune con l'altro utente.

3.2.3.3 Mostra tutti i suoi pazienti

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per mostrare la lista di tutti i pazienti dell'utente visualizzato.

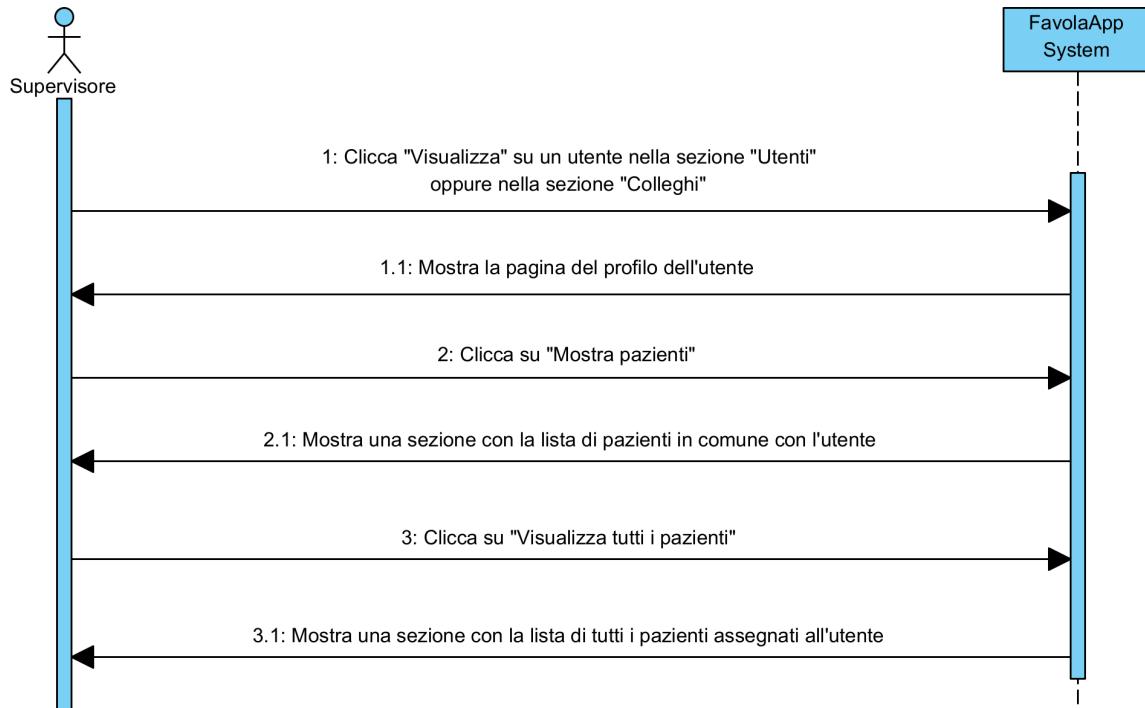


Figura 3.14: System Sequence Diagram per mostrare tutti i pazienti assegnati all'altro utente.

3.2.3.4 Visualizza i suoi report

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare la lista dei report realizzati da un altro utente tramite la sua pagina del profilo.

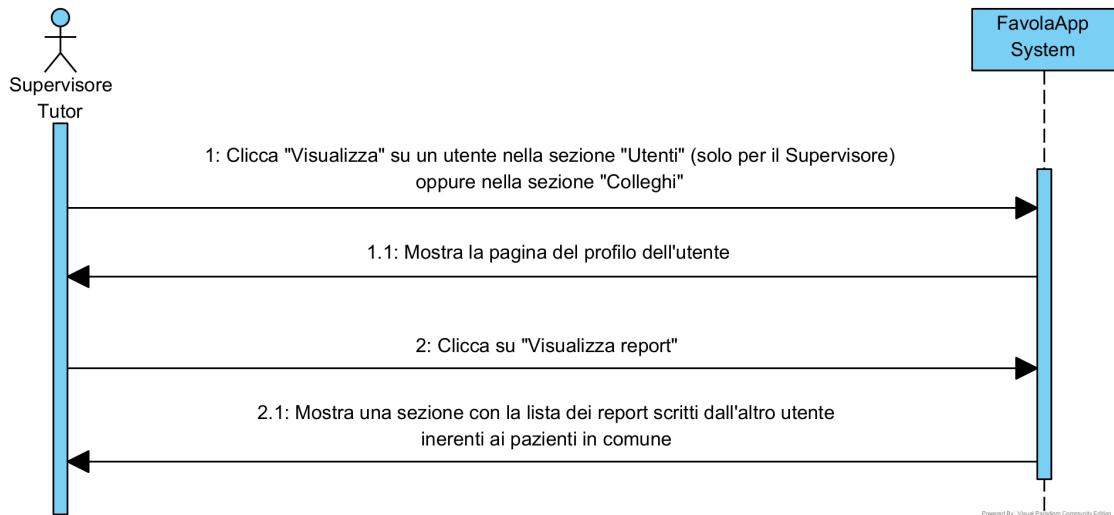


Figura 3.15: System Sequence Diagram per visualizzare i report realizzati dall'altro utente.

3.2.3.5 Visualizza informazioni utente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare le informazioni di un altro utente presenti sulla sua pagina del profilo.

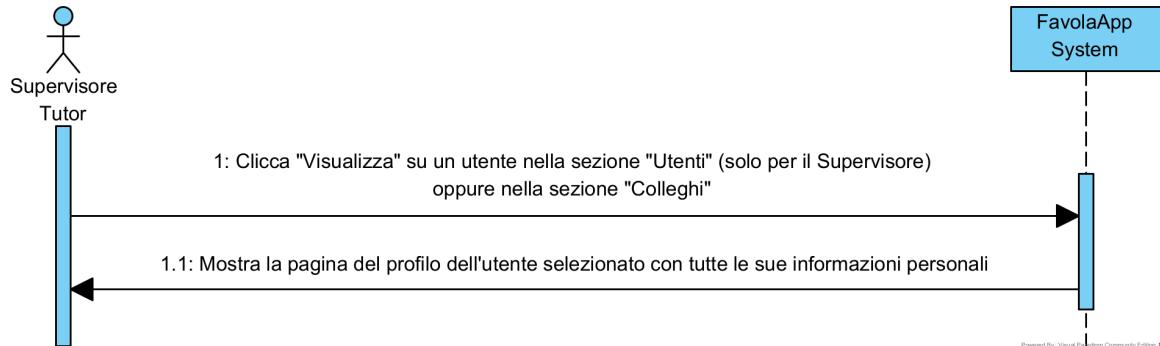


Figura 3.16: System Sequence Diagram per visualizzare le informazioni di un altro utente.

3.2.4 System Sequence Diagram – Pagina per la gestione degli utenti

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina degli utenti. I diagrammi illustrano le principali interazioni tra il Supervisore e il sistema, evidenziando le operazioni disponibili, come la visualizzazione dell'elenco degli utenti, la consultazione dei dettagli di un singolo utente, la modifica delle informazioni associate o l'eliminazione di un utente. Inoltre, viene rappresentata anche l'operazione di registrazione di un nuovo utente. Ogni diagramma descrive in dettaglio il flusso di messaggi scambiati, fornendo una chiara comprensione delle funzionalità offerte e delle condizioni necessarie per eseguire le varie operazioni.

3.2.4.1 Visualizza e gestisci gli utenti

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per mostrare la lista degli utenti sulla piattaforma e per le operazioni di visualizzazione, modifica ed eliminazione di un utente specifico della lista.

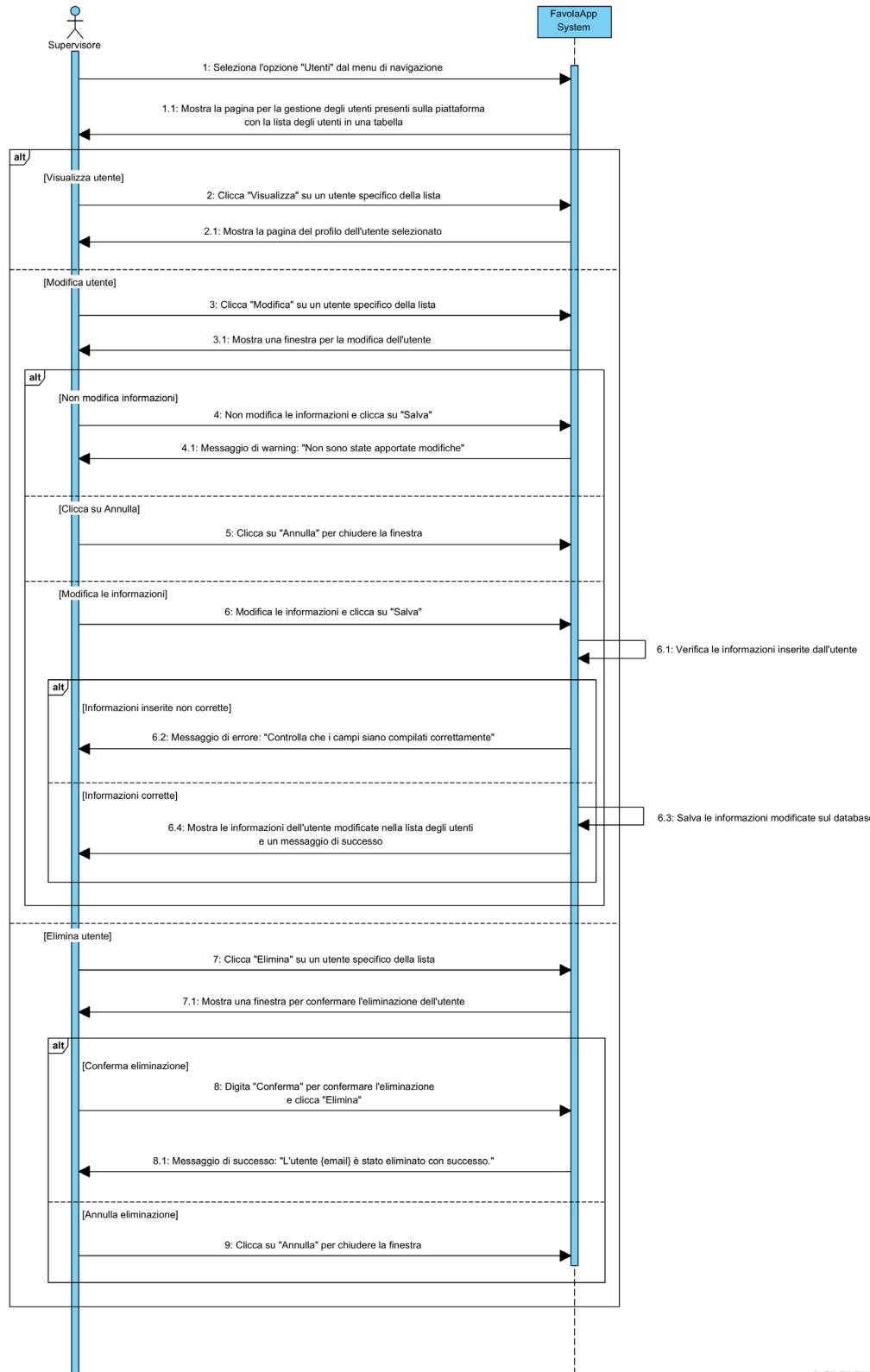


Figura 3.17: System Sequence Diagram per visualizzare la lista degli utenti e visualizzare, modificare ed eliminare un utente specifico.

3.2.4.2 Registra nuovo utente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per registrare un nuovo utente sulla piattaforma.

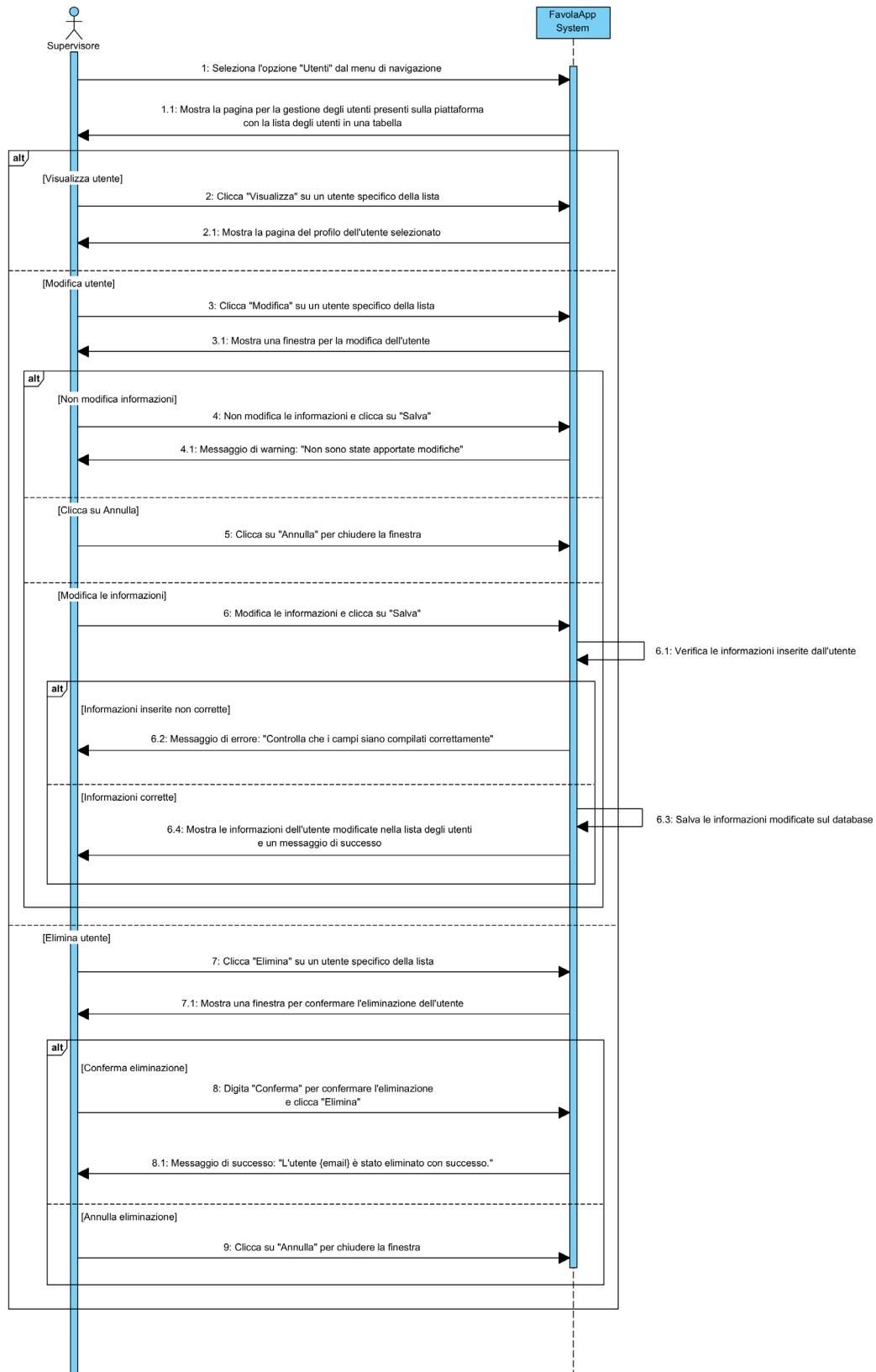


Figura 3.18: System Sequence Diagram per registrare un nuovo utente sulla piattaforma.

3.2.5 System Sequence Diagram – Pagina dei colleghi

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina dei colleghi. I diagrammi illustrano le principali interazioni tra il Tutor o il Supervisore e il sistema, evidenziando le operazioni disponibili, come la visualizzazione dell'elenco dei colleghi, ovvero gli utenti con pazienti in comune, e l'accesso al profilo di un collega. Inoltre, viene rappresentata l'operazione esclusiva del Supervisore di modifica delle informazioni di un collega. Ogni diagramma fornisce una descrizione dettagliata del flusso di messaggi scambiati, offrendo una chiara comprensione delle funzionalità disponibili e delle condizioni necessarie per svolgere le diverse operazioni.

3.2.5.1 Visualizza i miei colleghi

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per visualizzare la lista dei colleghi.

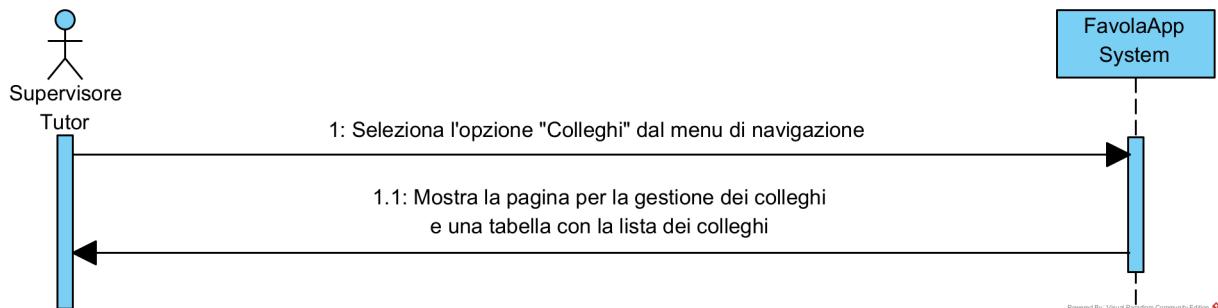


Figura 3.19: System Sequence Diagram per visualizzare i miei colleghi.

3.2.5.2 Visualizza collega

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per visualizzare le informazioni di un utente specifico presente nella lista dei colleghi.

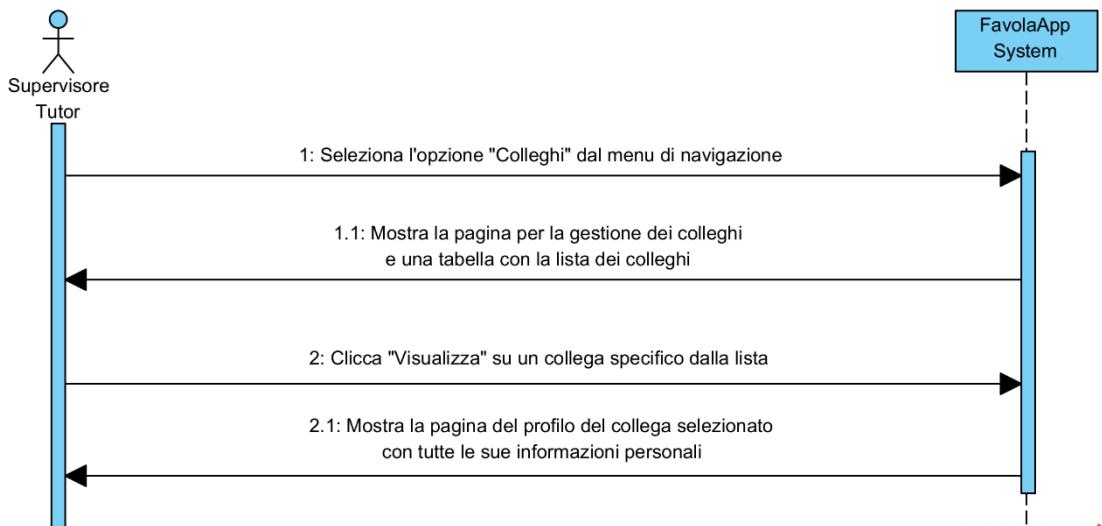


Figura 3.20: System Sequence Diagram per visualizzare le informazioni di un collega.

3.2.5.3 Modifica utente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per modificare le informazioni di un utente specifico presente nella lista dei colleghi.

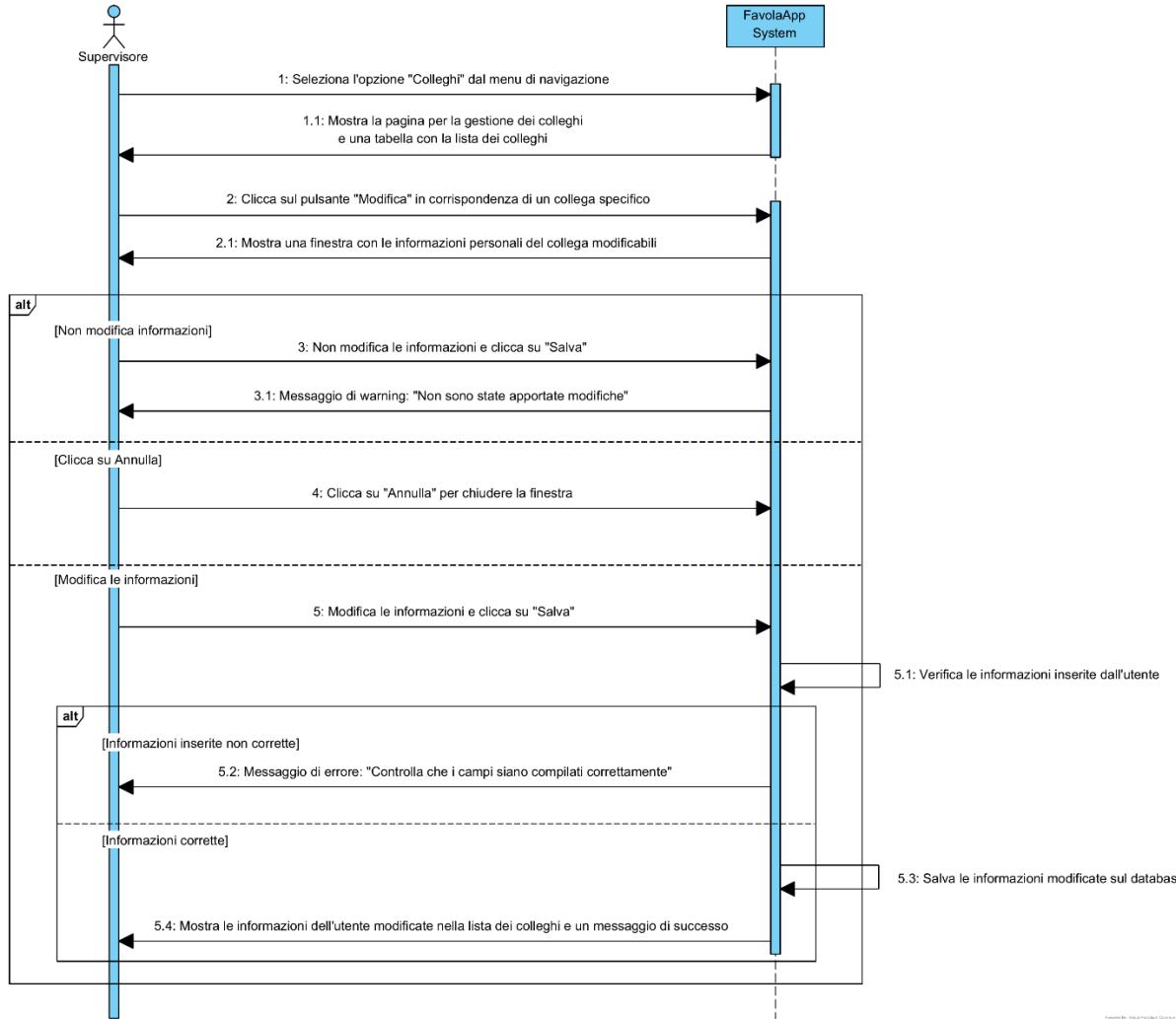


Figura 3.21: System Sequence Diagram per modificare le informazioni di un collega.

3.2.6 System Sequence Diagram – Pagina dei report

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina dei report. I diagrammi descrivono le principali interazioni tra il Tutor, il Supervisore e il sistema, evidenziando le operazioni disponibili, come la visualizzazione dell'elenco dei report scritti da sé stessi o dai colleghi e l'accesso ai dettagli di un singolo report. Inoltre, viene rappresentata la possibilità esclusiva del Supervisore di visualizzare tutti i report presenti nella piattaforma. Ogni diagramma offre una rappresentazione dettagliata del flusso di messaggi scambiati, mettendo in evidenza le funzionalità offerte e le condizioni necessarie per eseguire le varie operazioni.

3.2.6.1 Mostra i miei report

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare la lista dei report sui pazienti.

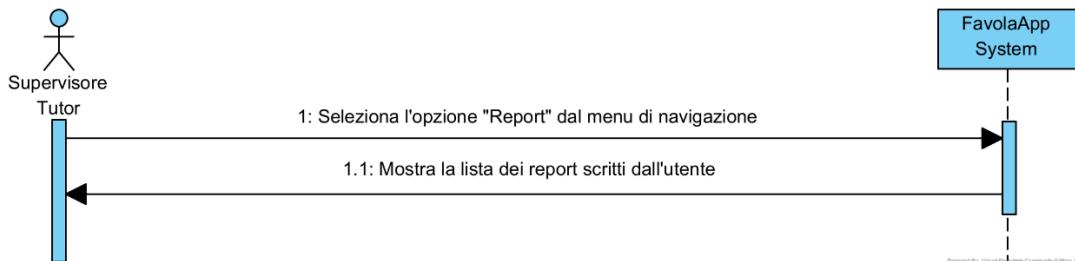


Figura 3.22: System Sequence Diagram per mostrare i report dell'utente.

3.2.6.2 Mostra i report dei colleghi

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare i report sui pazienti realizzati dai colleghi.

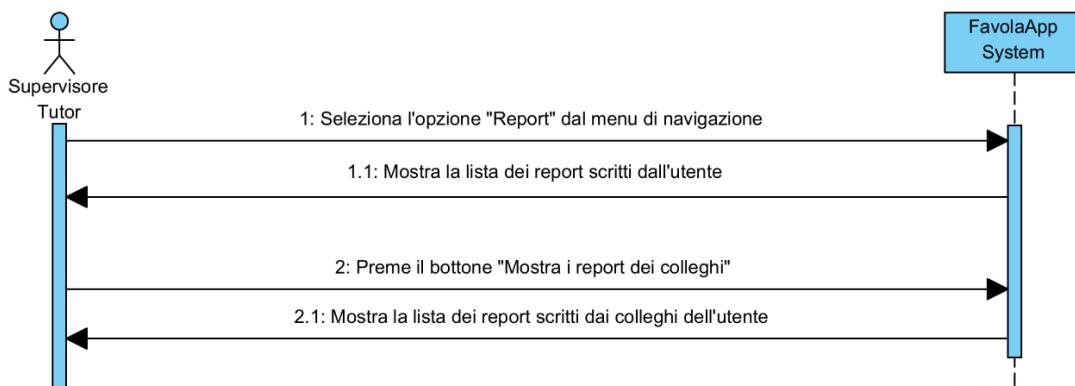


Figura 3.23: System Sequence Diagram per mostrare i report realizzati dai colleghi.

3.2.6.3 Mostra tutti i report

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per mostrare tutti i report scritti dagli utenti.

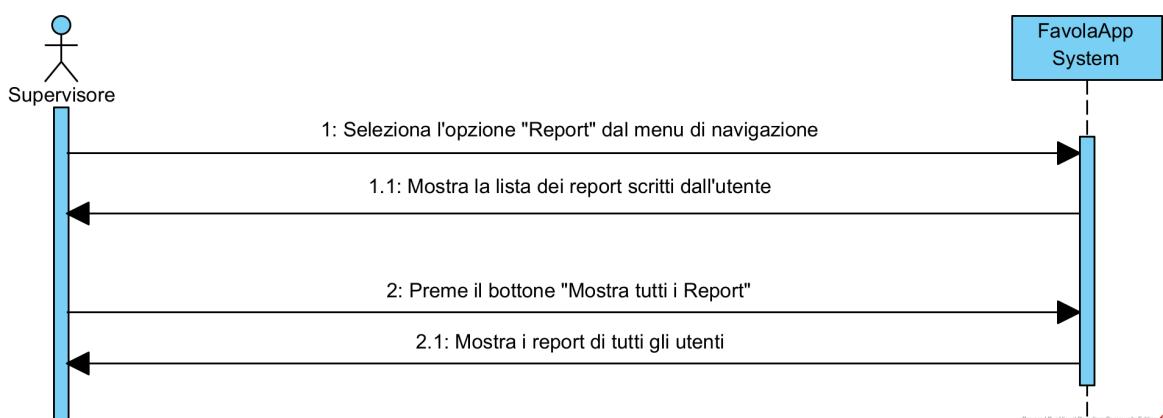


Figura 3.24: System Sequence Diagram per mostrare tutti i report scritti dagli utenti.

3.2.6.4 Operazioni sui report

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per mostrare le operazioni da poter eseguire su un report specifico.

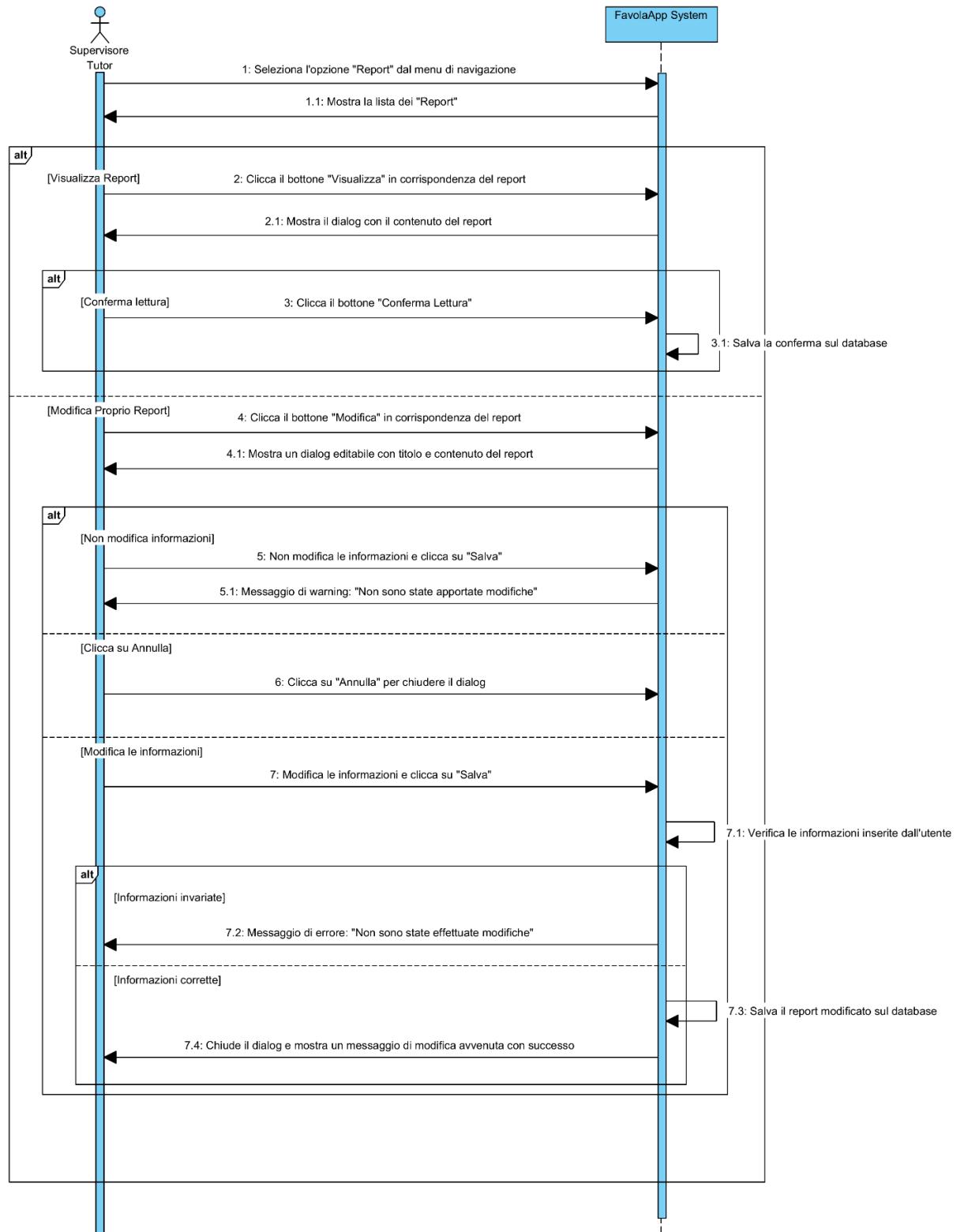


Figura 3.25: System Sequence Diagram per le operazioni su un report specifico.

3.2.6.5 Crea report

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per creare un nuovo report.

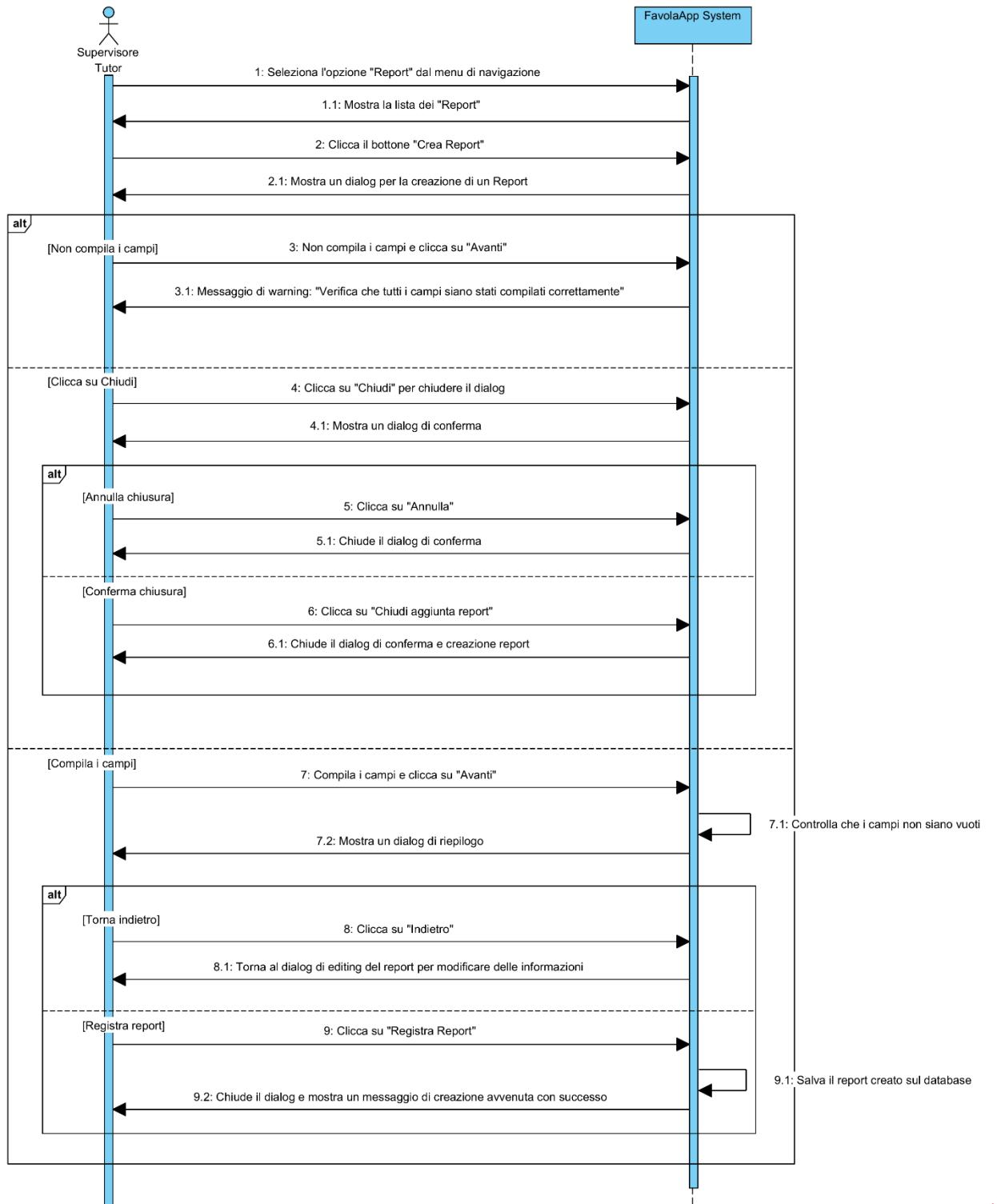


Figura 3.26: System Sequence Diagram per creare un nuovo report.

3.2.7 System Sequence Diagram – Pagina dei pazienti

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina dei pazienti. I diagrammi illustrano le interazioni principali tra l'utente e il sistema, evidenziando le operazioni disponibili, come l'aggiunta di un nuovo paziente, la modifica delle informazioni anagrafiche e la visualizzazione dei dettagli relativi a ciascun paziente. Ogni diagramma fornisce una rappresentazione dettagliata del flusso di messaggi scambiati, offrendo una chiara comprensione delle funzionalità disponibili e delle condizioni necessarie per l'esecuzione di tali operazioni.

3.2.7.1 Visualizza i miei pazienti

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per visualizzare la lista dei pazienti ad esso assegnati.

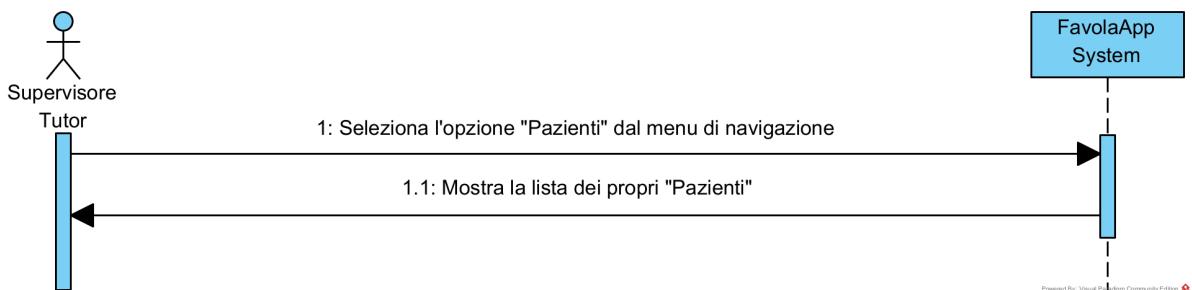


Figura 3.27: System Sequence Diagram per visualizzare i pazienti dell'utente.

3.2.7.2 Visualizza paziente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un utente autenticato e il sistema per visualizzare i dettagli di un paziente specifico.

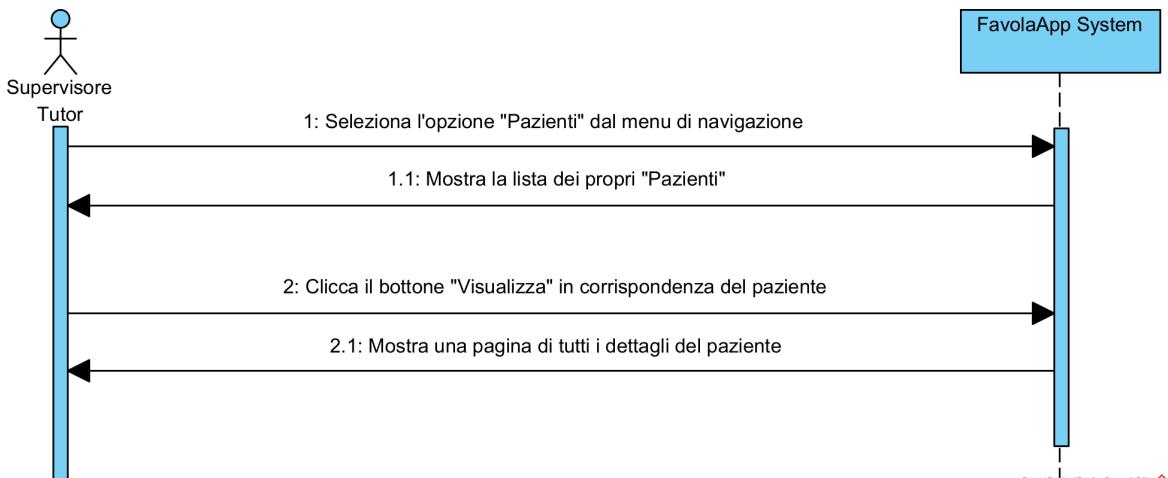


Figura 3.28: System Sequence Diagram per visualizzare i dettagli di un paziente.

3.2.7.3 Aggiungi paziente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per aggiungere un paziente.

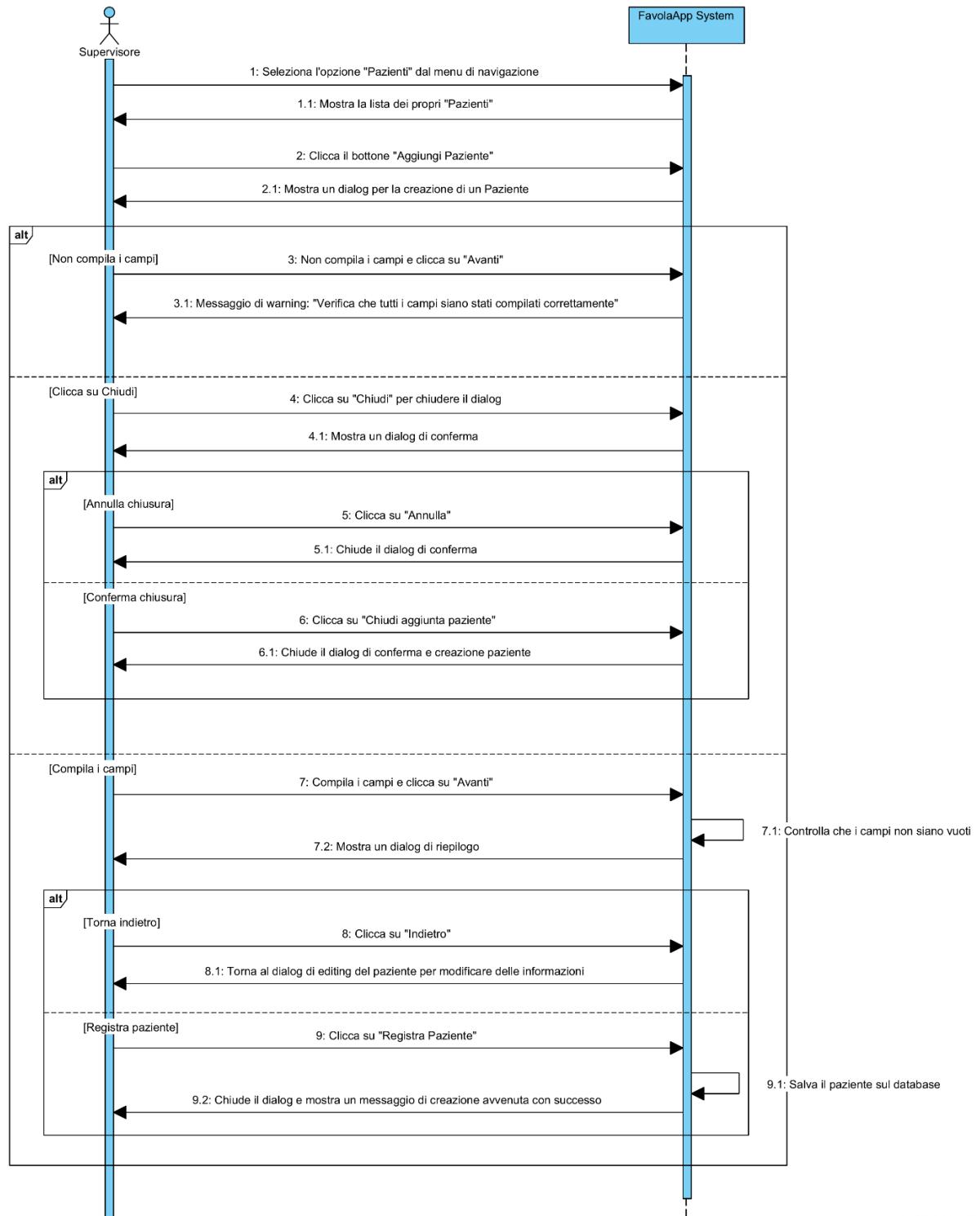


Figura 3.29: System Sequence Diagram per aggiungere un paziente nel sistema.

3.2.7.4 Modifica paziente

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per modificare un paziente.

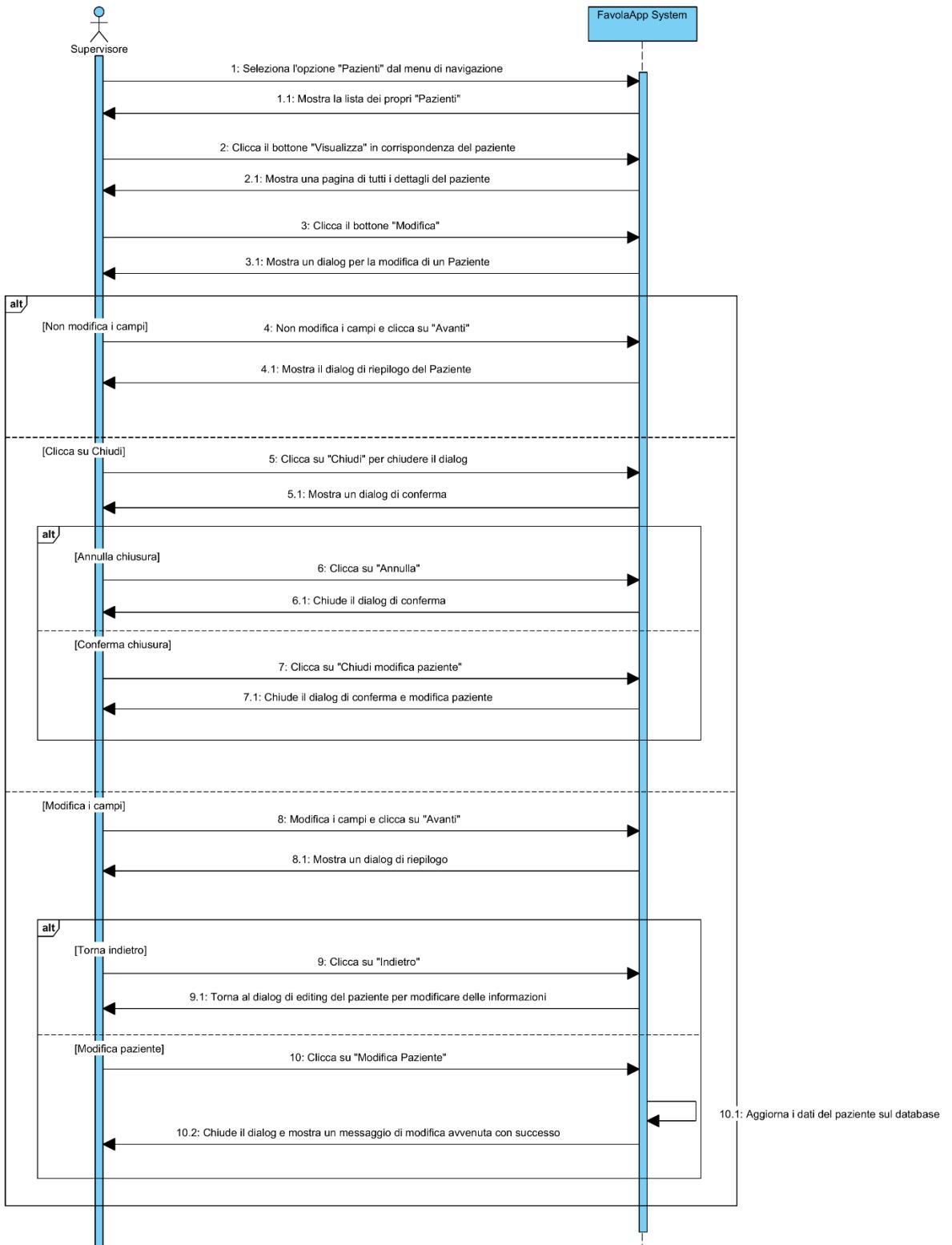


Figura 3.30: System Sequence Diagram per modificare un paziente.

3.2.7.5 Visualizza tutti i pazienti

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per visualizzare tutti i pazienti presenti nel sistema.

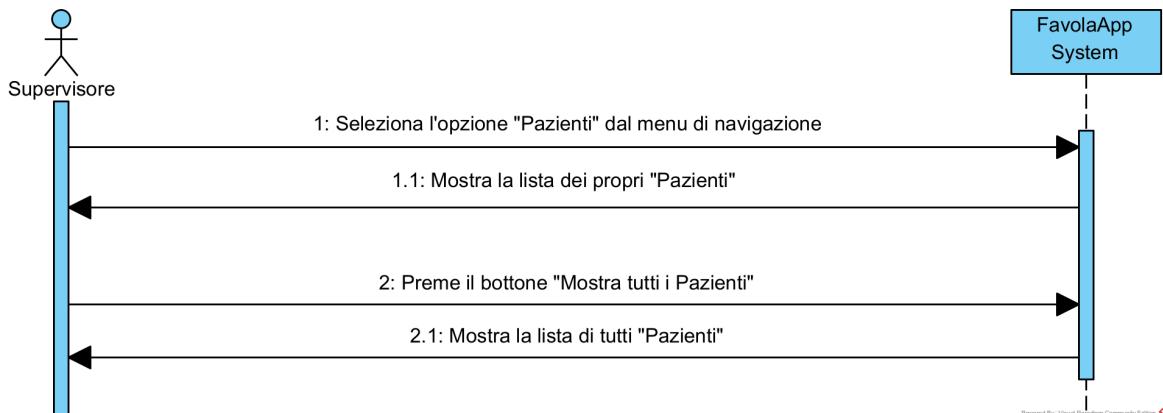


Figura 3.31: System Sequence Diagram per visualizzare tutti i pazienti presenti nel sistema.

3.2.8 System Sequence Diagram – Pagina dei report

Questa sezione presenta tutti i System Sequence Diagram relativi alla pagina delle attività. I diagrammi illustrano le interazioni principali tra un Supervisore e il sistema, evidenziando le operazioni disponibili, come la selezione del tipo di attività e visualizzare i dettagli di un'attività. Ogni diagramma offre una rappresentazione dettagliata del flusso di messaggi scambiati, fornendo una chiara comprensione delle funzionalità offerte e delle condizioni in cui tali operazioni possono essere eseguite.

3.2.8.1 Visualizza dettagli attività

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per visualizzare i dettagli di un'attività specifica.

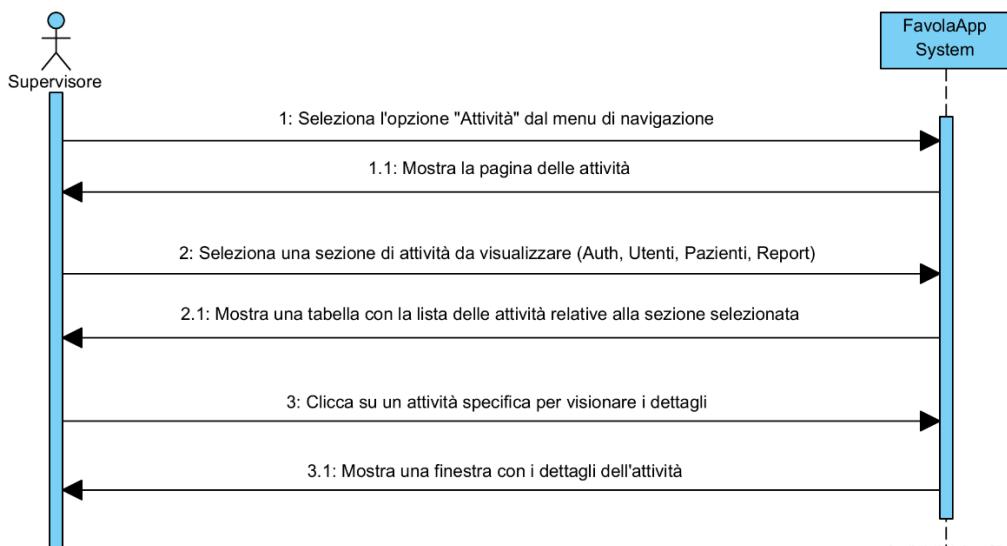


Figura 3.32: System Sequence Diagram per visualizzare i dettagli di un'attività.

3.2.8.2 Visualizza le attività

Il System Sequence Diagram riportato di seguito descrive l'interazione tra un Supervisore e il sistema per visualizzare le attività eseguite nella piattaforma dagli utenti.

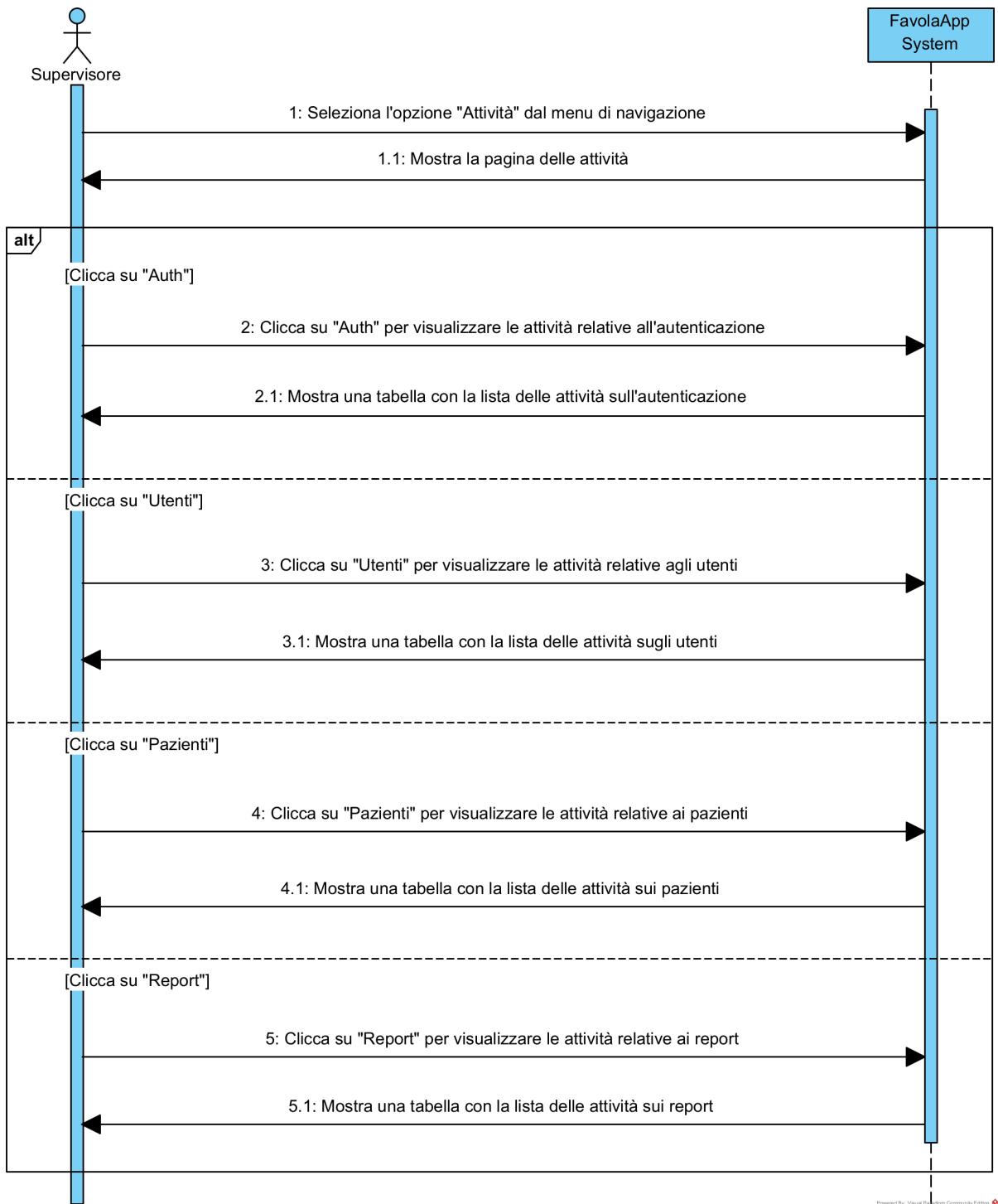


Figura 3.33: System Sequence Diagram per visualizzare le attività.

3.2.9 System Domain Model

Il System Domain Model è un modello concettuale che rappresenta gli elementi principali di un sistema software e le relazioni tra di essi, fornendo una visione astratta e statica della struttura del dominio operativo. Questo modello definisce le entità, i loro attributi e i ruoli che svolgono, descrivendo anche le interazioni e le dipendenze reciproche. Attraverso la rappresentazione grafica del System Domain Model, è possibile comprendere le componenti fondamentali del sistema e il loro contesto, facilitando così la progettazione. Tale modello funge da base per lo sviluppo delle funzionalità e l'implementazione del sistema, garantendo coerenza e chiarezza nella definizione degli elementi chiave.

Di seguito è riportato il modello del sistema, in cui l'associazione "Gestisce" rappresenta tutte le operazioni disponibili, quali visualizzazione, modifica ed eventuale eliminazione. Si segnala inoltre che l'associazione tra utenti identificati come "Colleghi" è stata omessa, poiché le uniche operazioni consentite in questo caso riguardano esclusivamente la visualizzazione da parte di un utente autenticato. Questa scelta è stata fatta anche per mantenere il modello più chiaro e visivamente ordinato.

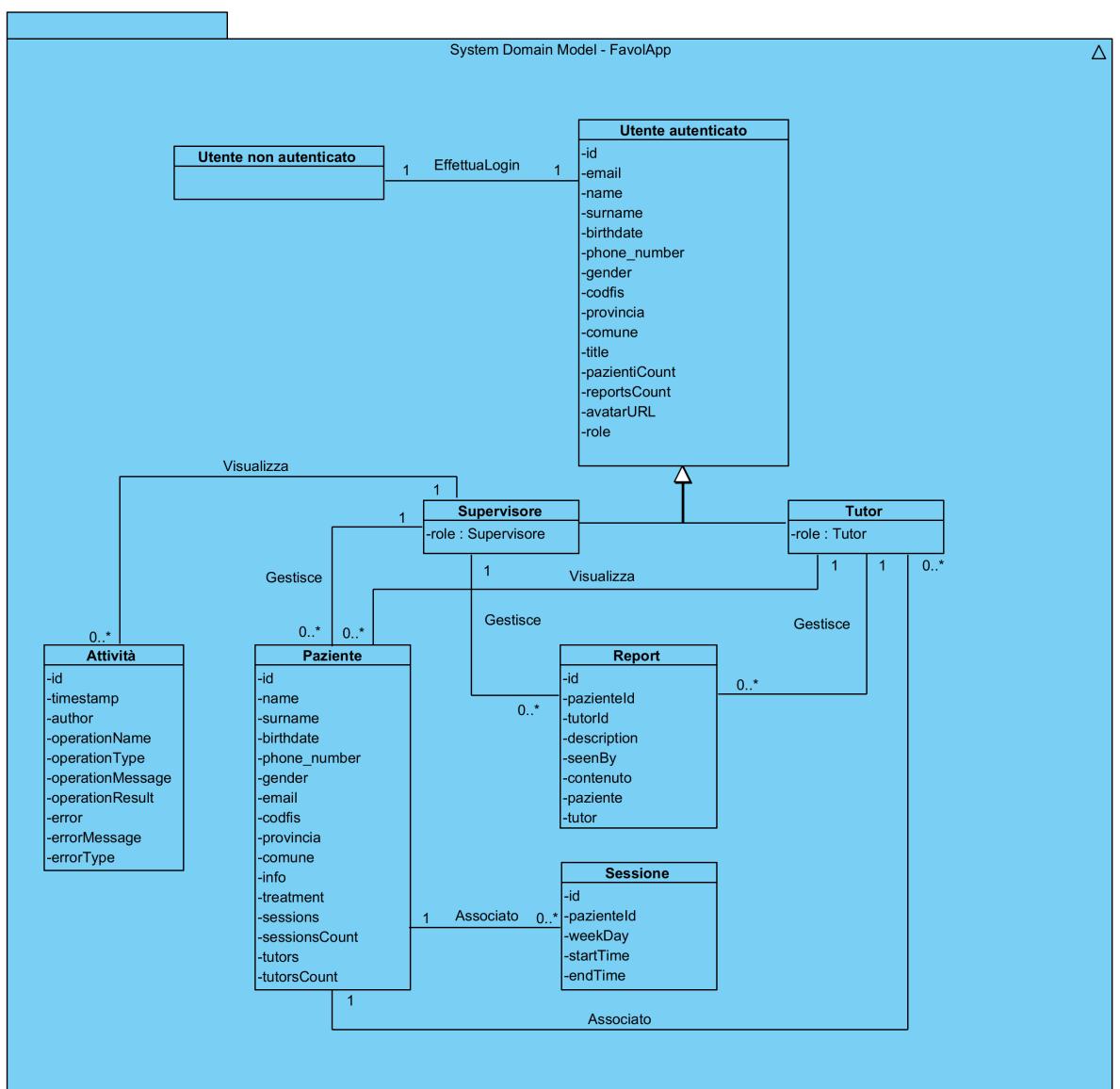


Figura 3.34: System Domain Model - FavolApp.

3.3 Web App Map

Di seguito la mappa dell'applicazione web dove vengono descritte tutte le sue sezioni:

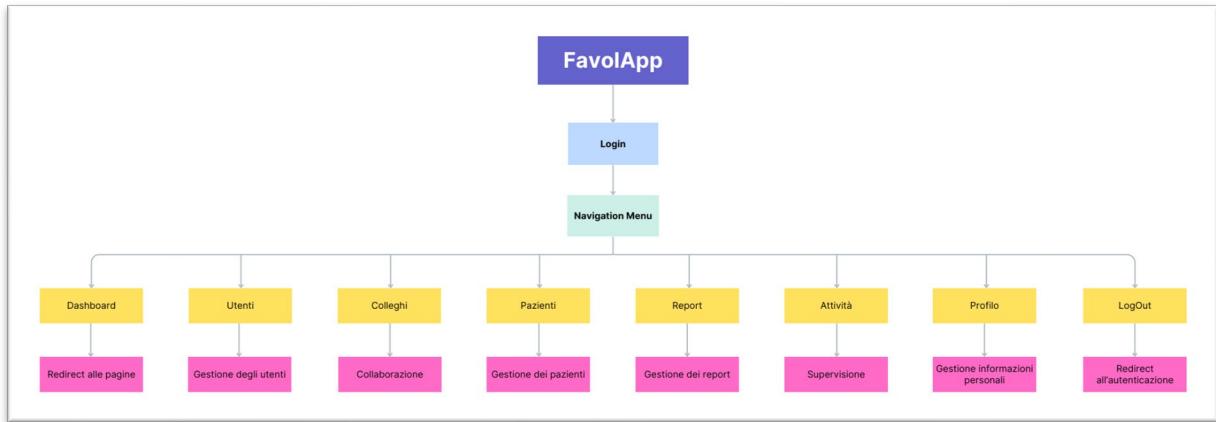


Figura 3.35: Web App Map per il Supervisore.

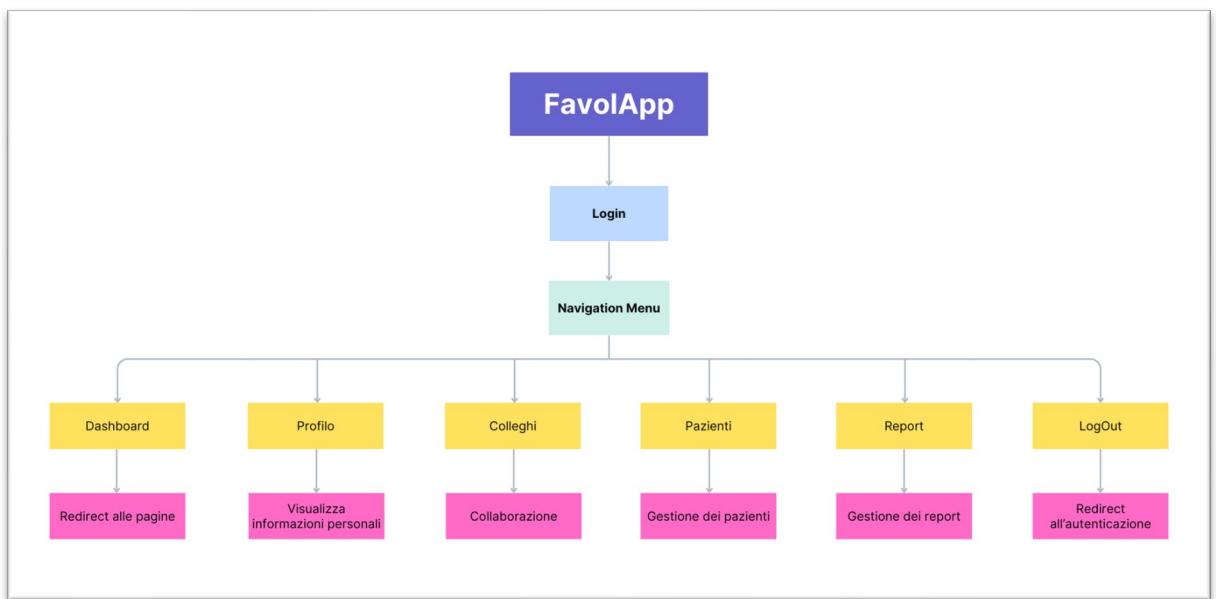


Figura 3.36: Web App Map per il Tutor.

3.4 Mockup

Di seguito vengono presentati i mockup dell'applicazione, realizzati utilizzando Figma. Questi mockup offrono una rappresentazione visiva dettagliata delle diverse sezioni dell'interfaccia utente e riflettono il design previsto per l'app una volta completata. Ogni immagine illustra la disposizione degli elementi, le scelte di layout e le funzionalità principali, permettendo di avere un'idea chiara di come sarà l'esperienza utente finale. I mockup servono anche come riferimento per il team di sviluppo e per eventuali miglioramenti o modifiche da apportare durante le fasi successive del progetto.

La pagina "Utenti" e "Attività" non sono accessibili ai Tutor e alcune operazioni potrebbero variare in base alla tipologia di utente.



Figura 3.37: Pagina di presentazione.

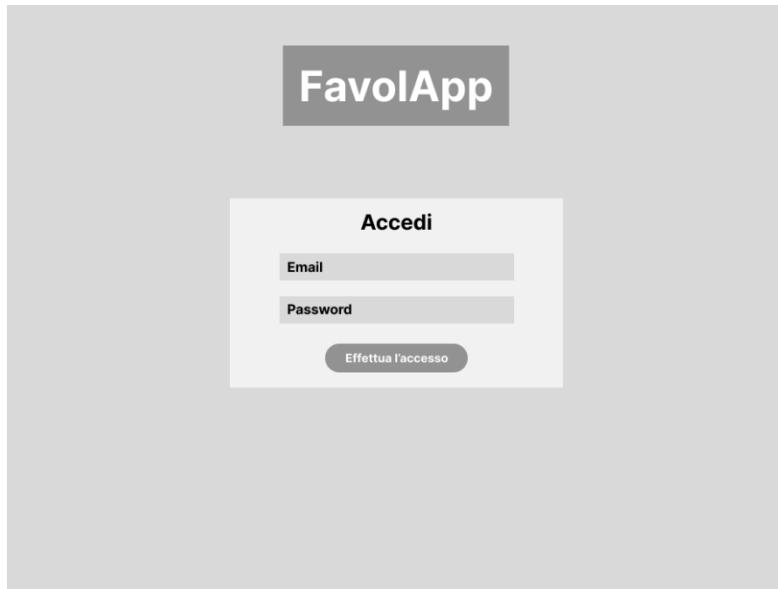


Figura 3.38: Pagina di autenticazione.

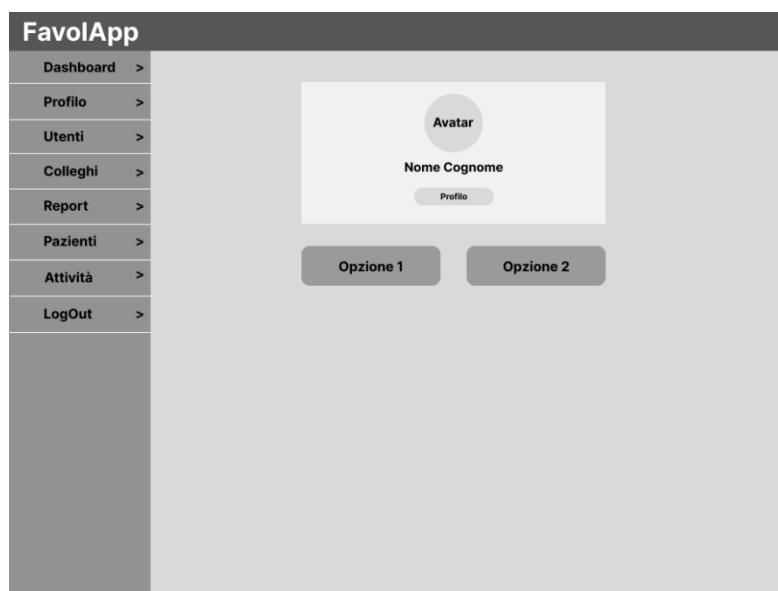


Figura 3.39: Pagina del profilo.

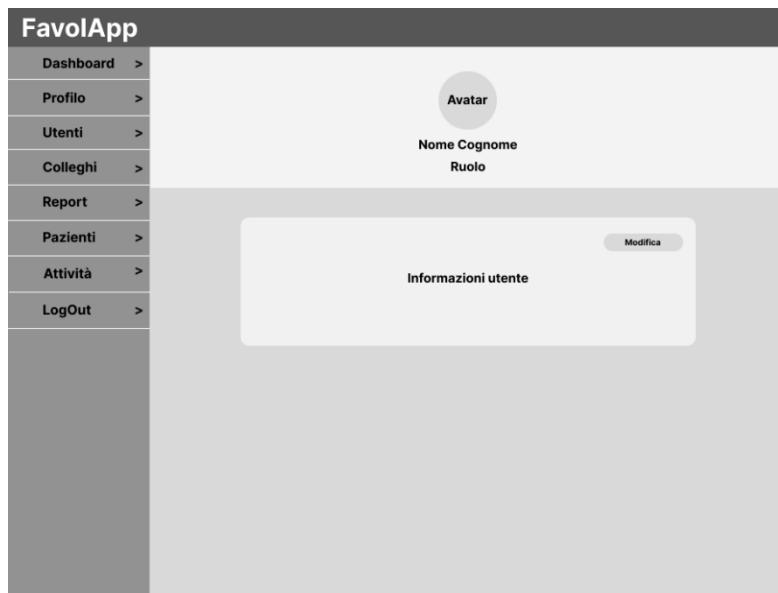


Figura 3.40: Pagina del profilo.

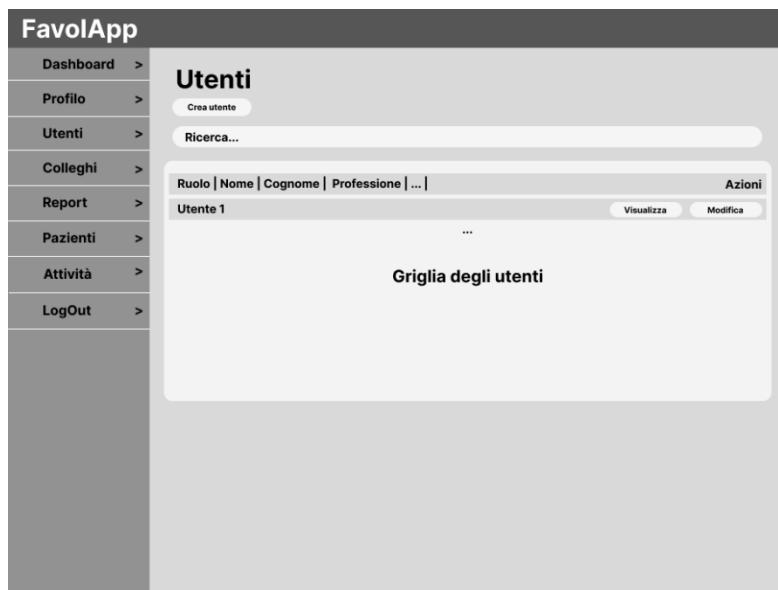


Figura 3.41: Pagina per la gestione degli utenti.



Figura 3.42: Pagina per la visualizzazione dei colleghi.



Figura 3.43: Pagina per la gestione dei report.



Figura 3.44: Pagina per la gestione dei pazienti.

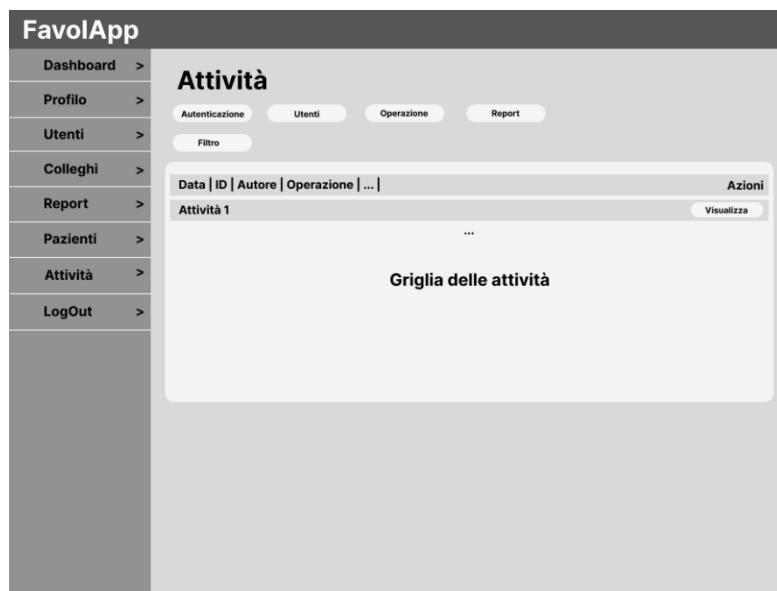


Figura 3.45: Pagina per la visualizzazione delle attività.

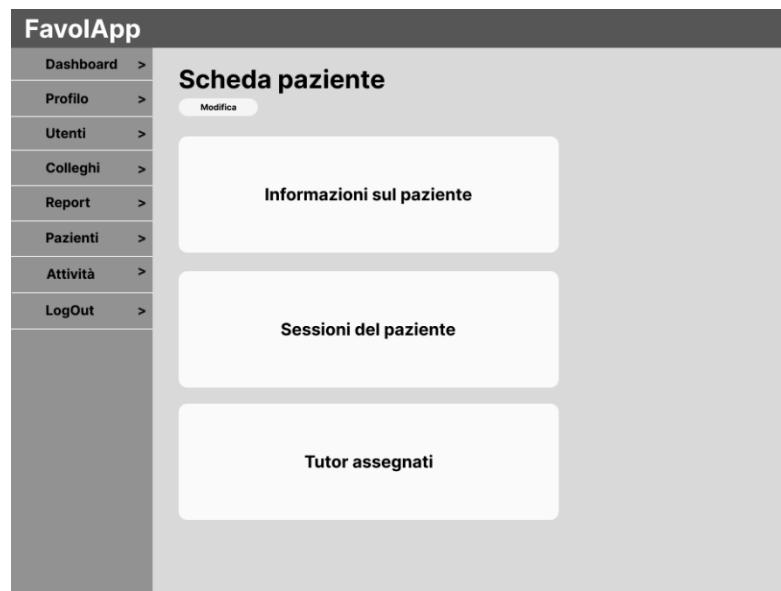


Figura 3.46: Pagina per la visualizzazione di un paziente.

4 Architettura e progettazione del software

In questa fase viene definita l'architettura del prodotto software, con un'attenzione particolare all'individuazione dei componenti che compongono il sistema e alle modalità di interazione tra di essi.

4.1 Decisioni architetturali e progettuali

In questa sezione vengono delineati gli stili e i pattern che guideranno lo sviluppo del progetto. Basandosi sulle esigenze e sugli obiettivi dell'applicazione, le decisioni architetturali e progettuali adottate sono le seguenti:

- **Pattern Architetturale (MVC)**

Il design dell'applicazione separa le responsabilità principali:

- **Model:** gestione e persistenza dei dati.
- **View:** presentazione e interazione con l'utente.
- **Controller:** logica applicativa e orchestrazione dei dati tra le componenti.

- **Stile Architetturale (Microservizi)**

L'applicazione adotta uno stile architetturale a Microservizi, suddividendo le funzionalità in servizi indipendenti che comunicano tra loro tramite interfacce ben definite. Ogni microservizio è progettato per gestire una specifica funzione o responsabilità, consentendo una maggiore flessibilità, scalabilità e facilità di manutenzione. Questo approccio facilita l'aggiornamento e l'evoluzione delle singole componenti senza influenzare l'intero sistema.

- **Pattern di Design**

- **GRASP (General Responsibility Assignment Software Patterns):** utilizzato per definire le responsabilità principali dei componenti del sistema e garantire una chiara separazione dei ruoli.
- **Observer:** implementato per sincronizzare in tempo reale i dati tra le diverse componenti del sistema.
- **Proxy:** utilizzato per astrarre la complessità dei servizi backend e semplificare la comunicazione con le componenti frontend.

4.1.1 Web Application

La scelta di sviluppare una Web Application è stata dettata dalla necessità di garantire massima portabilità e accessibilità. La natura web-based permette agli utenti di accedere all'applicazione da qualsiasi dispositivo dotato di un browser compatibile, senza necessità di installazioni. Questo approccio consente di beneficiare dei seguenti vantaggi:

- **Aggiornamenti automatici:** l'applicazione è sempre disponibile nella sua ultima versione senza necessità di intervento da parte dell'utente.
- **Bassi costi di manutenzione:** grazie all'adozione di infrastrutture che riducono la complessità gestionale e il lavoro amministrativo.
- **Alta scalabilità:** il sistema può gestire automaticamente il carico, adattandosi alle variazioni della domanda.

- **Facilità di manutenzione:** l'architettura basata su microservizi e la modularità del frontend semplificano la gestione delle funzionalità e l'applicazione di aggiornamenti.
- **Indipendenza dalla piattaforma:** l'applicazione è utilizzabile su qualsiasi dispositivo connesso a Internet.

4.2 Pattern architetturale – Model View Controller (MVC)

Il pattern architetturale scelto per l'applicazione è Model-View-Controller (MVC), che consente una chiara separazione delle responsabilità tra i dati (Model), la logica di gestione (Controller) e l'interfaccia utente (View). Questa separazione semplifica la manutenzione del sistema e rende il codice più modulare.

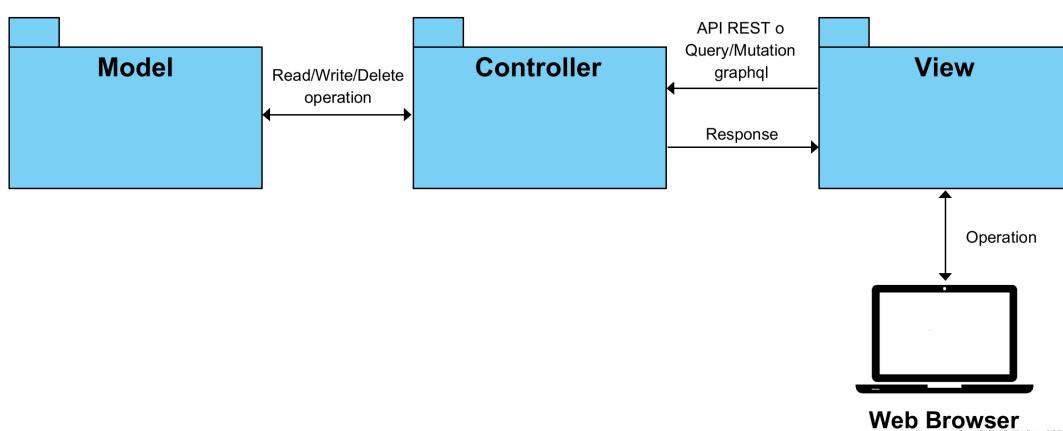


Figura 4.1: Model View Controller ad alto livello.

Il flusso del pattern MVC per questa applicazione segue questi passaggi:

- L'utente interagisce con l'interfaccia utente (View), ad esempio inviando una richiesta per visualizzare o aggiornare dei dati.
- La richiesta viene inviata al backend tramite un protocollo che consente lo scambio strutturato di dati.
- Il backend elabora la richiesta tramite un componente responsabile della logica applicativa (Controller).
- Il Controller interagisce con il sistema di gestione dei dati (Model) per recuperare, salvare o modificare i dati richiesti.
- I dati elaborati vengono restituiti alla View, che li presenta in modo leggibile e interattivo all'utente.

4.3 Stile architetturale – Microservizi

Le architetture a microservizi hanno origine da modelli come la SOA (Service-Oriented Architecture), con un'enfasi particolare sullo sviluppo software. Esse superano il tradizionale approccio monolitico, favorendo un modello più flessibile, incentrato sulla suddivisione del sistema in piccoli componenti indipendenti che corrispondono a specifiche funzionalità del software.

Questi componenti, o "microservizi", vengono progettati per essere disaccoppiati, garantendo un elevato grado di indipendenza durante lo sviluppo. Questo approccio consente di combinare i diversi moduli in modo personalizzabile per creare applicazioni su misura. Inoltre, la modularità permette di aggiornare o modificare i singoli componenti senza dover ricompilare o ridistribuire l'intero sistema, offrendo una notevole agilità operativa e la possibilità di riutilizzare i componenti in contesti differenti.

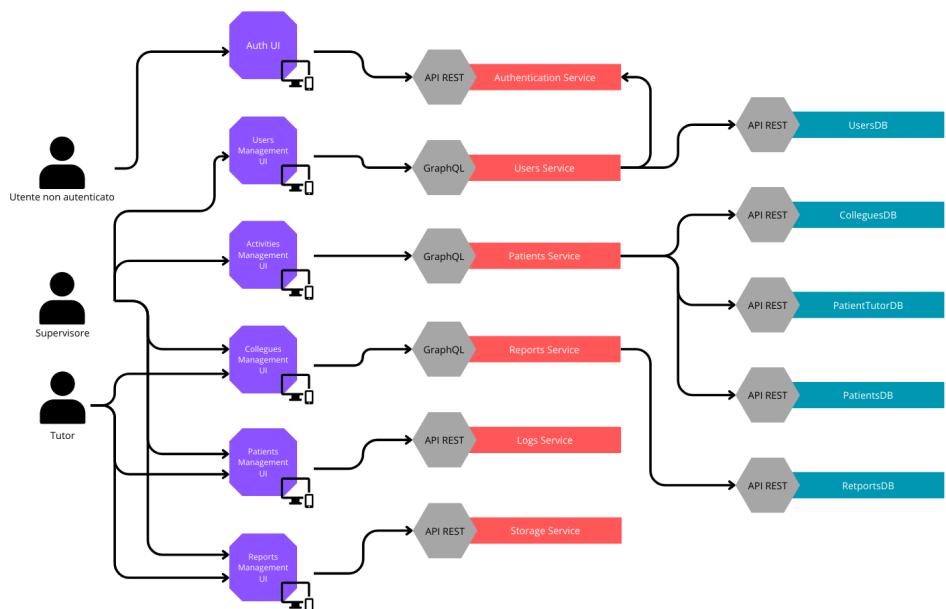


Figura 4.2: Architettura a microservizi ad alto livello.

La suddivisione in microservizi ha permesso di sviluppare e distribuire ciascun servizio in maniera autonoma, grazie al fatto che ogni componente è progettato per essere debolmente accoppiato.

4.4 GraphQL

GraphQL è un linguaggio di query per le API e un runtime per soddisfare quelle query con i dati. La sua principale caratteristica è la capacità di consentire ai client di richiedere solo i dati di cui hanno bisogno, con un formato e una struttura personalizzabile per ogni richiesta. Ciò si differenzia dai sistemi più tradizionali, come REST, in cui i client sono limitati alle strutture dei dati predefinite dagli endpoint.

In GraphQL, i client comunicano con il server tramite un singolo endpoint, inviando query strutturate che specificano chiaramente i dati richiesti. Il server, utilizzando lo schema GraphQL, interpreta la query, accede alle fonti di dati necessarie e restituisce solo le informazioni richieste, con una struttura corrispondente alla query stessa. Questo approccio consente di ottimizzare il flusso di dati, riducendo il carico della rete e migliorando l'efficienza complessiva del sistema.

4.4.1 Funzionamento

Il funzionamento di GraphQL si basa su quattro componenti principali: schema, query, mutazioni e subscription. Ognuno di questi elementi è progettato per gestire un aspetto specifico della comunicazione tra il client e il server.

Schema

Lo schema è il cuore dell'architettura di GraphQL. Definisce i tipi di dati disponibili, le loro relazioni e le operazioni che i client possono eseguire. È una sorta di "contratto" tra il client e il server, che garantisce che entrambe le parti abbiano una comprensione chiara delle possibilità offerte dall'API.

- **Tipi di dati:** Ogni entità è rappresentata da un tipo di oggetto. Ad esempio, un utente potrebbe essere rappresentato da un tipo User con campi come id, name, e email.
- **Query:** Sono definite nello schema per consentire al client di recuperare i dati. Ogni query specifica i tipi di dati e i campi disponibili.
- **Mutazioni:** Consentono di definire operazioni per creare, aggiornare o eliminare dati. Ogni mutazione specifica i campi che possono essere modificati e i dati da restituire.
- **Subscription:** Sono utilizzate per ricevere notifiche in tempo reale quando si verificano determinati eventi.

Query

Le query sono richieste inviate dal client per ottenere dati dal server. In GraphQL, le query sono flessibili e precise: il client specifica esattamente quali campi e quali relazioni desidera ottenere.

Mutazioni

Le mutazioni consentono al client di inviare richieste al server per modificare i dati. Queste operazioni sono simili ai metodi POST, PUT o DELETE in REST, ma sono definite esplicitamente nello schema.

Subscription

Le subscription sono un potente strumento per le applicazioni che richiedono aggiornamenti in tempo reale. Utilizzando protocolli come WebSocket, il server invia notifiche al client ogni volta che si verifica un evento specifico, come la creazione di un nuovo ordine o l'aggiornamento dello stato di un processo.

Questo approccio è particolarmente utile in applicazioni interattive, come dashboard, chat o sistemi di monitoraggio, dove i dati devono essere sempre aggiornati senza la necessità di richieste manuali da parte del client.

4.4.2 Vantaggi

GraphQL offre numerosi vantaggi che lo rendono una scelta sempre più popolare per lo sviluppo di API moderne. Grazie alla sua flessibilità e alla capacità di ottimizzare il flusso di dati tra client e server, si distingue per la possibilità di richiedere solo le informazioni necessarie, riducendo così inefficienze e sprechi. Inoltre, la centralizzazione delle richieste su un unico endpoint semplifica la gestione delle API e ne migliora la scalabilità. Con un design che mette il client al centro, GraphQL consente una modellazione precisa delle richieste e supporta relazioni complesse tra entità, il tutto mantenendo un'evoluzione graduale e non distruttiva. Questi aspetti lo rendono una soluzione ideale per rispondere alle esigenze delle applicazioni moderne.

- **Richieste mirate e ottimizzazione dei dati**

Uno dei vantaggi principali di GraphQL è la possibilità di richiedere solo i dati necessari. Ciò elimina il problema dell'over-fetching (ottenere più dati del necessario) e dell'under-fetching (ottenere dati insufficienti), tipici delle API REST.

- **Un singolo endpoint**

GraphQL centralizza tutte le richieste su un unico endpoint, semplificando la gestione delle API. Questo riduce la complessità dell'architettura e consente di mantenere un'interfaccia coerente anche con l'aggiunta di nuove funzionalità.

- **Flessibilità per il client**

L'approccio orientato al client consente di modellare le richieste in base alle esigenze specifiche dell'applicazione. Ciò significa che gli sviluppatori possono adattare facilmente l'API alle esigenze in evoluzione del frontend, senza modificare la logica del backend.

- **Supporto a relazioni complesse**

GraphQL è progettato per gestire relazioni complesse tra entità. Ad esempio, una query può richiedere dettagli su un prodotto, i suoi fornitori e gli ordini associati, il tutto in una singola richiesta. Questo semplifica la gestione dei dati complessi e migliora l'efficienza.

- **Evoluzione non distruttiva**

L'aggiunta di nuovi campi o tipi nello schema GraphQL non interrompe le richieste esistenti, consentendo un'evoluzione graduale dell'API. Questa caratteristica è particolarmente utile per mantenere la compatibilità con le versioni precedenti.

4.5 REST

REST (REpresentational State Transfer) rappresenta uno stile architettonico che definisce specifici vincoli per progettare sistemi distribuiti, utilizzando standard consolidati per la rappresentazione dei dati e il protocollo HTTP per il loro trasferimento. Quando un client invoca un'API RESTful, il server risponde fornendo una rappresentazione dello stato della risorsa richiesta.

Per ottenere una risorsa dal server, il client utilizza un identificatore univoco della risorsa (ad esempio, un URL) e specifica l'operazione desiderata attraverso i metodi HTTP, che determinano il tipo di interazione:

- **POST**: consente di creare una nuova risorsa.
- **GET**: recupera una o più risorse già esistenti.
- **PUT**: aggiorna una risorsa esistente o la crea, se non è già presente.
- **DELETE**: elimina una risorsa specificata.

4.6 Vista componenti e connettori

Questa sezione descrive l'organizzazione interna del sistema software attraverso i principali componenti e le loro interazioni. I componenti rappresentano unità autonome che offrono funzionalità specifiche per il sistema, mentre i connettori definiscono i meccanismi di comunicazione e scambio di informazioni tra di essi. Nel diagramma in Figura 4.3 si evidenzia la separazione tra elaborazione e interazione, mettendo in luce come i componenti siano collegati tra loro e al database attraverso connettori. I connettori supportano lo scambio di dati, richieste e risposte, adottando modelli di chiamata a procedure (procedure call) tra i diversi elementi del sistema.

Il diagramma rappresenta due componenti principali: Client e Server. Sul lato del Server, l'architettura è suddivisa in diversi microservizi, ciascuno progettato per gestire operazioni specifiche in linea con le funzionalità richieste dagli attori del sistema. Il coordinamento delle richieste è affidato a un componente centrale, il MainHandler, che riceve le richieste dal Client e le inoltra ai servizi appropriati. I microservizi principali comprendono:

- **Authentication Service**, che gestisce l'autenticazione degli utenti.
- **Users Service**, responsabile della gestione delle informazioni sugli utenti, incluse operazioni di lettura, scrittura e modifica.
- **Patients Service**, che si occupa della gestione dei dati relativi ai pazienti.
- **Reports Service**, progettato per elaborare e gestire i report.
- **Logs Service**, che si occupa della registrazione e tracciamento delle attività del sistema.
- **Storage Service**, che gestisce il caricamento e la conservazione dei dati.

Ciascun servizio è collegato al proprio database tramite connettori di tipo "data access", i quali permettono un'interazione diretta tra il microservizio e il rispettivo database, garantendo l'accesso ai dati in modo modulare e indipendente. Ad esempio, il UsersDBAccess consente al servizio utenti di interagire con il database degli utenti, mentre il PatientDBAccess gestisce le operazioni sui dati dei pazienti.

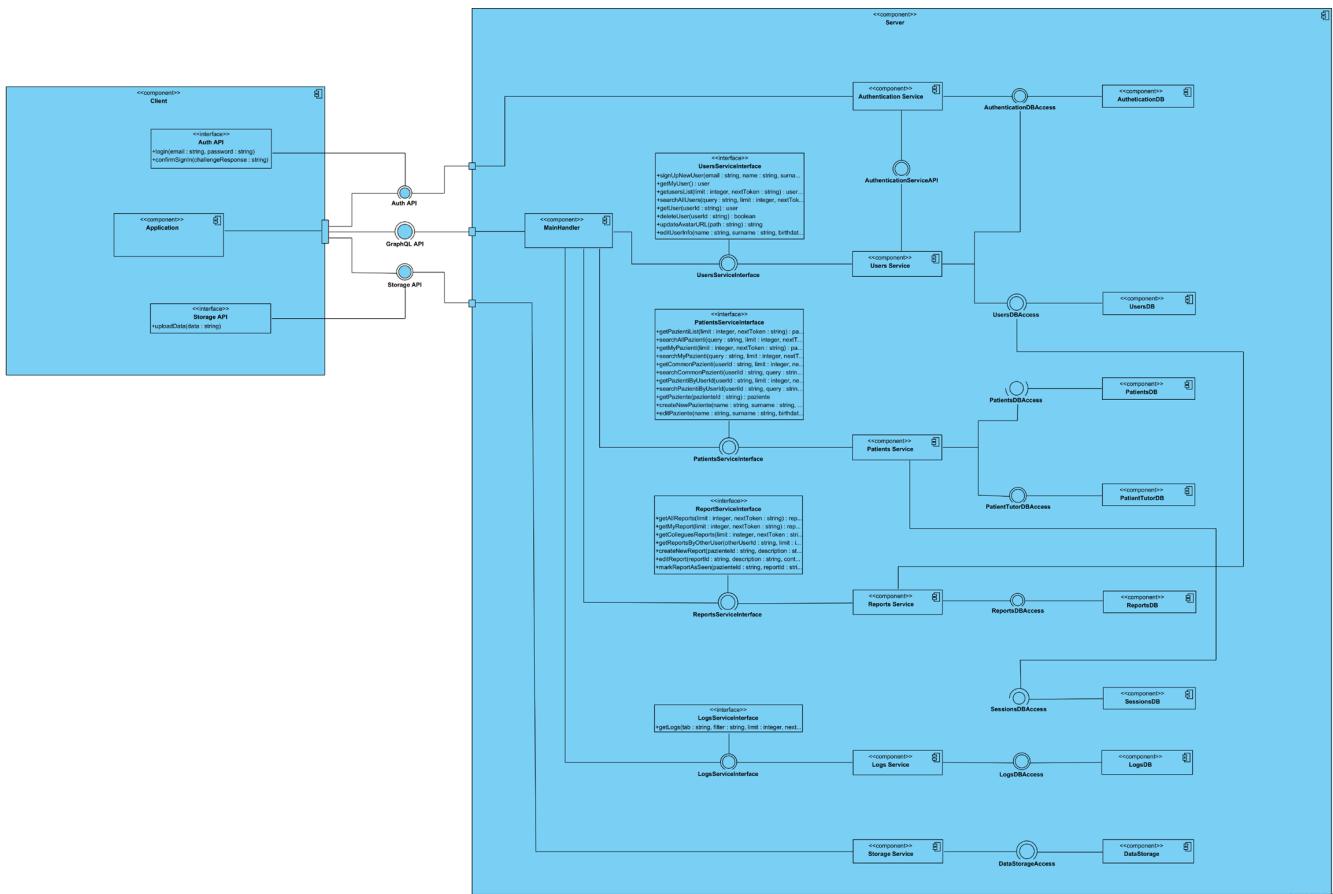


Figura 4.3: Vista componenti e connettori.

4.7 Sequence diagram di progettazione

In questa sezione vengono introdotti i Sequence Diagram di progettazione, che rappresentano le interazioni tra gli utenti e il sistema al livello intermedio, descrivendo il flusso delle comunicazioni attraverso i componenti identificati nella vista Componenti e Connettori. Questi diagrammi sono particolarmente utili per illustrare come le richieste degli utenti vengano gestite dal sistema, evidenziando il ruolo dei vari servizi e le relazioni tra i componenti.

Poiché ogni servizio del sistema offre una vasta gamma di funzionalità, non sarà possibile rappresentare tutte le operazioni disponibili. Per questo motivo, ci concentreremo sulle funzionalità più rappresentative e rilevanti per il sistema, selezionando quelle che offrono una chiara visione del comportamento del sistema e delle sue interazioni principali. Attraverso questi diagrammi, sarà possibile comprendere meglio i processi che avvengono dietro le quinte, dalle richieste iniziate dall'utente fino alla loro elaborazione e risposta finale.

Questa selezione consentirà di mantenere il focus sulle interazioni critiche e sui flussi di dati fondamentali, semplificando la comprensione e la valutazione del sistema, pur mantenendo una visione completa della sua architettura e funzionalità principali.

4.7.1 Sequence diagram di progettazione – Autenticazione

Il seguente sequence diagram illustra il flusso di interazione tra un utente non autenticato, l'applicazione client, il servizio di autenticazione e il database dell'autenticazione durante il processo di login.

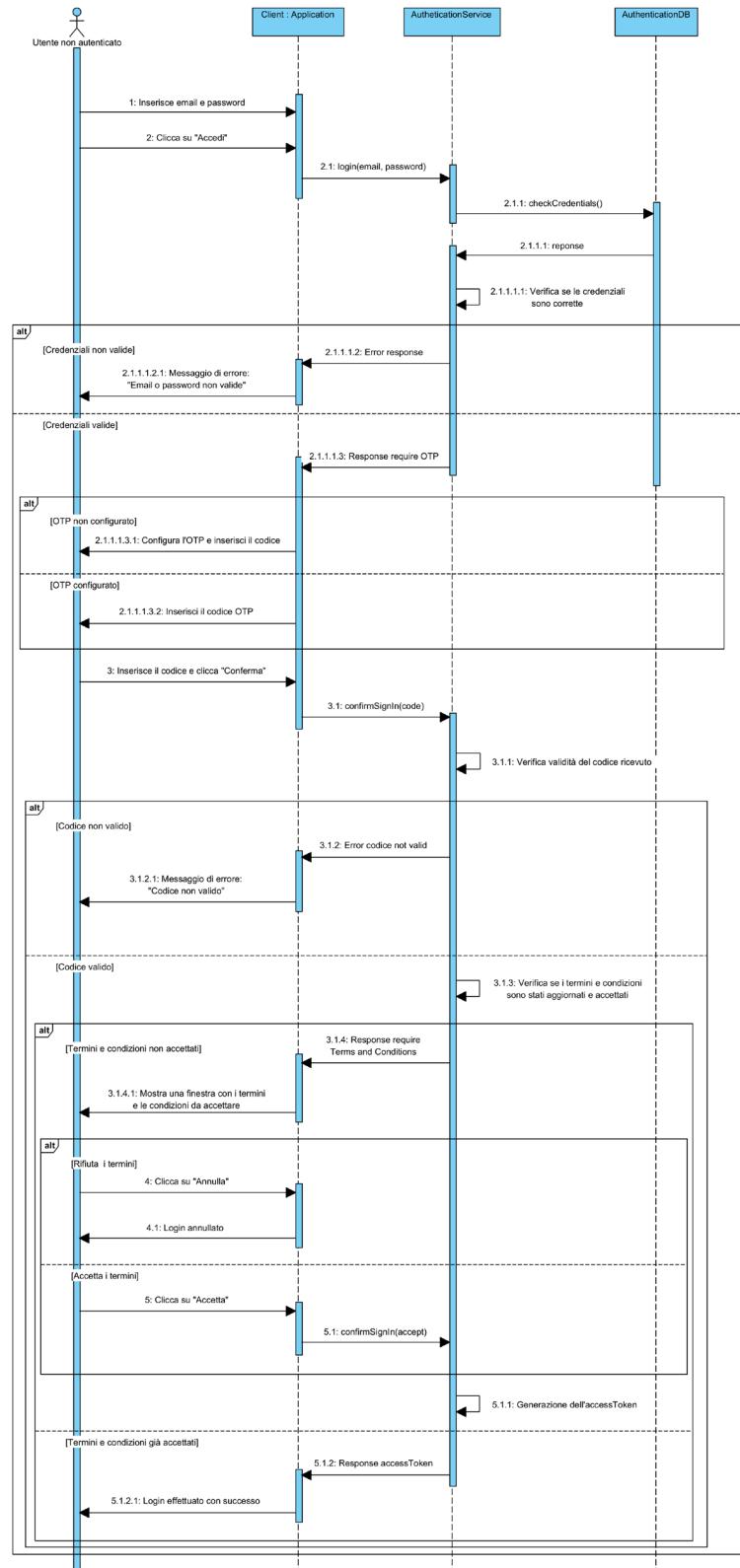


Figura 4.4: Sequence diagram di progettazione – Autenticazione.

Il processo si articola nei seguenti passaggi principali:

- 1 Inserimento delle credenziali:
 - L'utente inserisce l'email e la password e avvia la procedura di login cliccando su "Accedi".
 - L'applicazione client invia una richiesta al servizio di autenticazione con le credenziali fornite.
- 2 Verifica delle credenziali:
 - Il servizio di autenticazione interroga il database per verificare la validità delle credenziali fornite.
 - Se le credenziali sono errate, il sistema invia un messaggio di errore al client: "Email o password non valide".
 - Se le credenziali sono valide, viene richiesto un codice OTP per proseguire.
- 3 Gestione del codice OTP:
 - Se l'OTP non è configurato, il sistema guida l'utente nella configurazione e nella ricezione del codice.
 - Se l'OTP è già configurato, l'utente inserisce il codice ricevuto e conferma l'operazione.
 - In caso di codice non valido, il sistema restituisce un messaggio di errore.
 - Con un codice valido, si procede con la verifica di eventuali termini e condizioni.
- 4 Verifica dei termini e condizioni:
 - Se i termini e le condizioni non sono stati accettati o sono stati aggiornati, viene mostrata una finestra che richiede l'accettazione.
 - L'utente può scegliere di accettare o annullare l'operazione:
 - In caso di rifiuto, il login viene annullato.
 - In caso di accettazione, l'operazione prosegue.
- 5 Conferma e generazione del token:
 - Una volta accettati i termini (o se già accettati in precedenza), il sistema genera un access token e lo restituisce al client.
 - Il login viene completato con successo, e l'utente ottiene l'accesso al sistema.

4.7.2 Sequence diagram di progettazione – Utenti

Di seguito sono illustrati i sequence diagram relativi alle funzionalità più significative gestite dal UsersService. Questi diagrammi mostrano le principali interazioni tra i vari attori e componenti del sistema, evidenziando il flusso di richieste e risposte.

4.7.2.1 Registrazione utente

Il seguente sequence diagram descrive il processo di registrazione di un nuovo utente nel sistema, illustrando il flusso di interazione tra il supervisore (utente), l'applicazione client, il MainHandler, il UsersService, il database degli utenti (UsersDB) e il servizio di autenticazione (AuthenticationService).

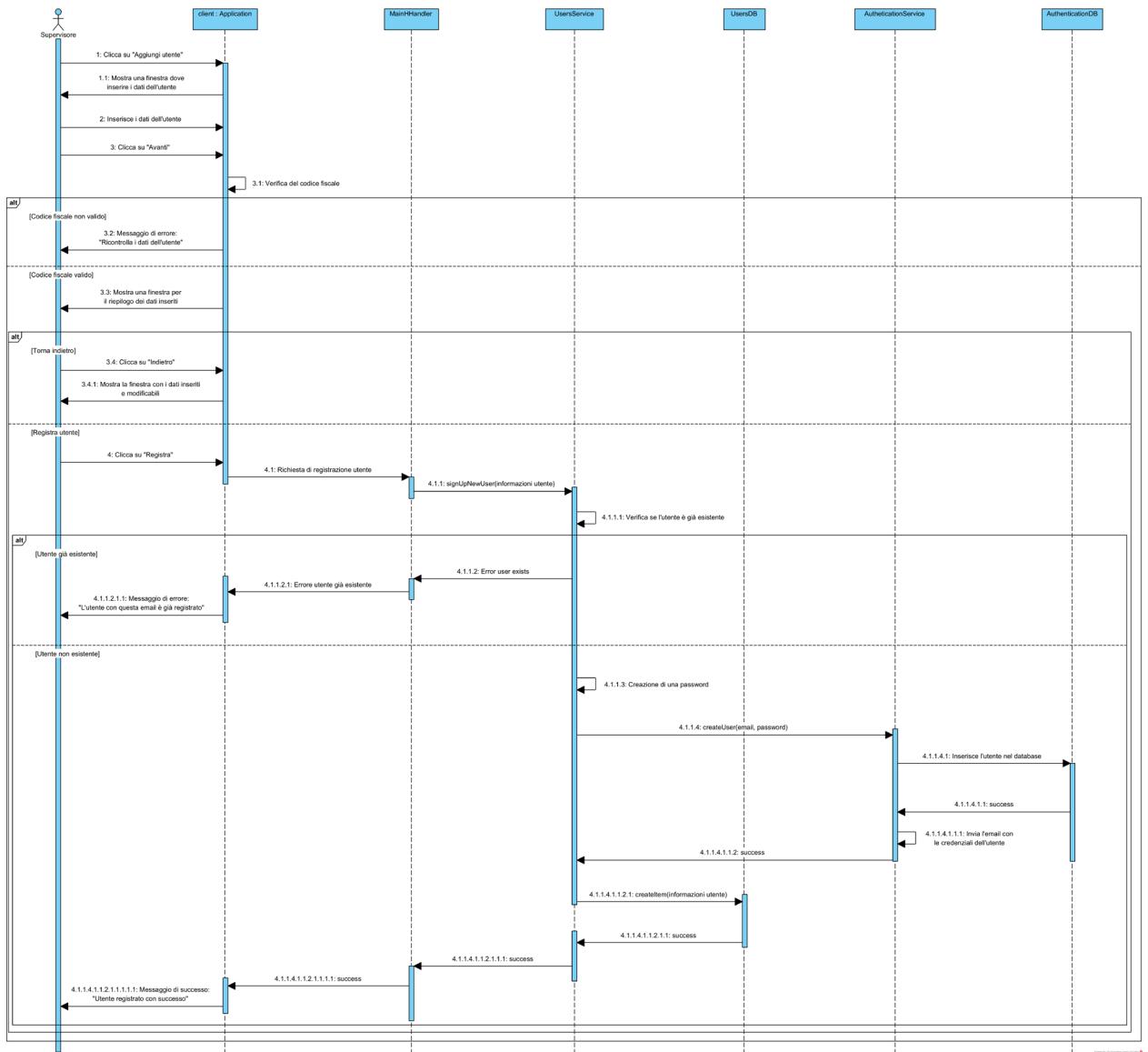


Figura 4.5: Sequence diagram di progettazione – Registrazione utente.

Il flusso è suddiviso nei seguenti passaggi principali:

- 1 Inizio della registrazione:
 - Il supervisore clicca su "Aggiungi utente" nell'applicazione.
 - L'app mostra una finestra dove vengono inseriti i dati del nuovo utente (ad esempio, nome, cognome, email, codice fiscale).
 - Una volta compilati i campi richiesti, l'utente clicca su "Avanti", e l'applicazione invia i dati al client per la verifica del codice fiscale.
 - 2 Verifica del codice fiscale:
 - Il client controlla la validità del codice fiscale.
 - Se il codice non è valido, viene mostrato un messaggio di errore: "Ricontrolla il codice fiscale".
 - In caso di codice valido, il processo prosegue con la verifica e conferma dei dati inseriti.

- 3 Conferma dei dati:
 - Viene mostrata una finestra riepilogativa per consentire al supervisore di verificare e modificare i dati prima della registrazione definitiva.
- 4 Creazione dell'utente:
 - Dopo aver confermato i dati, il supervisore clicca su "Registra".
 - Viene inviata una richiesta di registrazione al MainHandler che la inoltra allo UsersService.
 - Lo UsersService controlla se l'utente esiste già:
 - Se l'utente esiste, viene mostrato un messaggio di errore: "Utente con questa email già registrato".
 - Se l'utente non esiste, il sistema genera una password e procede alla registrazione.
- 5 Inserimento nel database e invio email:
 - Il UsersService aggiunge il nuovo utente al database degli utenti.
 - Il AuthenticationService invia un'email con le credenziali di accesso all'utente appena registrato.
 - Infine, viene mostrato un messaggio di successo: "Utente registrato con successo".

4.7.2.2 Ottenimento utente

Il seguente sequence diagram rappresenta il processo di visualizzazione del profilo di un utente da parte di un altro utente (ad esempio, un supervisore o un tutor). Viene descritto il flusso di interazione tra il client, il UsersService, il database degli utenti (UsersDB) e il database delle associazioni paziente-tutor (PatientTutorDB).

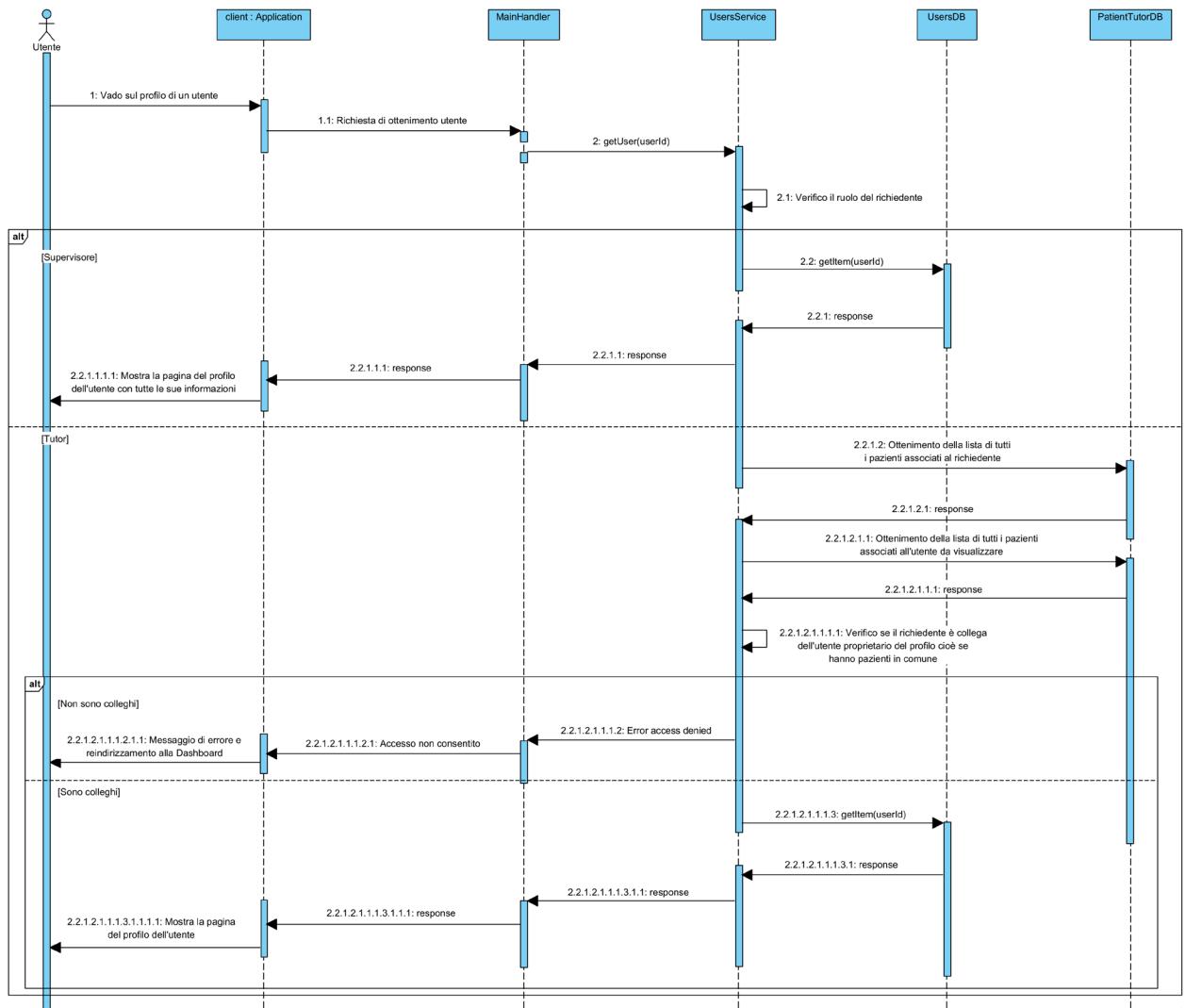


Figura 4.6: Sequence diagram di progettazione – Ottenimento utente.

Il flusso è suddiviso nei seguenti passaggi principali:

- 1 Richiesta di accesso al profilo:
 - L'utente seleziona un profilo da visualizzare.
 - Il client invia una richiesta al MainHandler che la inoltra allo UserService per ottenere i dettagli del profilo dell'utente specificato.
- 2 Verifica del ruolo del richiedente:
 - Il UserService verifica il ruolo dell'utente richiedente (ad esempio, supervisore o tutor).
 - A seconda del ruolo, vengono determinati i permessi di accesso ai dati richiesti.
- 3 Recupero dei dati dal database:
 - Il UserService interroga il UsersDB per ottenere i dettagli del profilo dell'utente specificato.
 - I dati vengono restituiti al client per essere visualizzati.

- 4 Comportamento specifico per il ruolo di tutor:
 - Se il richiedente è un tutor, il UserService effettua ulteriori verifiche per determinare se il tutor è collegato all'utente il cui profilo vuole visualizzare, controllando la presenza di pazienti in comune nel PatientTutorDB.
 - In caso di connessione valida (es. pazienti condivisi), il profilo viene mostrato.
 - Se non ci sono collegamenti validi, viene restituito un messaggio di errore con accesso negato e il richiedente viene reindirizzato alla dashboard.
- 5 Visualizzazione del profilo:
 - In caso di accesso autorizzato, il client visualizza il profilo completo dell'utente con tutte le informazioni disponibili.

4.7.2.3 Elimina utente

Il seguente sequence diagram descrive il processo di eliminazione di un utente dal sistema, illustrando le interazioni tra il supervisore, il client, il MainHandler, il UserService, il database degli utenti (UsersDB), il servizio di autenticazione (AuthenticationService) e il database dell'autenticazione (AuthenticationDB).

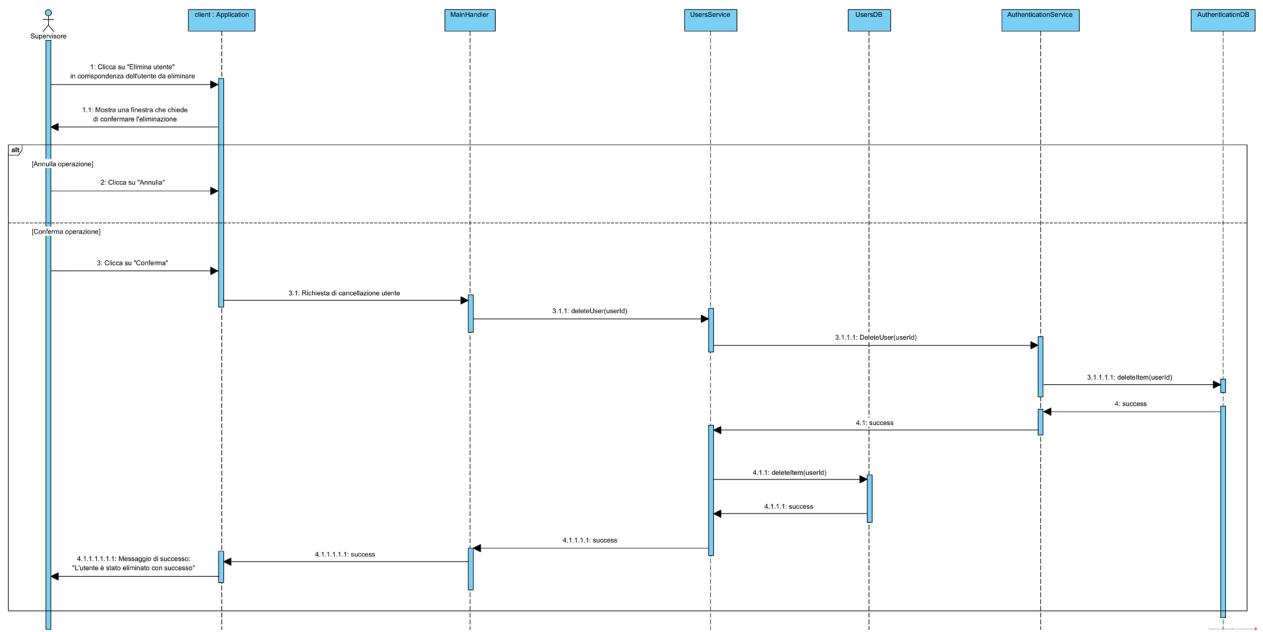


Figura 4.7: Sequence diagram di progettazione – Elimina utente.

Il flusso principale si articola come segue:

1 Inizio dell'eliminazione:

- Il supervisore seleziona l'opzione "Elimina utente" corrispondente all'utente da rimuovere.
 - L'applicazione client mostra una finestra di conferma per chiedere al supervisore se intende procedere con l'operazione.

2 Scelta dell'operazione:

- Il supervisore può scegliere di annullare l'operazione cliccando su "Annulla". In questo caso, l'eliminazione viene interrotta.
 - Se il supervisore conferma l'eliminazione cliccando su "Conferma", il client invia una richiesta al MainHandler che la inoltra allo UserService per avviare il processo di rimozione.

3 Eliminazione dell'utente:

- Il UserService inoltra la richiesta al UsersDB per rimuovere l'utente specificato (identificato dal suo userId).
 - Dopo che l'utente è stato eliminato dal UsersDB, il UserService invia una richiesta al AuthenticationService per eliminare eventuali credenziali associate all'utente.
 - Il AuthenticationService rimuove le credenziali dell'utente dal AuthenticationDB.

4 Conferma dell'eliminazione:

- Una volta completata l'eliminazione sia dal database degli utenti che da quello delle credenziali, il UserService restituisce un messaggio di successo al client.
 - Il client notifica al supervisore che "L'utente è stato eliminato con successo".

4.7.3 Sequence diagram di progettazione – Pazienti

Di seguito sono illustrati i sequence diagram relativi alle funzionalità più significative gestite dal PatientsService. Questi diagrammi mostrano le principali interazioni tra i vari attori e componenti del sistema, evidenziando il flusso di richieste e risposte.

4.7.3.1 Creazione paziente

Il seguente sequence diagram descrive il processo di creazione di un nuovo paziente nel sistema da parte di un supervisore. Viene mostrata l'interazione tra il supervisore, il client, il MainHandler, il PatientService, il PatientDB, il SessionDB e il PatientTutorDB.

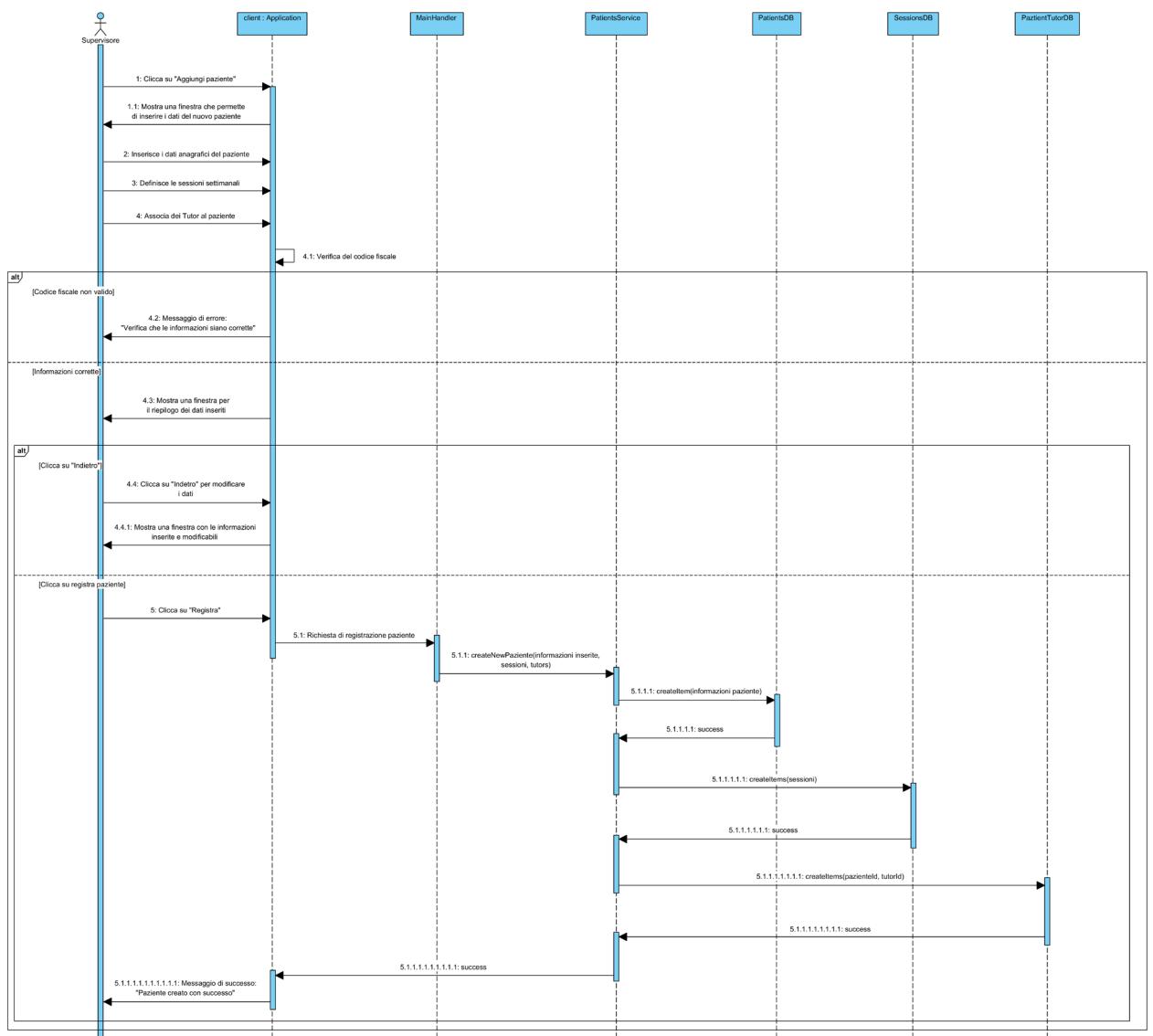


Figura 4.8: Sequence diagram di progettazione – Creazione Paziente.

Il flusso principale si sviluppa come segue:

- 1 Inizio del processo di creazione:
 - Il supervisore clicca su "Aggiungi paziente" nell'applicazione client.
 - Il client visualizza una finestra per raccogliere i dettagli del nuovo paziente, tra cui dati anagrafici, sessioni settimanali e tutor associati.
- 2 Verifica del codice fiscale:
 - Dopo aver inserito i dati richiesti, il client verifica la validità del codice fiscale.
 - Se il codice fiscale non è valido, viene mostrato un messaggio di errore: "Verifica che le informazioni siano corrette".
- 3 Revisione dei dati:
 - In caso di codice fiscale valido, il supervisore può scegliere di rivedere i dati inseriti, visualizzando una finestra di riepilogo.
 - Se necessario, il supervisore può modificare le informazioni e visualizzarle nuovamente per conferma.
- 4 Registrazione del paziente:
 - Quando i dati sono completi e corretti, il supervisore clicca su "Registra".
 - Il client invia una richiesta al MainHandler che la inoltre al PatientService per creare il nuovo paziente, includendo le informazioni personali, le sessioni settimanali e i tutor associati.
- 5 Salvataggio nel database:
 - Il PatientService registra i dati del paziente nel PatientDB.
 - Successivamente, registra le sessioni nel SessionDB e le associazioni tutor-paziente nel PatientTutorDB.
 - Ogni operazione restituisce un messaggio di successo.
- 6 Conferma della creazione:
 - Una volta completato il processo, il PatientService notifica il client con un messaggio di successo: "Paziente creato con successo".
 - Il client visualizza il messaggio per informare il supervisore del completamento dell'operazione.

4.7.3.2 Modifica paziente

Il seguente sequence diagram rappresenta il processo di modifica delle informazioni di un paziente da parte di un supervisore. Viene mostrata l'interazione tra il supervisore, il client, il MainHandler, il PatientService, il PatientDB, il SessionDB, e il PatientTutorDB.

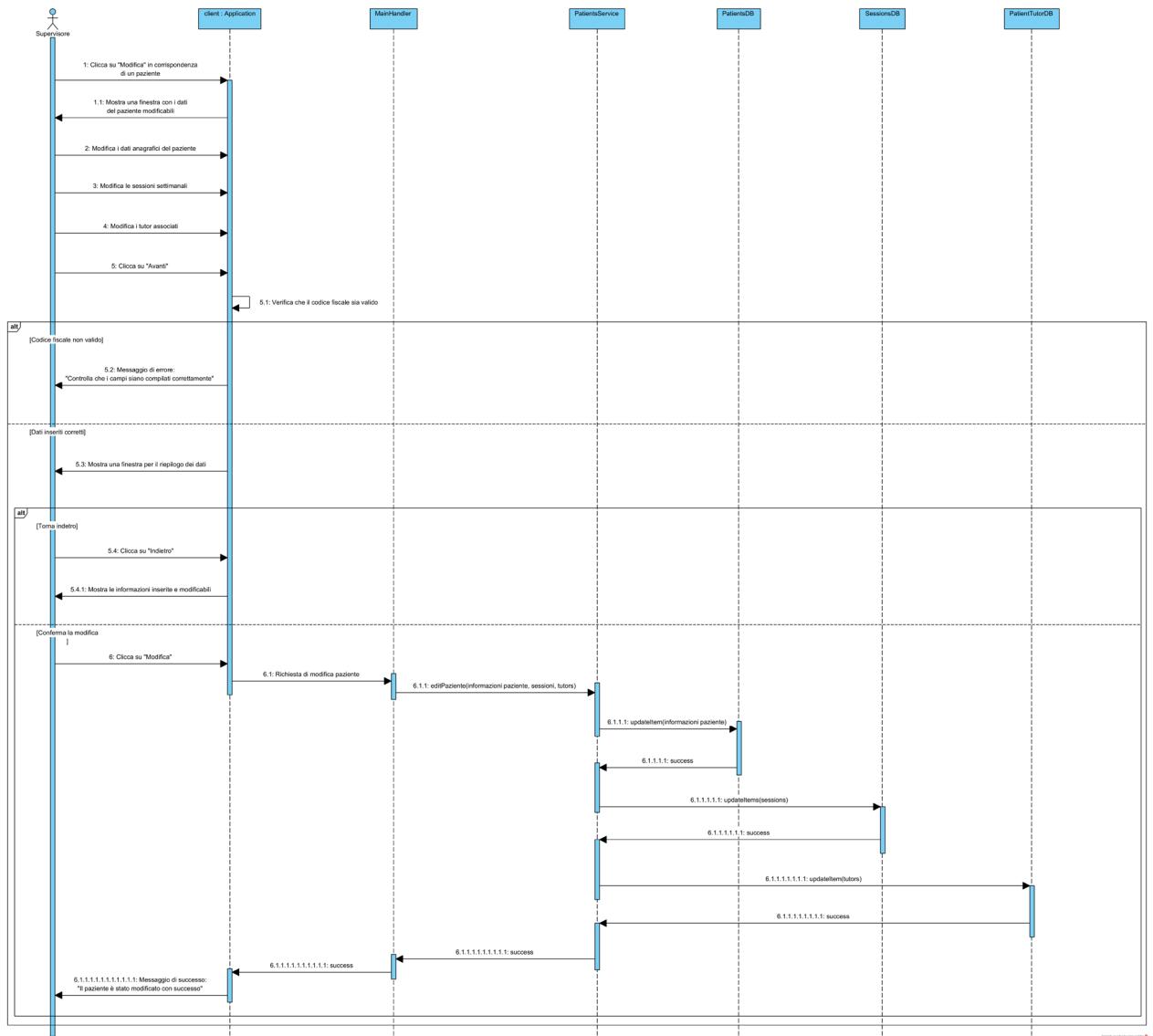


Figura 4.9: Sequence diagram di progettazione – Modifica Paziente.

Il flusso principale si articola nei seguenti passaggi:

- 1 Avvio della modifica:
 - Il supervisore clicca su "Modifica" accanto al paziente che desidera aggiornare.
 - Il client mostra una finestra con i dati attuali del paziente, permettendo al supervisore di modificare i dati anagrafici, le sessioni settimanali e i tutor associati.
- 2 Verifica del codice fiscale:
 - Dopo aver completato le modifiche, il supervisore clicca su "Avanti".
 - Il client verifica che il codice fiscale sia valido.
 - In caso di codice fiscale non valido, viene mostrato un messaggio di errore: "Controlla che i campi siano compilati correttamente".

- 3 Revisione dei dati:
 - Se il codice fiscale è valido, il supervisore può rivedere i dati aggiornati attraverso una finestra di riepilogo.
 - Se necessario, il supervisore può modificare ulteriormente i dati cliccando su "Rivedi".
- 4 Conferma della modifica:
 - Una volta confermati i dati, il supervisore clicca su "Modifica".
 - Il client invia una richiesta al MainHandler che la inoltra al PatientService per aggiornare le informazioni del paziente.
- 5 Aggiornamento nei database:
 - Il PatientService aggiorna le informazioni anagrafiche nel PatientDB.
 - Successivamente, aggiorna le sessioni nel SessionDB e le associazioni tutor-paziente nel PatientTutorDB.
 - Ogni operazione restituisce un messaggio di successo.
- 6 Conferma della modifica al supervisore:
 - Dopo che tutte le modifiche sono state applicate con successo, il PatientService invia una conferma al client.
 - Il client notifica il supervisore con il messaggio: "Il paziente è stato modificato con successo".

4.7.4 Sequence diagram di progettazione – Report

Di seguito sono illustrati i sequence diagram relativi alle funzionalità più significative gestite dal ReportsService. Questi diagrammi mostrano le principali interazioni tra i vari attori e componenti del sistema, evidenziando il flusso di richieste e risposte.

4.7.4.1 Creazione report

Il seguente sequence diagram descrive il processo di creazione di un nuovo report da parte di un supervisore o tutor. Viene illustrata l'interazione tra l'utente, il client, il ReportsService e il ReportsDB.

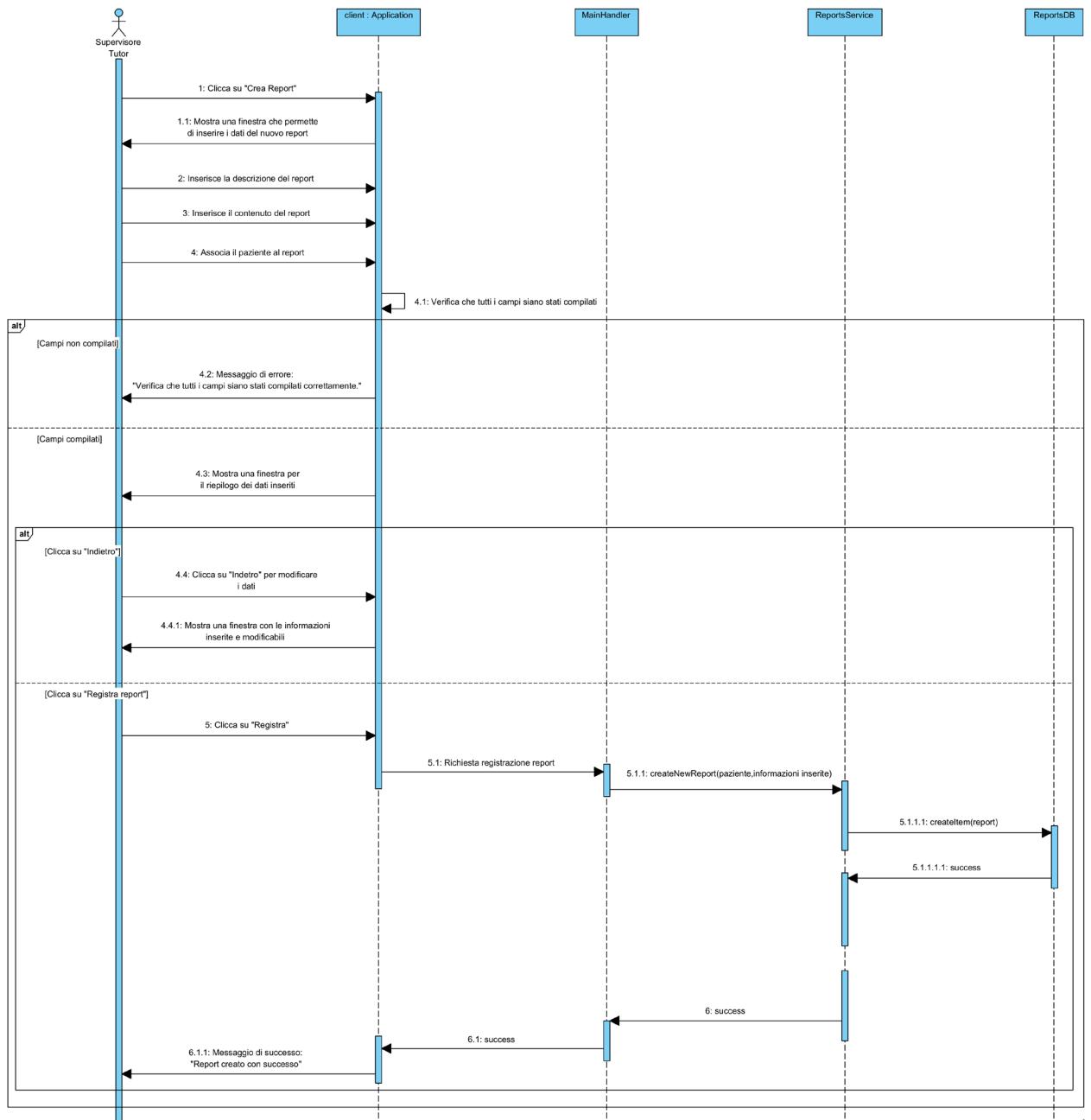


Figura 4.10: Sequence diagram di progettazione – Creazione report.

Il flusso principale si sviluppa nei seguenti passaggi:

1 Avvio della creazione del report:

- Il supervisore o tutor clicca su "Crea Report" nel client.
- Il client mostra una finestra dove è possibile inserire i dettagli del nuovo report, tra cui:
- La descrizione del report.
- Il contenuto del report.
- L'associazione del report a un paziente specifico.

2 Verifica dei campi:

- Una volta inseriti i dati, il client verifica che tutti i campi obbligatori siano stati compilati correttamente.

- Se i campi non sono completi, il sistema restituisce un messaggio di errore: "Verifica che tutti i campi siano stati compilati correttamente".
- In caso di successo, il client mostra una finestra di riepilogo con i dati inseriti.

3 Revisione dei dati:

- L'utente ha la possibilità di tornare indietro cliccando su "Indietro" per modificare i dati inseriti.
- Se clicca su "Indietro", il client mostra nuovamente la finestra di modifica con i dati precedentemente compilati.

4 Registrazione del report:

- Quando i dati sono stati confermati, l'utente clicca su "Registra".
- Il client invia una richiesta al MainHandler che la inoltra al ReportsService per creare il nuovo report.
- Il ReportsService registra il report nel ReportsDB utilizzando l'operazione createItem(report).

5 Conferma del successo:

- Una volta che il report è stato creato con successo nel ReportsDB, il sistema restituisce un messaggio di conferma al client.
- Il client notifica l'utente con il messaggio: "Report creato con successo".

4.7.4.2 Modifica report

Il seguente sequence diagram descrive il processo di modifica delle informazioni di un report da parte di un supervisore o tutor. Viene illustrata l'interazione tra l'utente, il client, il ReportsService e il ReportsDB.

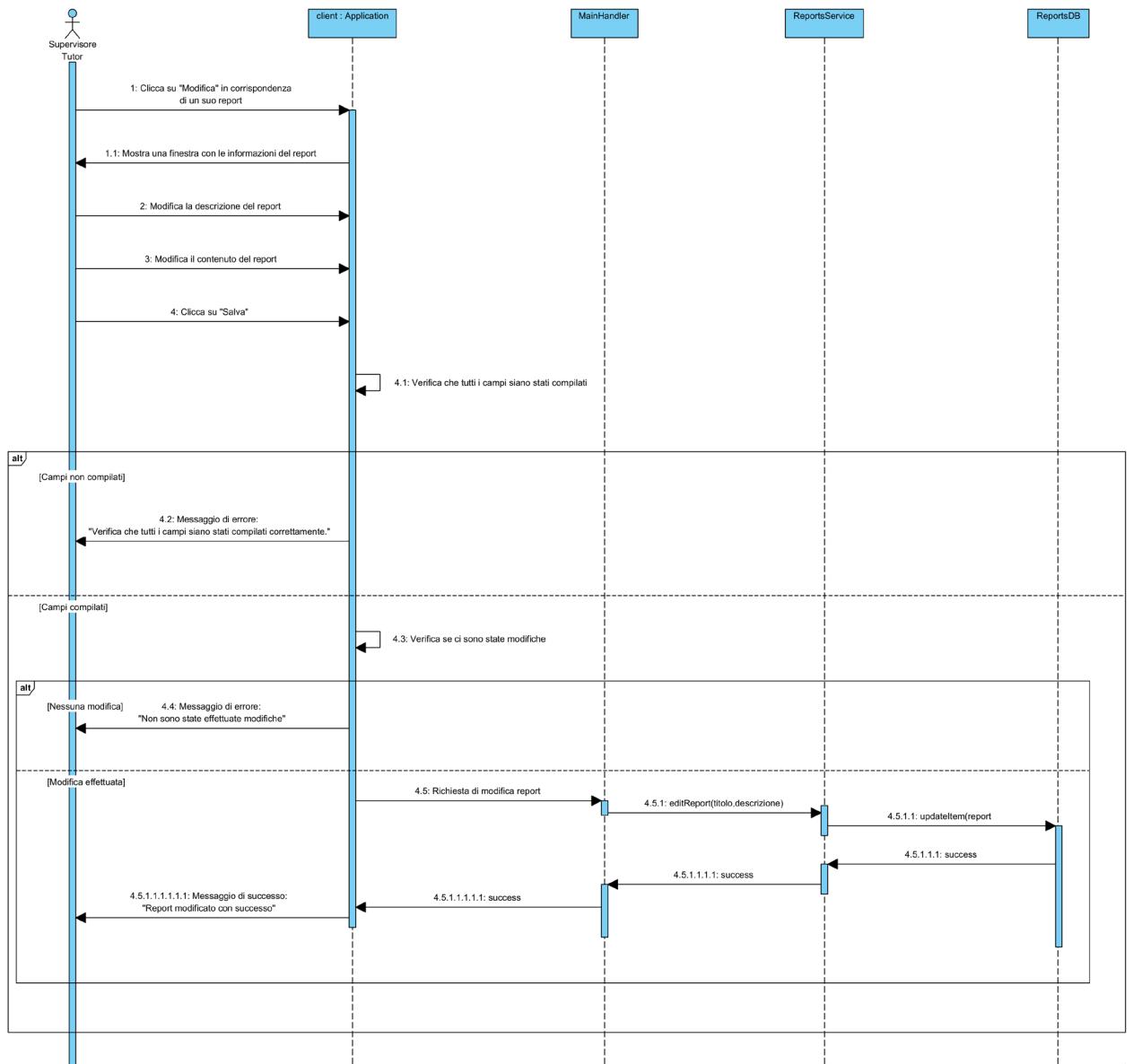


Figura 4.11: Sequence diagram di progettazione – Modifica report.

I flusso principale si articola nei seguenti passaggi:

- 1 Avvio della modifica:
 - Il supervisore o tutor clicca su "Modifica" accanto al report che desidera aggiornare.
 - Il client mostra una finestra con i dettagli correnti del report, permettendo all'utente di modificarne la descrizione e il contenuto.
- 2 Modifica dei dati:
 - L'utente modifica i campi necessari, come la descrizione o il contenuto del report.
 - Una volta completate le modifiche, clicca su "Salva".
- 3 Verifica dei campi compilati:
 - Il client verifica che tutti i campi obbligatori siano stati compilati correttamente.

- Se i campi non sono completi, il sistema restituisce un messaggio di errore: "Verifica che tutti i campi siano stati compilati correttamente".
- 4 Controllo delle modifiche:
- Se i campi sono compilati correttamente, il ReportsService, ottenuta la richiesta dal MainHandler, verifica se sono state effettuate modifiche rispetto ai dati originali.
 - Se non ci sono modifiche, viene mostrato un messaggio di errore: "Non sono state effettuate modifiche".
- 5 Aggiornamento del report:
- Se sono state effettuate modifiche, il ReportsService invia una richiesta al ReportsDB per aggiornare i dati del report.
 - Il database aggiorna le informazioni e restituisce un messaggio di successo.
- 6 Conferma della modifica:
- Una volta completata l'operazione, il sistema restituisce un messaggio al client.
 - Il client notifica l'utente con il messaggio: "Report modificato con successo".

4.7.5 Sequence diagram di progettazione – Attività

Il seguente sequence diagram descrive il processo mediante il quale un supervisore può visualizzare la lista delle attività relative a diversi ambiti del sistema (ad esempio, autenticazione, utenti, report o pazienti). Il flusso di interazione coinvolge il supervisore, il client, il LogsService e il LogsDB.

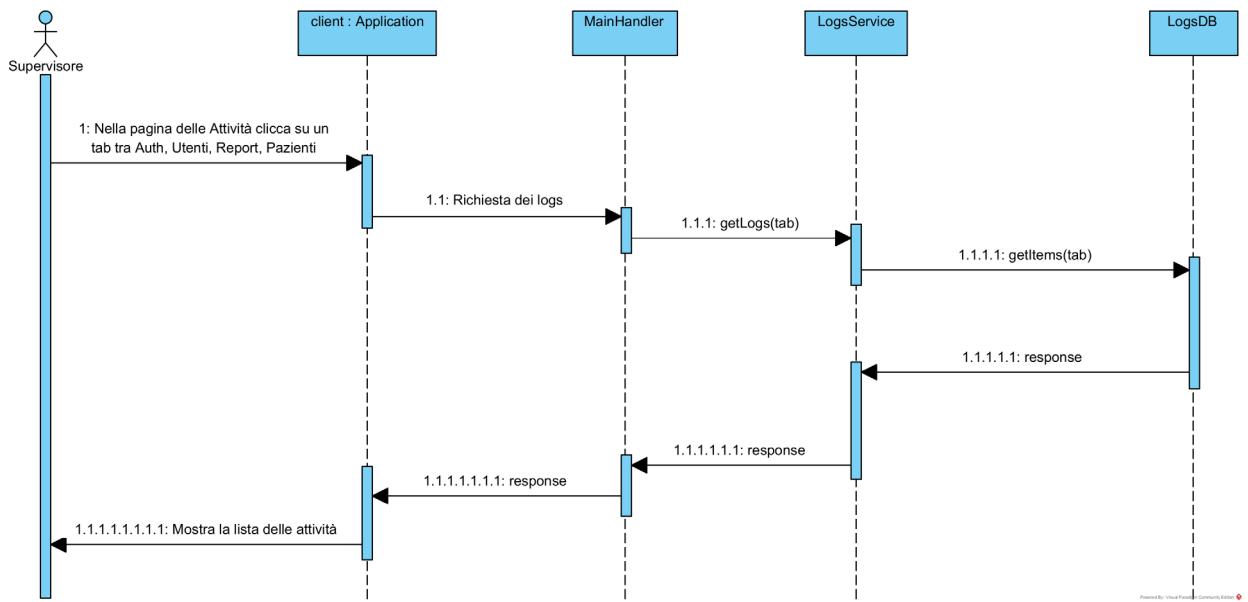


Figura 4.12: Sequence diagram di progettazione – Attività.

I passaggi principali sono i seguenti:

- 1 Richiesta di visualizzazione delle attività:
 - Il supervisore accede alla pagina delle attività e seleziona un tab specifico (ad esempio, Auth, Utenti, Report, Pazienti).
 - Il client invia una richiesta al MainHanler e la inoltra al LogsService specificando il tab selezionato.

- 2 Recupero dei dati dal database:
 - Il LogsService interroga il LogsDB per recuperare le attività corrispondenti al tab selezionato.
 - Il database restituisce i dati richiesti al LogsService.

- 3 Restituzione dei dati al client:
 - Il LogsService invia i dati ricevuti dal database al client.
 - Il client utilizza i dati per mostrare al supervisore la lista delle attività corrispondenti al tab selezionato.

- 4 Visualizzazione delle attività:
 - Il client mostra la lista delle attività al supervisore, che può quindi visualizzare i dettagli relativi alle operazioni del sistema.

4.7.6 Sequence diagram di progettazione – Storage

Il seguente sequence diagram rappresenta il processo attraverso il quale un utente può caricare o aggiornare l'avatar del proprio profilo. Viene illustrata l'interazione tra l'utente, il client, il StorageService, il DataStorage, il UsersService e il UsersDB

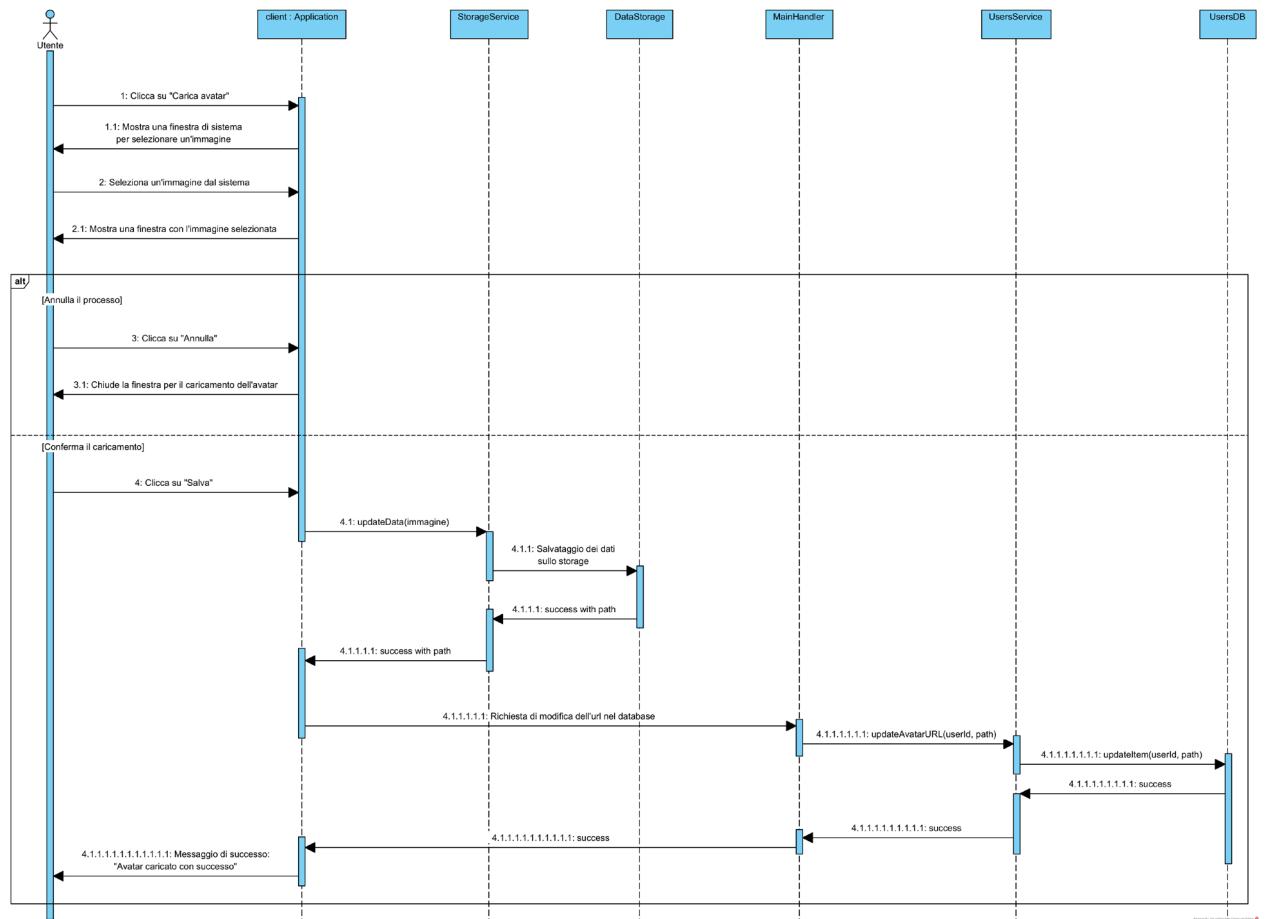


Figura 4.13: Sequence diagram di progettazione – Storage.

Il flusso principale si articola nei seguenti passaggi:

- 1 Avvio del caricamento dell'avatar:
 - L'utente clicca su "Carica avatar" nell'interfaccia client.
 - Il client mostra una finestra di sistema per selezionare un'immagine.
 - L'utente sceglie un'immagine, e il client la visualizza in un'anteprima.
- 2 Scelta dell'operazione:
 - L'utente può annullare il processo cliccando su "Annulla", e il client chiude la finestra senza effettuare alcuna modifica.
 - Se l'utente conferma l'operazione cliccando su "Salva", il client invia la richiesta al StorageService per salvare l'immagine selezionata.
- 3 Salvataggio dell'immagine nello storage:
 - Il StorageService salva l'immagine nel DataStorage.
 - Una volta completata l'operazione, il DataStorage restituisce un percorso (path) che identifica l'immagine salvata.
 - Inoltra il path dell'immagine al client.

- 4 Aggiornamento dell'URL dell'avatar nel database:
 - Il client inoltra il percorso dell'immagine al MainHanlder che la inoltra allo UsersService, richiedendo l'aggiornamento del profilo utente nel UsersDB.
 - Il UsersService aggiorna l'URL dell'avatar nel database con il percorso fornito.
 - Al termine dell'operazione, il UsersService restituisce un messaggio di successo.
- 5 Conferma del caricamento all'utente:
 - Il client notifica l'utente con un messaggio: "Avatar caricato con successo".

4.8 Activity Diagram

Gli Activity Diagram sono strumenti utili per rappresentare in modo chiaro e schematico il flusso delle attività e delle operazioni all'interno del sistema. Questi diagrammi permettono di descrivere il comportamento dinamico del software, evidenziando i passaggi logici necessari per l'esecuzione di una determinata funzione. Attraverso di essi, è possibile modellare i flussi di controllo e i flussi di dati, offrendo una visione dettagliata delle interazioni tra le varie componenti.

Per una migliore organizzazione e leggibilità, è stata adottata la variante Swimlane, che consente di suddividere le attività in corsie separate, ciascuna dedicata a una specifica componente o attore del sistema. Questa scelta aiuta a identificare chiaramente le responsabilità e il ruolo di ogni elemento coinvolto nel flusso. Nei diagrammi sviluppati, le corsie rappresentano, ad esempio, il Client, i vari servizi applicativi e i database correlati, riflettendo in maniera fedele l'architettura del sistema.

4.8.1 Activity Diagram – Autenticazione

Il seguente activity diagram rappresenta il flusso del processo di autenticazione per un utente non autenticato, evidenziando le interazioni tra il Client, il AuthenticationService e il AuthenticationDB.

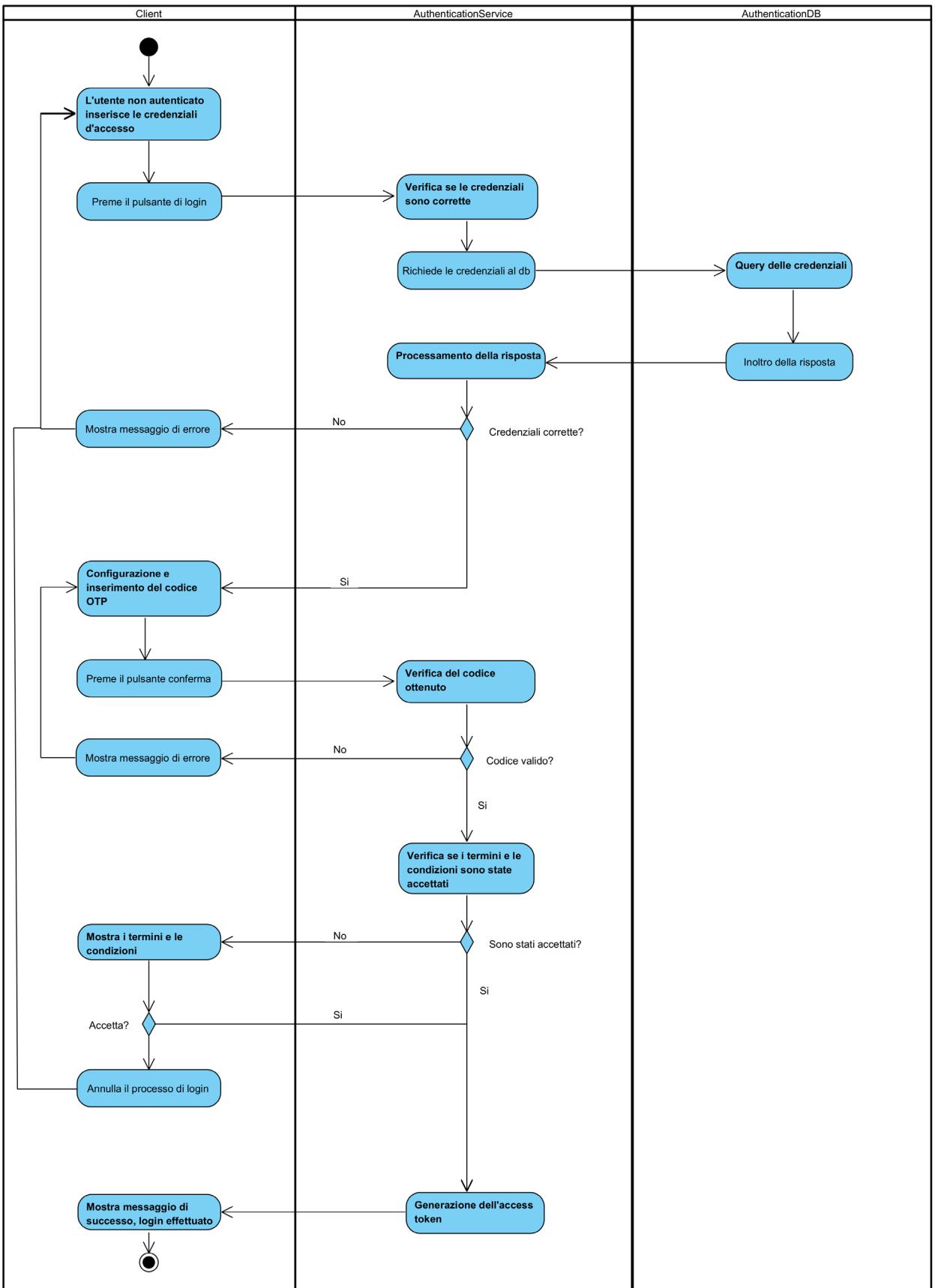


Figura 4.14: Activity Diagram – Autenticazione.

Di seguito viene descritto il flusso passo per passo:

- 1 Inserimento delle credenziali:
 - L'utente non autenticato inserisce le proprie credenziali di accesso (email e password) attraverso l'interfaccia del client.
 - Successivamente, preme il pulsante di login.
- 2 Verifica delle credenziali:
 - Il AuthenticationService riceve le credenziali e verifica la loro correttezza richiedendo le informazioni al AuthenticationDB.
 - Il database esegue una query per verificare l'esistenza e la validità delle credenziali e restituisce una risposta.
- 3 Processamento della risposta:
 - Il AuthenticationService elabora la risposta ricevuta dal database:
 - Credenziali non corrette: Viene mostrato un messaggio di errore sul client, e l'utente viene reindirizzato al punto di partenza per riprovare.
 - Credenziali corrette: Si procede alla fase successiva, che prevede la configurazione o l'inserimento del codice OTP.
- 4 Gestione del codice OTP:
 - Se il codice OTP è necessario, l'utente lo configura e lo inserisce.
 - Dopo aver premuto il pulsante di conferma, il codice viene verificato:
 - Codice non valido: Viene mostrato un messaggio di errore e l'utente deve reinserire il codice.
 - Codice valido: Si procede alla fase successiva.
- 5 Verifica dei termini e delle condizioni:
 - Il sistema verifica se i termini e le condizioni sono stati già accettati:
 - Non accettati: Viene mostrata una finestra con i termini e le condizioni.
 - L'utente può accettare, proseguendo con il login, oppure annullare il processo di autenticazione.
 - Accettati: Si passa direttamente alla generazione del token.
- 6 Generazione dell'access token:
 - Una volta completati tutti i passaggi con successo, il AuthenticationService genera un access token per l'utente.
 - Il client visualizza un messaggio di successo e il login viene considerato completato.

4.8.2 Activity Diagram – Utenti

Gli Activity Diagram sono stati impiegati per rappresentare i flussi operativi relativi alla gestione degli utenti, inclusi i processi di creazione, eliminazione e ottenimento dei dati di un utente. Attraverso questi diagrammi, è possibile visualizzare in modo chiaro e dettagliato il comportamento del sistema durante l'interazione con gli utenti, evidenziando ogni passo logico e decisionale.

4.8.2.1 Registrazione utente

Il seguente activity diagram rappresenta il processo di registrazione di un nuovo utente, evidenziando le interazioni tra il Client, il MainHandler, il UserService, il UsersDB, l'AuthenticationService e l'AuthenticationDB. Il diagramma illustra ogni fase del flusso, dal momento in cui un supervisore avvia la registrazione, fino al completamento dell'operazione.

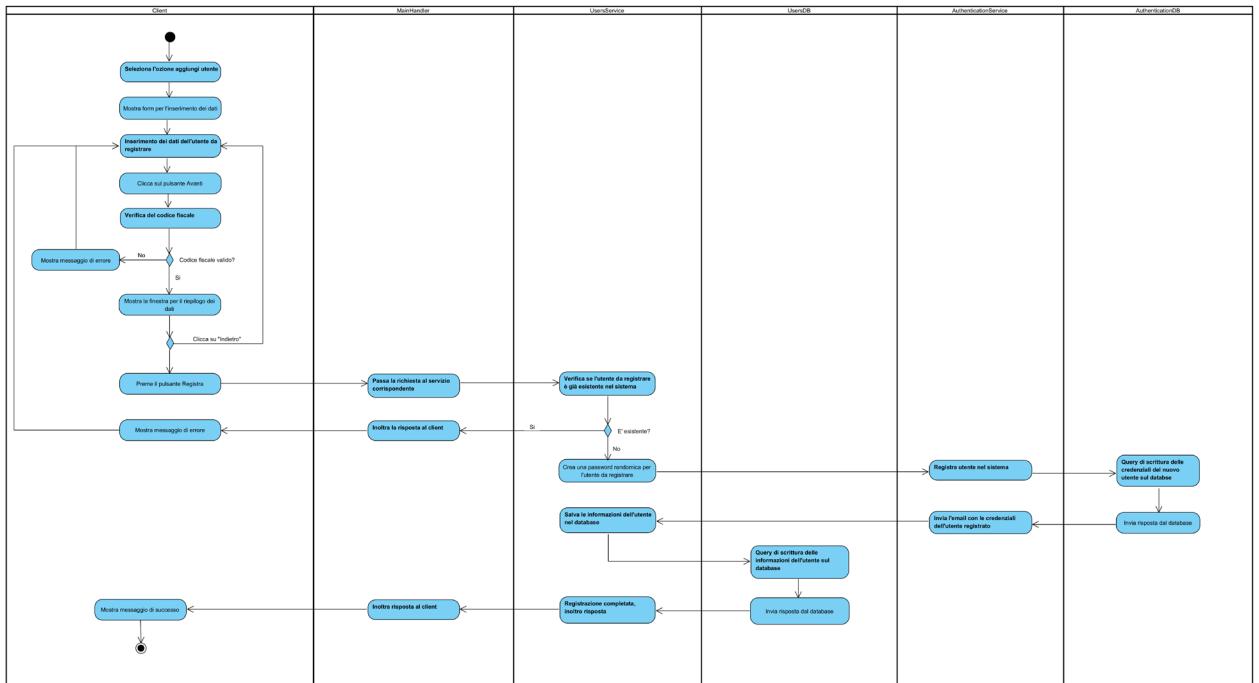


Figura 4.15: Activity Diagram – Registrazione utente.

Avvio della registrazione:

- 1 Il supervisore seleziona l'opzione "Aggiungi utente" sul client.
 - Il client visualizza un form per l'inserimento dei dati dell'utente da registrare (nome, email, codice fiscale, ecc.).
- 2 Inserimento dei dati:
 - Il supervisore compila il form con i dati richiesti e preme il pulsante "Avanti".
- 3 Verifica del codice fiscale:
 - Il client verifica la validità del codice fiscale:
 - Codice non valido: Viene mostrato un messaggio di errore sul client, e il supervisore può correggere i dati e riprovare.
 - Codice valido: Il client mostra una finestra di riepilogo dei dati inseriti.
- 4 Revisione dei dati:
 - Il supervisore può scegliere di tornare indietro cliccando su "Indietro" per modificare i dati, oppure cliccare su "Registra" per procedere con la registrazione.

- 5 Verifica dell'esistenza dell'utente:
 - Il UserService verifica se l'utente esiste già nel sistema interrogando il UsersDB:
 - Utente esistente: Viene mostrato un messaggio di errore, e il supervisore può interrompere il processo.
 - Utente non esistente: Il sistema procede con la creazione del nuovo utente.

- 6 Creazione e salvataggio dell'utente:
 - Il UserService genera una password casuale per il nuovo utente.
 - I dati dell'utente vengono salvati nel UsersDB con una query di scrittura, mentre le credenziali vengono salvate nell'AuthenticationDB tramite l'AuthenticationService.

- 7 Invio delle credenziali via email:
 - Una volta completata la registrazione, il sistema invia un'email all'utente con le credenziali di accesso generate.

- 8 Completamento del processo:
 - Il sistema restituisce un messaggio di successo al client: "Utente registrato con successo".
 - Il processo termina con la registrazione dell'utente nel sistema.

4.8.2.2 Ottenimento utente

Il seguente activity diagram descrive il processo mediante il quale un utente (ad esempio, un supervisore o un tutor) richiede di visualizzare il profilo di un altro utente. Il diagramma evidenzia le interazioni tra il Client, il MainHandler, il UserService, il UsersDB, e il PatientTutorDB.

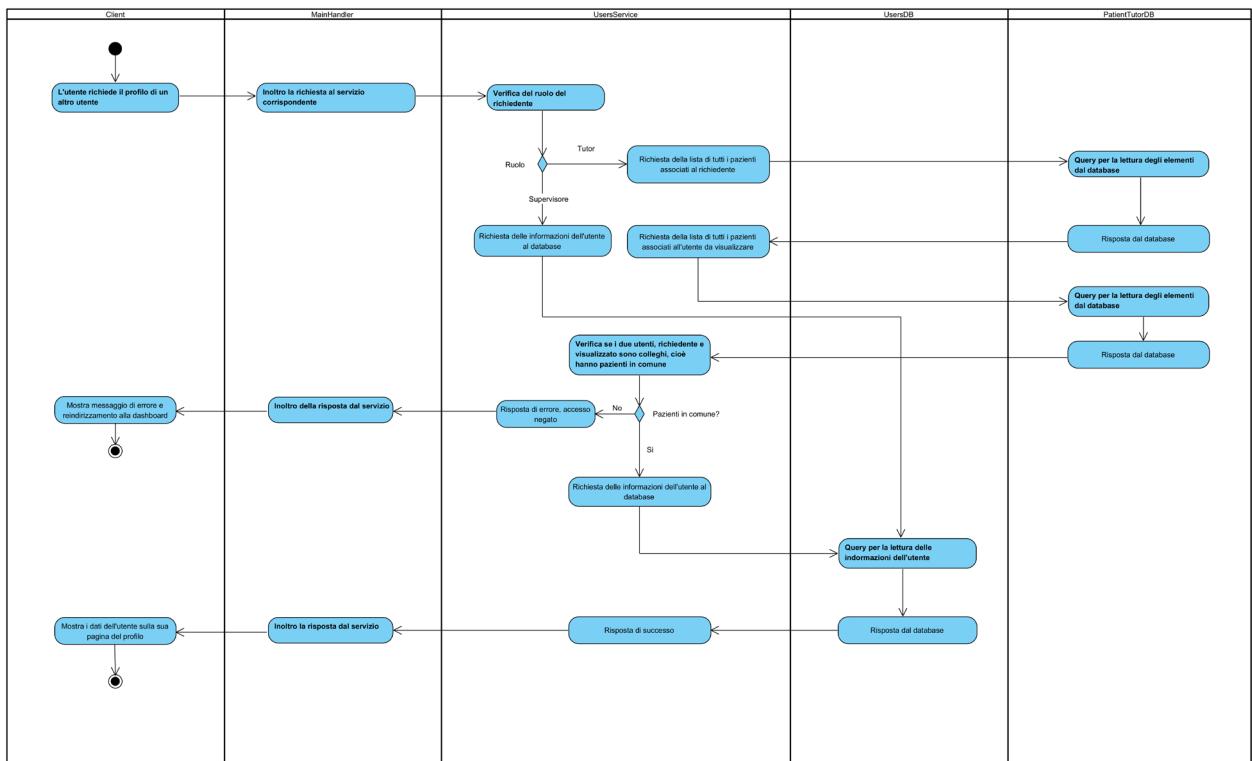


Figura 4.16: Activity Diagram – Ottenimento utente.

Di seguito, il flusso dettagliato delle attività:

- 1 Richiesta del profilo:
 - L'utente avvia il processo richiedendo il profilo di un altro utente tramite il client.
 - La richiesta viene inoltrata al MainHandler, che la instrada al UserService.
- 2 Verifica del ruolo del richiedente:
 - Il UserService verifica il ruolo del richiedente (ad esempio, tutor o supervisore).
 - Tutor: Il sistema richiede la lista di tutti i pazienti associati al tutor al PatientTutorDB.
 - Supervisore: Il sistema richiede direttamente le informazioni dell'utente al UsersDB.
- 3 Verifica della relazione tra gli utenti:
 - Se il richiedente è un tutor, il sistema verifica se esiste una relazione tra il richiedente e l'utente di cui si sta visualizzando il profilo (ad esempio, pazienti in comune).
 - Per effettuare questa verifica, il sistema esegue query al PatientTutorDB e riceve una risposta:
 - No: Se non c'è relazione, viene restituita una risposta di errore con accesso negato e l'utente viene reindirizzato alla dashboard.
 - Sì: Se c'è relazione, il sistema prosegue con la richiesta delle informazioni.
- 4 Ottenimento dei dati dell'utente:
 - Il UserService interroga il UsersDB per recuperare i dati dell'utente richiesto.
 - Il database restituisce le informazioni richieste.
- 5 Risposta al client:
 - Il UserService inoltra la risposta al MainHandler, che la rimanda al client.
 - Il client visualizza le informazioni del profilo dell'utente sulla pagina dedicata.
- 6 Gestione degli errori:
 - In caso di errori (ad esempio, utenti non collegati o profilo non accessibile), il client visualizza un messaggio di errore e reindirizza l'utente alla dashboard.

4.8.2.3 Eliminazione utente

Il seguente activity diagram descrive il processo di eliminazione di un utente dal sistema, evidenziando le interazioni tra il Client, il MainHandler, il UserService, il UsersDB, l'AuthenticationService, e l'AuthenticationDB. Il flusso segue il percorso dall'inizio dell'azione da parte del supervisore fino al completamento dell'eliminazione dell'utente.

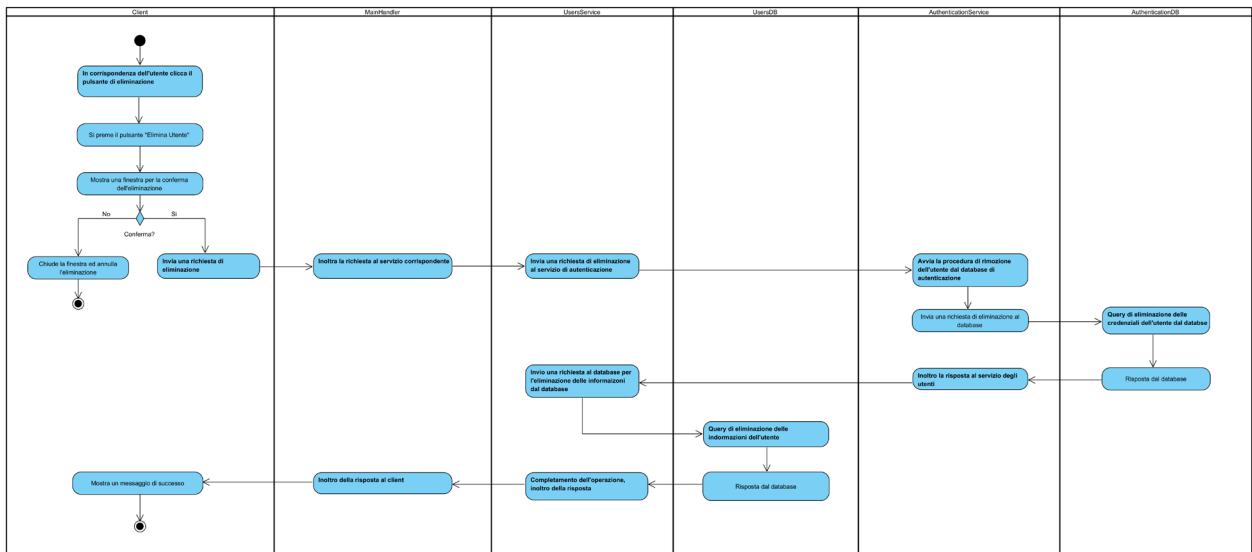


Figura 4.17: Activity Diagram – Eliminazione utente.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio dell'eliminazione:
 - L'utente (supervisore) clicca sul pulsante "Elimina utente" accanto all'utente che desidera rimuovere.
 - Il client mostra una finestra per confermare l'eliminazione.
- 2 Decisione sulla conferma:
 - L'utente può scegliere tra:
 - No: La finestra si chiude e l'operazione viene annullata.
 - Sì: Viene inviata una richiesta al MainHandler per avviare il processo di eliminazione.
- 3 Inoltro della richiesta al servizio:
 - Il MainHandler inoltra la richiesta al UserService, che gestisce il processo di eliminazione.
- 4 Eliminazione delle informazioni utente:
 - Il UserService esegue i seguenti passaggi:
 - Invia una richiesta all'AuthenticationService per eliminare le credenziali dell'utente.
 - L'AuthenticationService invia una query all'AuthenticationDB per rimuovere le credenziali associate.
 - Una volta completata l'eliminazione, il AuthenticationService invia una risposta al UserService.
 - Il UserService invia una query al UsersDB per eliminare le informazioni relative all'utente nel database.
 - Il UsersDB completa l'eliminazione e restituisce una risposta al UserService.

- 5 Completamento dell'operazione:
 - Dopo che tutte le informazioni dell'utente sono state eliminate dai database, il UserService invia una risposta al MainHandler.
 - Il MainHandler inoltra la risposta al client.
- 6 Conferma al supervisore:
 - Il client notifica l'utente con un messaggio di successo: "Utente eliminato con successo".
 - Il flusso termina con l'eliminazione completa dell'utente dal sistema.

4.8.3 Activity Diagram – Pazienti

Gli Activity Diagram sono stati impiegati per rappresentare i flussi operativi relativi alla gestione dei pazienti, inclusi i processi di creazione e modifica dei dati di un paziente. Attraverso questi diagrammi, è possibile visualizzare in modo chiaro e dettagliato il comportamento del sistema durante l'interazione con gli utenti, evidenziando ogni passo logico e decisionale.

4.8.3.1 Creazione paziente

Il seguente activity diagram descrive il processo di creazione di un nuovo paziente, evidenziando i passaggi principali dall'inizio della richiesta da parte del supervisore fino alla registrazione dei dati del paziente nei database. Le interazioni coinvolgono il Client, il MainHandler, il PatientService, e i database associati (PatientDB, SessionDB, e PatientTutorDB).

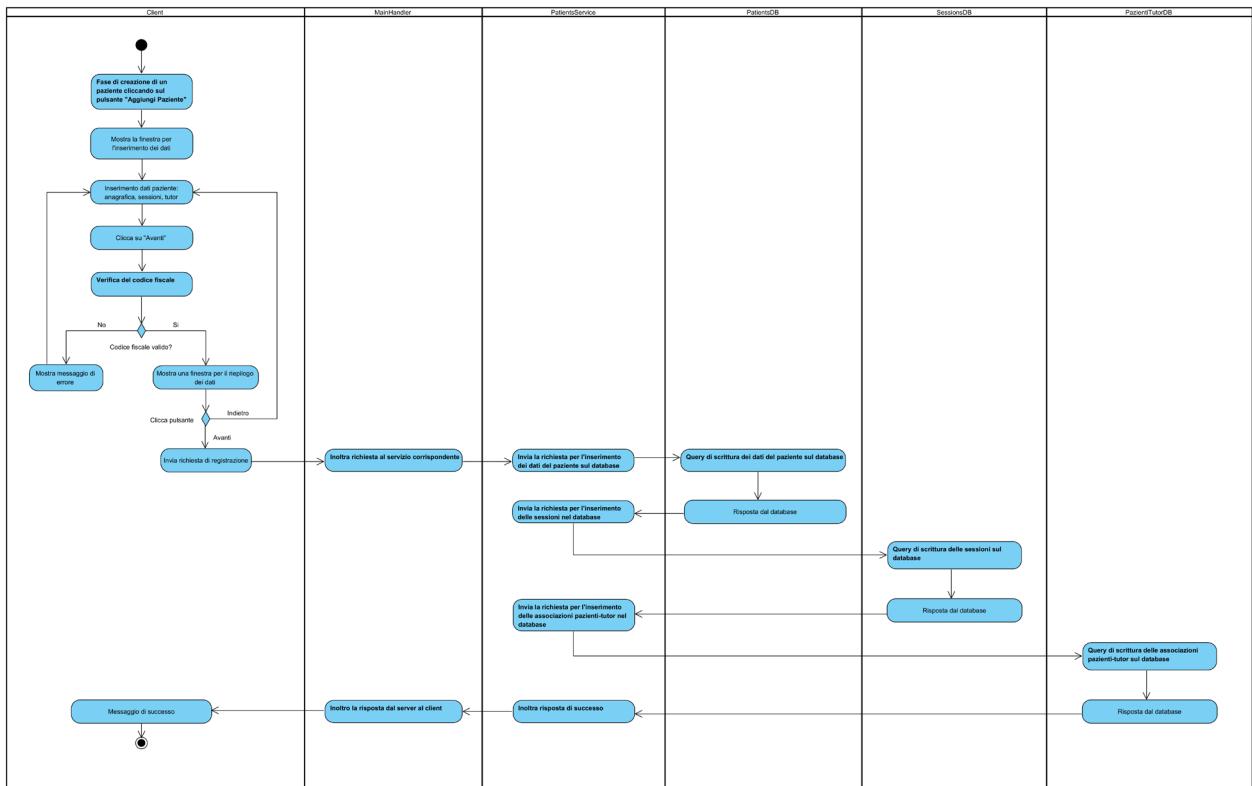


Figura 4.18: Activity Diagram – Creazione paziente.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio della creazione del paziente:
 - L'utente (supervisore) clicca sul pulsante "Aggiungi Paziente" nel client.
 - Il client visualizza una finestra per l'inserimento dei dati del paziente, inclusi:
 - Informazioni anagrafiche.
 - Dati sulle sessioni settimanali.
 - Assegnazione del tutor.
- 2 Inserimento dei dati e verifica:
 - Dopo aver compilato il form, il supervisore clicca su "Avanti".
 - Il sistema effettua una verifica del codice fiscale:
 - Codice fiscale non valido: Mostra un messaggio di errore e l'utente può correggere i dati e riprovare.
 - Codice fiscale valido: Mostra una finestra di riepilogo per consentire al supervisore di controllare i dati inseriti.
- 3 Revisione dei dati:
 - Il supervisore può cliccare su "Indietro" per modificare i dati oppure su "Avanti" per proseguire con la registrazione.
- 4 Invio della richiesta al servizio:
 - Il client invia i dati al MainHandler, che inoltra la richiesta al PatientService.

5 Salvataggio dei dati nei database:

- Il PatientService gestisce le seguenti operazioni:
 - Inserimento dei dati del paziente nel PatientDB: Viene inviata una query per salvare i dati anagrafici del paziente.
 - Inserimento delle sessioni nel SessionDB: Salva i dati relativi alle sessioni settimanali pianificate per il paziente.
 - Inserimento delle associazioni paziente-tutor nel PatientTutorDB: Registra le relazioni tra paziente e tutor.
- Ogni database restituisce una risposta di conferma al PatientService.

6 Conferma del successo:

- Il PatientService invia una risposta al MainHandler, che la inoltra al client.
- Il client notifica al supervisore con un messaggio: "Paziente registrato con successo".

4.8.3.2 Modifica paziente

Il seguente activity diagram rappresenta il processo di modifica delle informazioni di un paziente, evidenziando le interazioni tra il Client, il MainHandler, il PatientService, e i database associati (PatientDB, SessionDB, e PatientTutorDB).

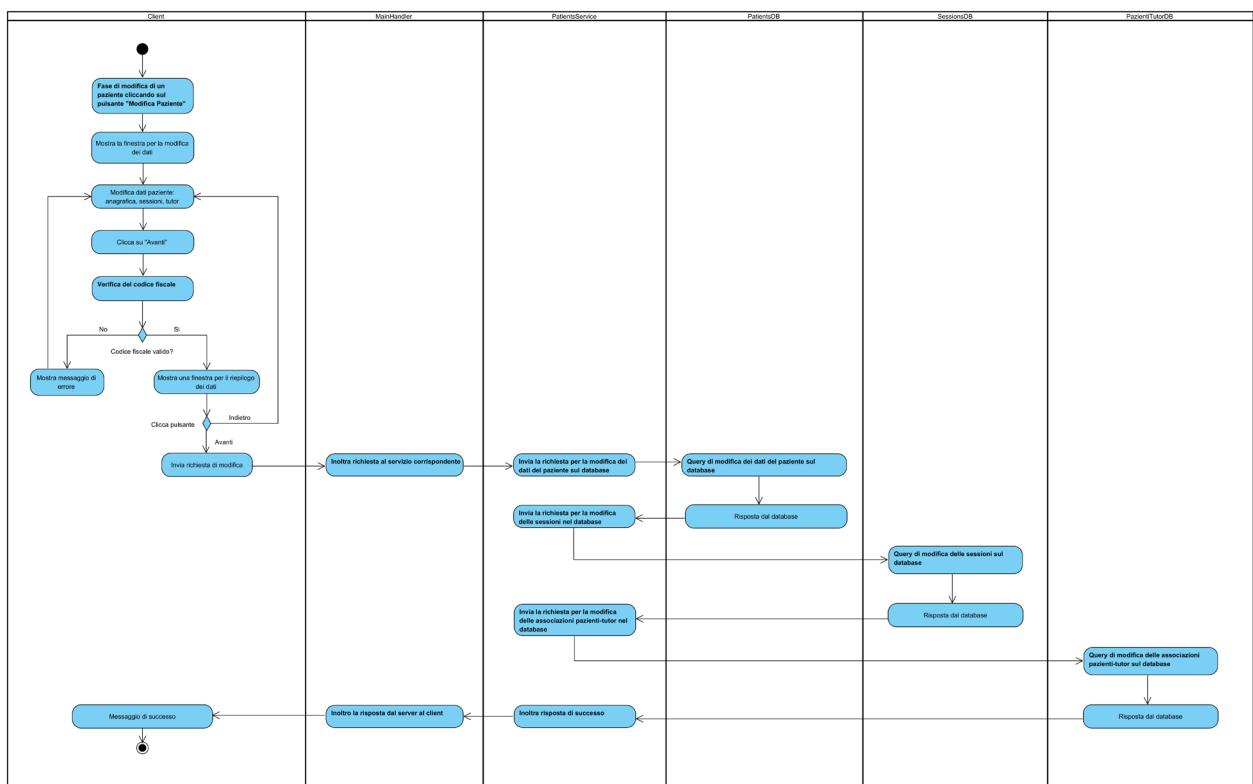


Figura 4.19: Activity Diagram – Modifica paziente.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio della modifica:
 - L'utente (supervisore) clicca sul pulsante "Modifica Paziente" accanto al paziente di interesse.
 - Il client visualizza una finestra che consente di modificare i dati del paziente, come:
 - Informazioni anagrafiche.
 - Dati sulle sessioni settimanali.
 - Assegnazione dei tutor.
- 2 Inserimento dei dati e verifica:
 - Dopo aver aggiornato i dati, il supervisore clicca su "Avanti".
 - Il sistema effettua una verifica del codice fiscale:
 - Codice fiscale non valido: Viene mostrato un messaggio di errore e l'utente può correggere i dati e riprovare.
 - Codice fiscale valido: Viene mostrata una finestra di riepilogo per consentire al supervisore di controllare i dati aggiornati.
- 3 Revisione dei dati:
 - Il supervisore può cliccare su "Indietro" per modificare nuovamente i dati o su "Avanti" per confermare e procedere con la registrazione delle modifiche.
- 4 Invio della richiesta al servizio:
 - Il client invia i dati al MainHandler, che inoltra la richiesta al PatientService.
- 5 Aggiornamento dei dati nei database:
 - Il PatientService gestisce le seguenti operazioni:
 - Modifica dei dati del paziente nel PatientDB: Viene inviata una query per aggiornare i dati anagrafici del paziente.
 - Modifica delle sessioni nel SessionDB: Aggiorna i dati relativi alle sessioni settimanali pianificate.
 - Modifica delle associazioni paziente-tutor nel PatientTutorDB: Aggiorna le relazioni tra paziente e tutor.
 - Ogni database restituisce una risposta di conferma al PatientService.
- 6 Conferma del successo:
 - Il PatientService invia una risposta al MainHandler, che la inoltra al client.
 - Il client notifica al supervisore con un messaggio: "Paziente modificato con successo".

4.8.4 Activity Diagram – Report

Gli Activity Diagram sono stati impiegati per rappresentare i flussi operativi relativi alla gestione deireport, inclusi i processi di creazione e modifica dei dati di un report. Attraverso questi diagrammi, è possibile visualizzare in modo chiaro e dettagliato il comportamento del sistema durante l'interazione con gli utenti, evidenziando ogni passo logico e decisionale.

4.8.4.1 Creazione report

Il seguente activity diagram descrive il processo di creazione di un nuovo report, evidenziando le interazioni tra il Client, il MainHandler, il ReportService, e il ReportDB.

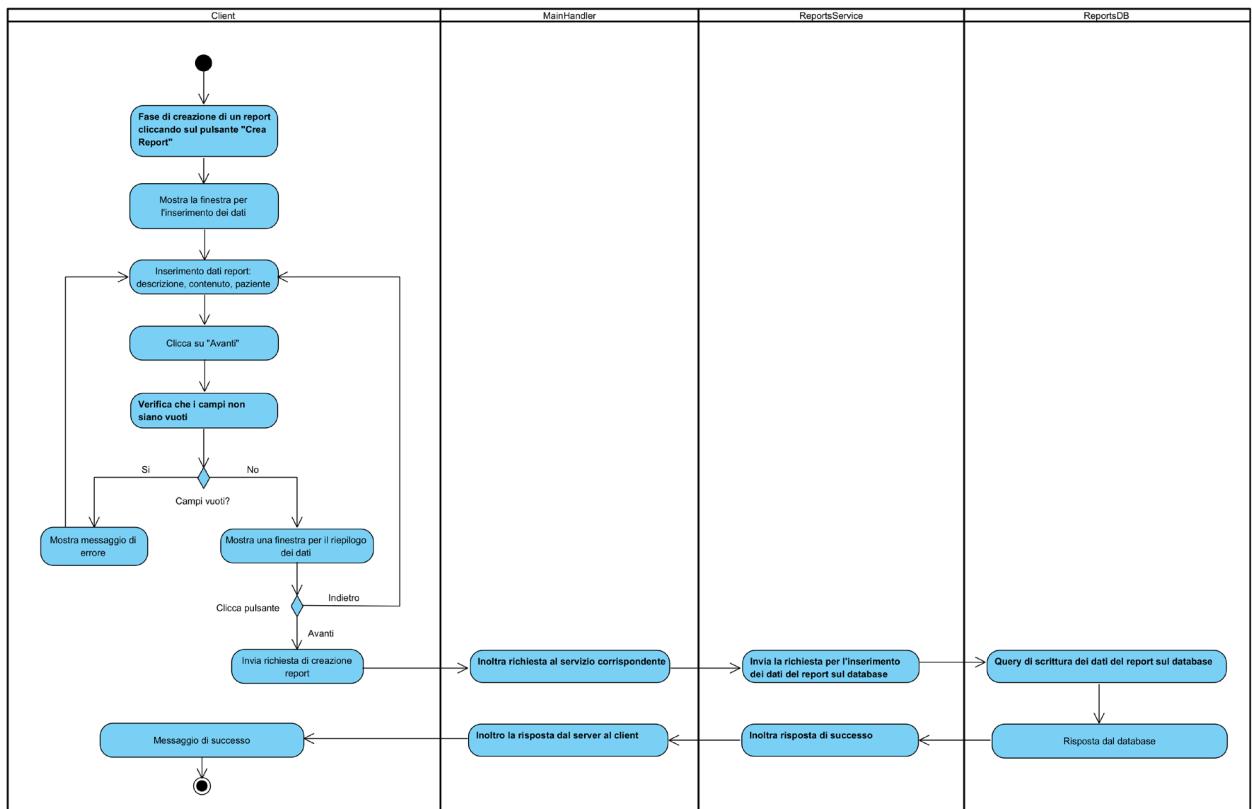


Figura 4.20: Activity Diagram – Creazione report.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio della creazione del report:
 - L'utente (supervisore o tutor) clicca sul pulsante "Crea Report" nel client.
 - Il client visualizza una finestra che permette di inserire i dati necessari per il report, inclusi:
 - Descrizione del report.
 - Contenuto dettagliato.
 - Associazione con un paziente.
- 2 Inserimento dei dati e verifica:
 - Dopo aver compilato i dati richiesti, l'utente clicca su "Avanti".
 - Il sistema verifica che tutti i campi siano stati compilati:
 - Campi vuoti: Viene mostrato un messaggio di errore e l'utente può correggere i dati e riprovare.
 - Campi completi: Viene mostrata una finestra di riepilogo per consentire all'utente di controllare le informazioni inserite.

- 3 Revisione dei dati:
 - L'utente può scegliere di cliccare su "Indietro" per modificare i dati oppure su "Avanti" per confermare la creazione del report.
- 4 Invio della richiesta al servizio:
 - Il client invia la richiesta di creazione al MainHandler, che la inoltra al ReportService.
- 5 Inserimento dei dati nel database:
 - Il ReportService invia una richiesta al ReportDB per inserire i dati del nuovo report.
 - Il database esegue una query di scrittura per registrare le informazioni e restituisce una risposta di successo.
- 6 Conferma del successo:
 - Il ReportService invia una risposta al MainHandler, che la inoltra al client.
 - Il client notifica all'utente che il report è stato creato con successo tramite un messaggio.

4.8.4.2 Modifica report

Il seguente activity diagram descrive il processo di modifica di un report esistente, evidenziando le interazioni tra il Client, il MainHandler, il ReportService, e il ReportDB.

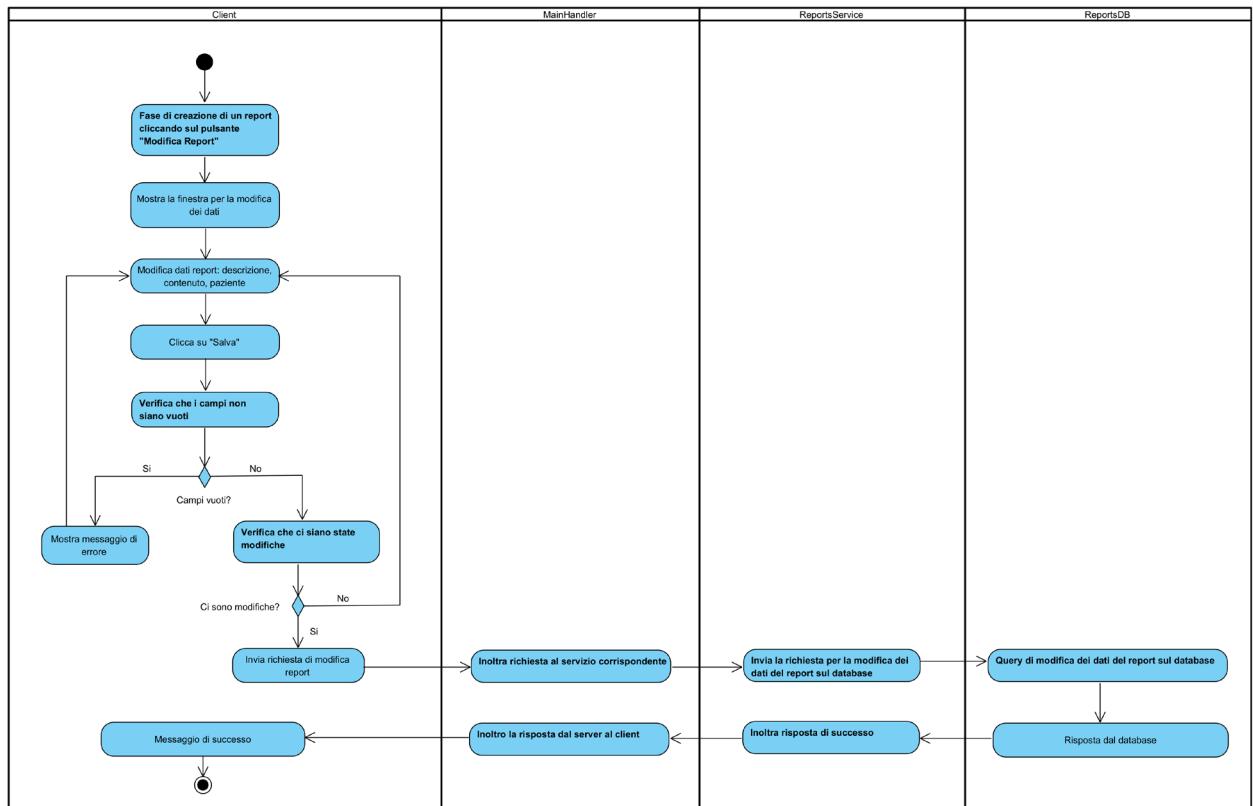


Figura 4.21: Activity Diagram – Modifica report.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio della modifica del report:
 - L'utente (supervisore o tutor) clicca sul pulsante "Modifica Report" accanto al report da aggiornare.
 - Il client mostra una finestra che permette di modificare i dati del report, inclusi:
 - Descrizione del report.
 - Contenuto dettagliato.
 - Associazione con un paziente.
- 2 Inserimento dei dati e verifica:
 - Dopo aver modificato i dati richiesti, l'utente clicca su "Salva".
 - Il sistema verifica che tutti i campi siano stati compilati:
 - Campi vuoti: Viene mostrato un messaggio di errore e l'utente può correggere i dati e riprovare.
 - Campi completi: Si procede alla fase successiva.
- 3 Controllo delle modifiche:
 - Il sistema verifica se ci sono state effettivamente modifiche rispetto ai dati precedenti:
 - Nessuna modifica: Mostra un messaggio di errore che segnala la mancanza di cambiamenti.
 - Modifiche rilevate: Si procede con l'aggiornamento dei dati.
- 4 Invio della richiesta di modifica al servizio:
 - Il client invia la richiesta di modifica al MainHandler, che la inoltra al ReportService.
- 5 Aggiornamento dei dati nel database:
 - Il ReportService invia una richiesta al ReportDB per aggiornare i dati del report esistente.
 - Il database esegue una query di modifica per aggiornare le informazioni e restituisce una risposta di successo.
- 6 Conferma del successo:
 - Il ReportService invia una risposta al MainHandler, che la inoltra al client.
 - Il client notifica all'utente che il report è stato aggiornato con successo tramite un messaggio.

4.8.5 Activity Diagram – Attività

Il seguente activity diagram descrive il processo di consultazione dei log relativi alle attività del sistema, evidenziando le interazioni tra il Client, il MainHandler, il LogsService, e il LogsDB. Il flusso mostra come un utente selezioni una categoria di log e come il sistema restituisca i dati richiesti.

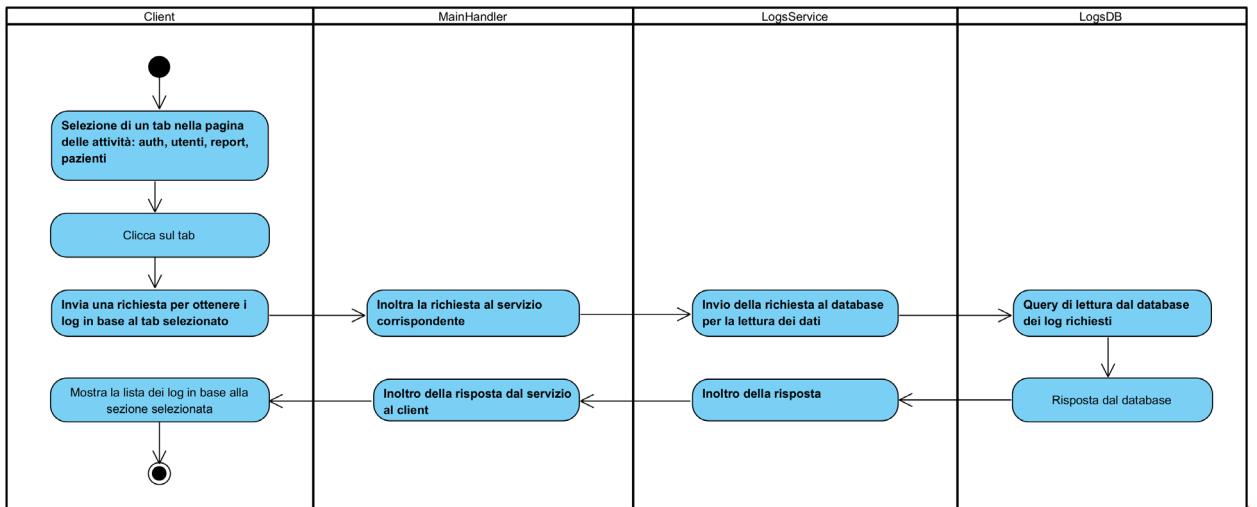


Figura 4.22: Activity Diagram – Attività.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio della consultazione dei log:
 - L'utente accede alla pagina delle attività sul client, che presenta diverse categorie di log:
 - Auth: Log relativi all'autenticazione.
 - Utenti: Log relativi alle operazioni sugli utenti.
 - Report: Log relativi ai report.
 - Pazienti: Log relativi ai pazienti.
 - L'utente clicca sul tab corrispondente alla categoria di log che desidera consultare.
- 2 Invio della richiesta:
 - Il client invia una richiesta al MainHandler per ottenere i log della categoria selezionata.
- 3 Elaborazione della richiesta:
 - Il MainHandler inoltra la richiesta al LogsService, che si occupa di gestire l'accesso ai dati di log.
- 4 Accesso ai dati nel database:
 - Il LogsService invia una richiesta al LogsDB per leggere i log della categoria richiesta.
 - Il LogsDB esegue una query per recuperare i dati corrispondenti e restituisce la risposta al LogsService.
- 5 Inoltro della risposta:
 - Il LogsService inoltra i dati al MainHandler, che li restituisce al client.
- 6 Visualizzazione dei log:
 - Il client mostra la lista dei log in base alla categoria selezionata, completando così il processo.

4.8.6 Activity Diagram – Storage

Il seguente activity diagram descrive il processo di caricamento e aggiornamento dell'avatar di un utente, evidenziando le interazioni tra il Client, il StorageService, il DataStorage, il MainHandler, il UsersService, e il UsersDB.

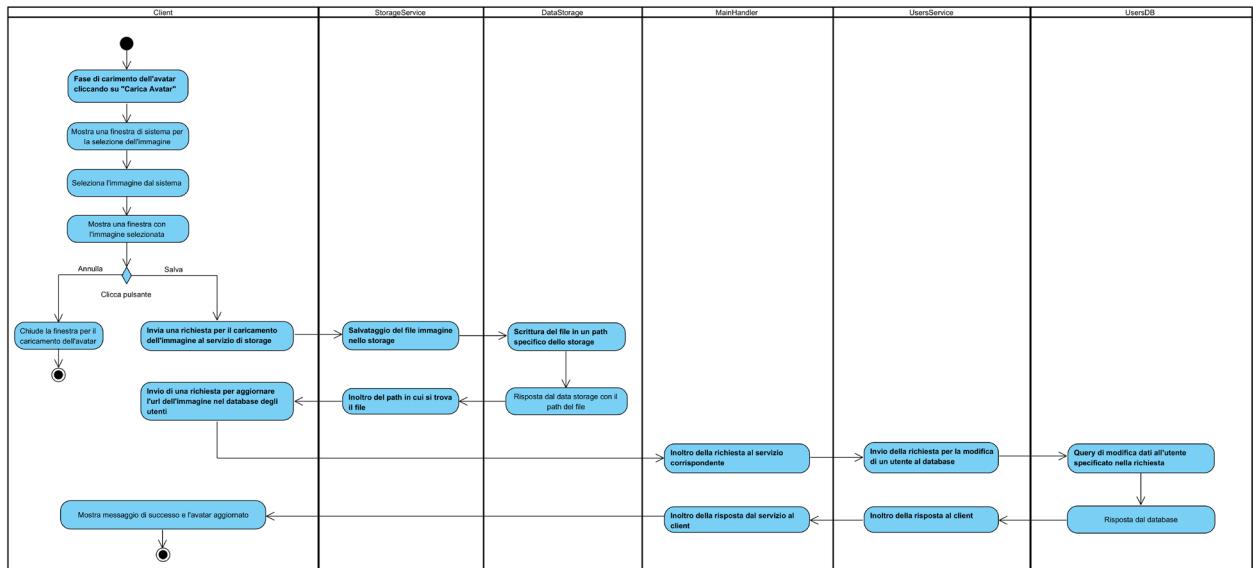


Figura 4.23: Activity Diagram – Storage.

Di seguito, il flusso dettagliato delle attività:

- 1 Avvio del caricamento dell'avatar:
 - L'utente accede alla funzionalità di caricamento dell'avatar cliccando sul pulsante "Carica Avatar" nel client.
 - Il sistema mostra una finestra per la selezione dell'immagine dal file system.
- 2 Selezione dell'immagine:
 - L'utente seleziona un file immagine dal proprio dispositivo.
 - Una finestra di anteprima viene visualizzata per mostrare l'immagine selezionata.
 - L'utente può decidere se:
 - Annullare il caricamento: Chiude la finestra senza salvare alcuna modifica.
 - Salvare l'immagine: Procede con il caricamento.
- 3 Invio della richiesta al servizio di storage:
 - Il client invia una richiesta al StorageService per caricare l'immagine selezionata.
 - Il StorageService salva il file nello storage attraverso il DataStorage:
 - Il file viene scritto in un path specifico.
 - Il DataStorage restituisce il percorso (path) del file salvato.
- 4 Aggiornamento del database degli utenti:
 - Il StorageService inoltra il percorso del file al MainHandler, che lo passa al UsersService.

- Il UserService invia una richiesta al UsersDB per aggiornare l'URL dell'avatar dell'utente con il nuovo path.
- Il UsersDB esegue la query di aggiornamento e restituisce una risposta di successo.

5 Conferma del successo:

- Il UserService inoltra la risposta al MainHandler, che la restituisce al client.
- Il client mostra un messaggio di successo, indicando che l'avatar è stato aggiornato correttamente.

4.9 Communication Diagram

I Communication Diagram vengono utilizzati per rappresentare le interazioni tra i vari attori e le funzionalità del sistema. Questo tipo di diagramma mette in evidenza i partecipanti coinvolti e la sequenza dei messaggi scambiati tra loro. Ogni partecipante può rappresentare un elemento chiave all'interno di un caso d'uso specifico. Per questa documentazione, sono stati realizzati i Communication Diagram relativi ai casi d'uso più significativi e prioritari, al fine di illustrare con chiarezza i flussi comunicativi principali del sistema.

4.9.1 Communication Diagram – Autenticazione

Il seguente Communication Diagram descrive il flusso di comunicazione tra un utente non autenticato e le componenti principali coinvolte nel processo di autenticazione. Il diagramma illustra le interazioni e i messaggi scambiati durante la verifica delle credenziali, l'inserimento del codice OTP e l'accettazione dei termini e condizioni.

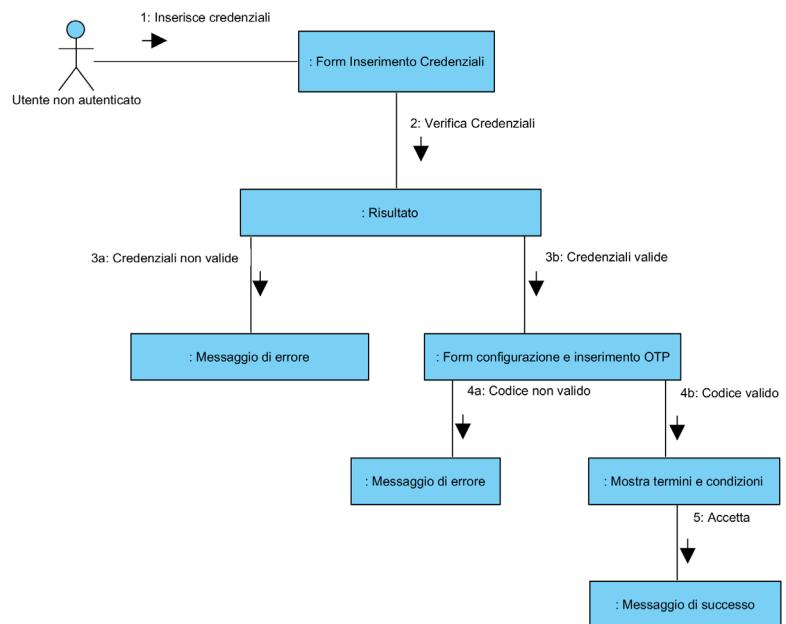


Figura 4.24: Communication Diagram – Autenticazione.

1 Inserimento delle credenziali:

- L'utente non autenticato invia il primo messaggio inserendo le credenziali nella Form Inserimento Credenziali.

- 2 Verifica delle credenziali:
 - La Form Inserimento Credenziali invia una richiesta al sistema per verificare la validità delle credenziali.
 - Il risultato della verifica conduce a due possibili percorsi:
 - Credenziali non valide (3a): Viene mostrato un Messaggio di errore all'utente.
 - Credenziali valide (3b): L'utente viene indirizzato al Form configurazione e inserimento OTP.
- 3 Gestione dell'OTP:
 - L'utente interagisce con il Form configurazione e inserimento OTP inserendo un codice OTP:
 - Codice non valido (4a): Viene mostrato un Messaggio di errore, e l'utente può riprovare.
 - Codice valido (4b): L'utente procede al passaggio successivo.
- 4 Termini e condizioni:
 - Dopo l'inserimento corretto dell'OTP, il sistema mostra i Termini e condizioni.
 - L'utente può scegliere di accettare i termini, inviando il messaggio corrispondente.
- 5 Conferma del successo:
 - Una volta completati tutti i passaggi correttamente, viene mostrato un Messaggio di successo che conferma l'avvenuta autenticazione.

4.9.2 Communication Diagram – Utenti

I seguenti Communication Diagram sono stati utilizzati per rappresentare le interazioni principali relative alla gestione degli utenti, includendo i processi di creazione, modifica e ottenimento delle informazioni di un utente. Questi diagrammi permettono di evidenziare in maniera chiara e dettagliata i partecipanti coinvolti e la sequenza di messaggi scambiati, offrendo una visione completa delle comunicazioni tra le diverse componenti del sistema e gli utenti.

4.9.2.1 Registrazione utente

Il seguente Communication Diagram illustra le interazioni principali tra il supervisore e le componenti coinvolte nel processo di registrazione di un nuovo utente, includendo la verifica

del codice fiscale, il riepilogo dei dati inseriti e il controllo di esistenza dell'utente.

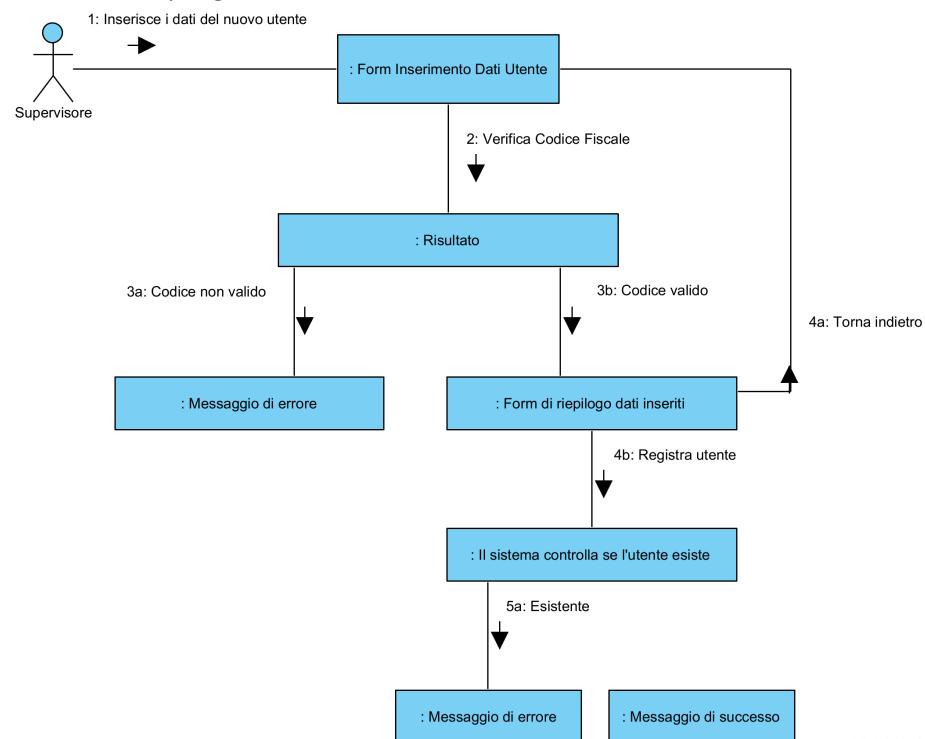


Figura 4.25: Communication Diagram – Registrazione utente.

1 Inserimento dei dati del nuovo utente:

- Il supervisore inizia il processo compilando i campi nella Form Inserimento Dati Utente con le informazioni del nuovo utente.

2 Verifica del codice fiscale:

- La Form Inserimento Dati Utente invia una richiesta per verificare la validità del codice fiscale inserito.
- In base al risultato:
 - Codice non valido (3a): Viene mostrato un Messaggio di errore, e il supervisore può correggere i dati.
 - Codice valido (3b): Si passa alla fase successiva.

3 Riepilogo dei dati inseriti:

- La Form di riepilogo dati inseriti mostra le informazioni compilate, consentendo al supervisore di confermare o tornare indietro per modificare i dati (4a).

4 Registrazione dell'utente:

- Dopo la conferma, la Form di riepilogo dati inseriti invia una richiesta per registrare l'utente.

5 Controllo di esistenza dell'utente:

- Il sistema verifica se l'utente da registrare è già presente:
 - Utente già esistente (5a): Viene mostrato un Messaggio di errore.
 - Utente non esistente: Il processo di registrazione viene completato e il supervisore riceve un Messaggio di successo.

4.9.2.2 Ottenimento utente

Il seguente Communication Diagram rappresenta il flusso di interazione tra il supervisore o tutor e il sistema per ottenere il profilo di un utente. Il diagramma descrive il controllo dei permessi e le relative azioni in base al risultato.

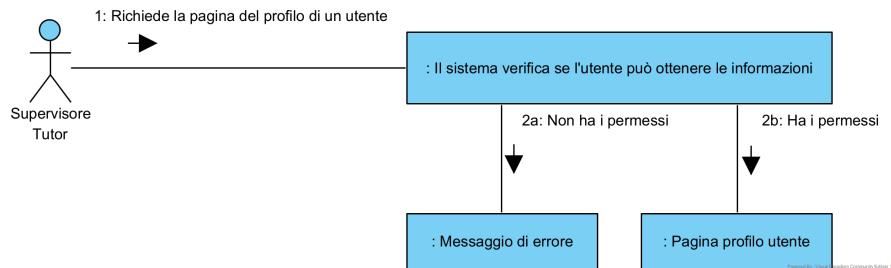


Figura 4.26: Communication Diagram – Ottenimento utente.

1 Richiesta del profilo utente:

- Il supervisore o tutor invia una richiesta al sistema per visualizzare il profilo di un utente.

2 Verifica dei permessi:

- Il sistema verifica se l'utente richiedente ha i permessi per accedere alle informazioni del profilo richiesto.
- Due scenari sono possibili:
 - Non ha i permessi (2a):
 - Il sistema mostra un Messaggio di errore, impedendo l'accesso.
 - Ha i permessi (2b):
 - Il sistema mostra la Pagina del profilo utente con le informazioni richieste.

4.9.2.3 Eliminazione utente

Il seguente Communication Diagram rappresenta il processo di eliminazione di un utente dal sistema da parte del supervisore. Mostra chiaramente i passaggi decisionali e le interazioni tra le componenti coinvolte.

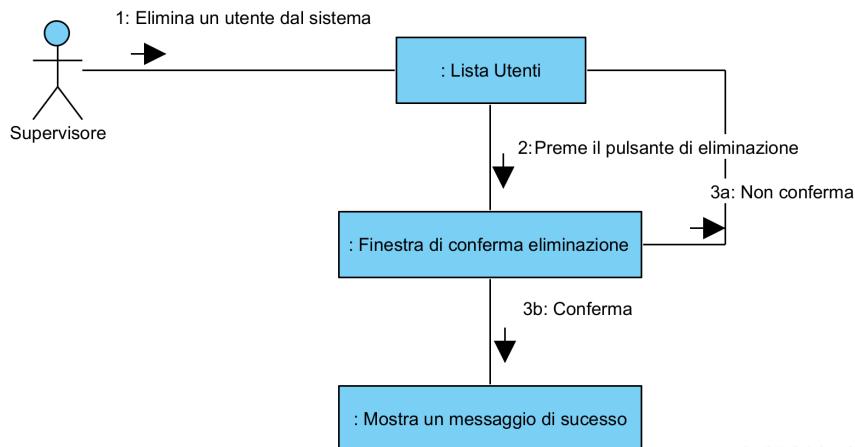


Figura 4.27: Communication Diagram – Eliminazione utente.

- 1 Avvio dell'eliminazione:
 - Il supervisore seleziona un utente dalla Lista Utenti e avvia la procedura di eliminazione premendo il pulsante corrispondente.
- 2 Finestra di conferma eliminazione:
 - Viene visualizzata una finestra di conferma per assicurarsi che il supervisore intenda procedere con l'operazione.
- 3 Decisione del supervisore:
 - Due possibilità:
 - Non conferma (3a):
 - Il supervisore annulla l'operazione, e il processo termina senza modifiche.
 - Conferma (3b):
 - Il supervisore conferma l'eliminazione, e il sistema procede.
- 4 Conferma dell'eliminazione:
 - Dopo la conferma, il sistema rimuove l'utente e mostra un Messaggio di successo al supervisore, indicando che l'operazione è stata completata.

4.9.3 Communication Diagram – Pazienti

I seguenti Communication Diagram sono stati utilizzati per rappresentare le interazioni principali relative alla gestione dei pazienti, includendo i processi di creazione e modifica delle informazioni di un paziente. Attraverso questi diagrammi è possibile identificare chiaramente i partecipanti coinvolti e la sequenza dei messaggi scambiati. Essi forniscono una visione dettagliata delle comunicazioni tra le diverse componenti del sistema, evidenziando il flusso di informazioni e le decisioni prese durante l'interazione con gli utenti.

4.9.3.1 Creazione paziente

Il seguente Communication Diagram relativo alla creazione di un paziente descrive le interazioni principali che avvengono durante questo processo.

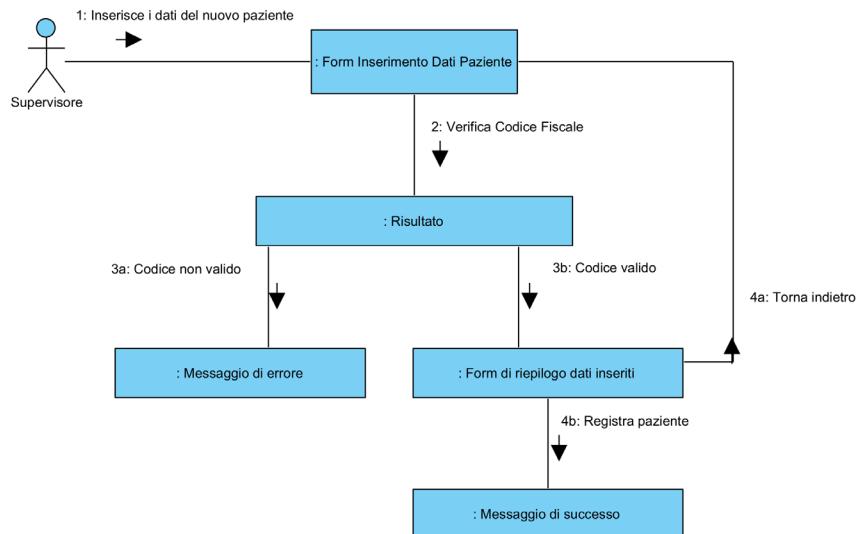


Figura 4.28: Communication Diagram – Creazione paziente.

- 1 Il Supervisore avvia il processo inserendo i dati del nuovo paziente attraverso il form Inserimento Dati Paziente.
- 2 Il form invia una richiesta per verificare il Codice Fiscale del paziente. Questo passaggio è cruciale per garantire la validità dei dati.
- 3 Il sistema produce un Risultato:
 - Se il codice non è valido (3a), viene mostrato un Messaggio di errore informando il supervisore dell'anomalia.
 - Se il codice è valido (3b), viene visualizzato un form di riepilogo dati inseriti, consentendo al supervisore di confermare o modificare i dati.
- 4 A questo punto:
 - Se necessario, il supervisore può tornare indietro per correggere eventuali errori nei dati (4a).
 - In caso di conferma, il supervisore seleziona l'opzione di Registra paziente (4b), e il sistema procede con la registrazione del paziente.
- 5 Dopo la registrazione, viene mostrato un Messaggio di successo, confermando il completamento dell'operazione.

4.9.3.2 Modifica paziente

Il seguente Communication Diagram relativo alla modifica dei dati di un paziente evidenzia le interazioni principali necessarie per portare a termine il processo.

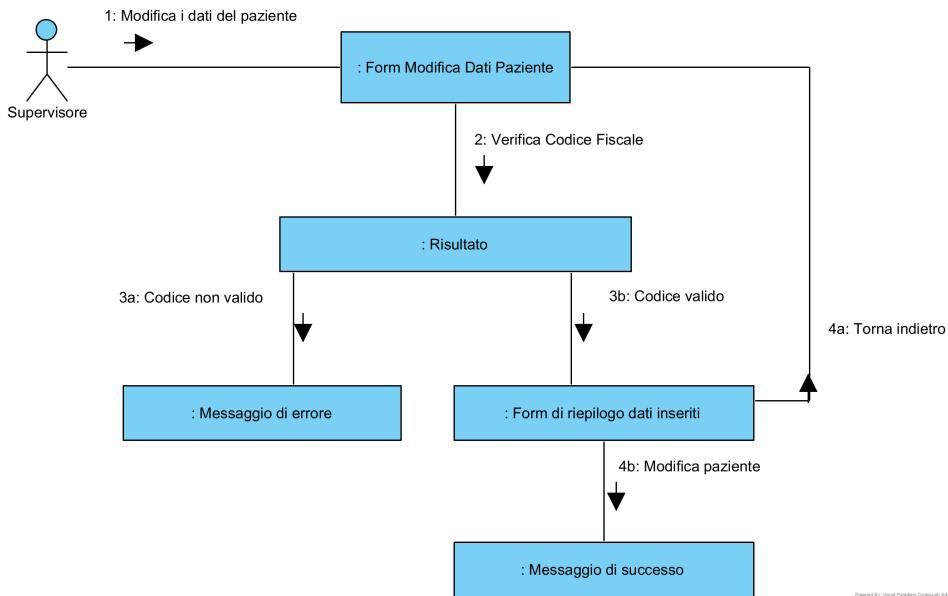


Figura 4.29: Communication Diagram – Modifica paziente.

- 1 Il Supervisore inizia l'operazione accedendo alla Form Modifica Dati Paziente, dove inserisce i nuovi dati del paziente.
- 2 La form avvia una verifica del Codice Fiscale per accertarsi che sia corretto e valido.
- 3 Una volta completata la verifica:
 - Se il codice risulta non valido (3a), il sistema restituisce un Messaggio di errore, informando il supervisore dell'incongruenza.
 - Se il codice è valido (3b), il sistema mostra un Form di riepilogo dati inseriti, consentendo al supervisore di rivedere e confermare i dati aggiornati.
- 4 Il supervisore ha la possibilità di:
 - Tornare indietro per correggere eventuali errori (4a).
 - Procedere con la modifica selezionando l'opzione Modifica paziente (4b), con la quale il sistema applica le modifiche richieste.
- 5 Al termine del processo, il sistema mostra un Messaggio di successo, confermando che la modifica è stata effettuata correttamente.

4.9.4 Communication Diagram – Report

I seguenti Communication Diagram sono stati utilizzati per rappresentare le principali interazioni legate alla gestione dei report, inclusi i processi di creazione e modifica delle informazioni di un report. Attraverso questi diagrammi è possibile delineare con chiarezza i partecipanti coinvolti e la sequenza dei messaggi scambiati tra loro. Questi strumenti offrono una rappresentazione dettagliata delle comunicazioni tra le varie componenti del sistema, mettendo in evidenza il flusso di informazioni e le decisioni prese durante le interazioni con gli utenti.

4.9.4.1 Creazione report

Il seguente Communication Diagram illustra il flusso di interazione per la creazione di un report. Il processo inizia con il supervisore o il tutor che inserisce i dati del report attraverso il modulo di inserimento dedicato.

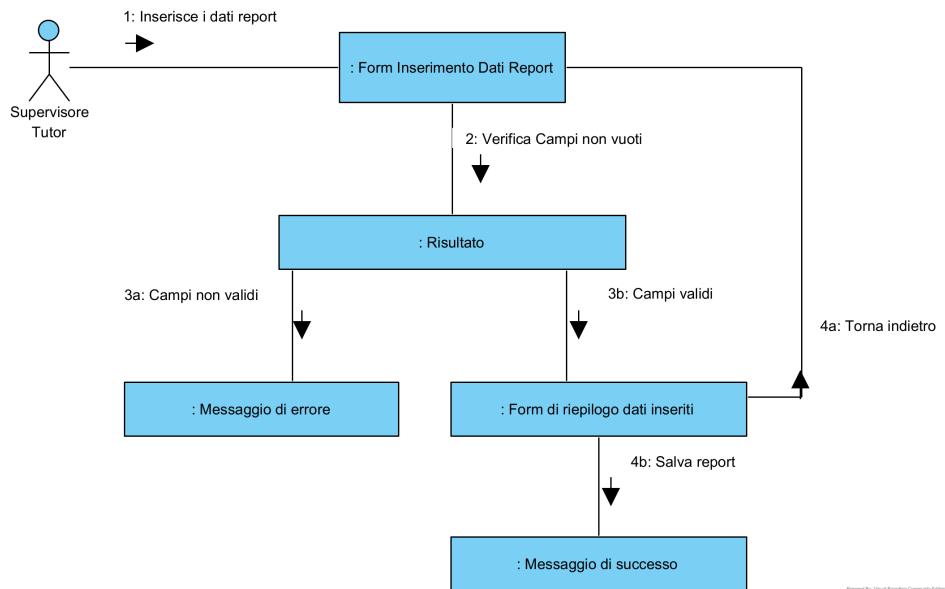


Figura 4.30: Communication Diagram – Creazione report.

- 1 Inserimento dei dati del report: Il supervisore o tutor interagisce con il modulo di inserimento dati.
- 2 Verifica dei campi: Il sistema effettua una validazione per controllare che tutti i campi siano compilati.
- 3 Risultato della verifica:
 - 3a: Campi non validi: Se i campi non sono validi, viene mostrato un messaggio di errore al supervisore o tutor, che può tornare indietro per correggerli.
 - 3b: Campi validi: In caso di validazione positiva, viene mostrato un modulo con il riepilogo dei dati inseriti.
- 4 Salvataggio del report:
 - 4b: Se il riepilogo è confermato, il sistema salva il report e mostra un messaggio di successo.
 - 4a: In alternativa, l'utente può tornare indietro per apportare modifiche ai dati.

4.9.4.2 Modifica report

Il seguente Communication Diagram rappresenta il flusso di interazione per la modifica di un report da parte del supervisore o tutor. Il processo è strutturato in modo da verificare la validità dei campi e apportare le modifiche richieste.

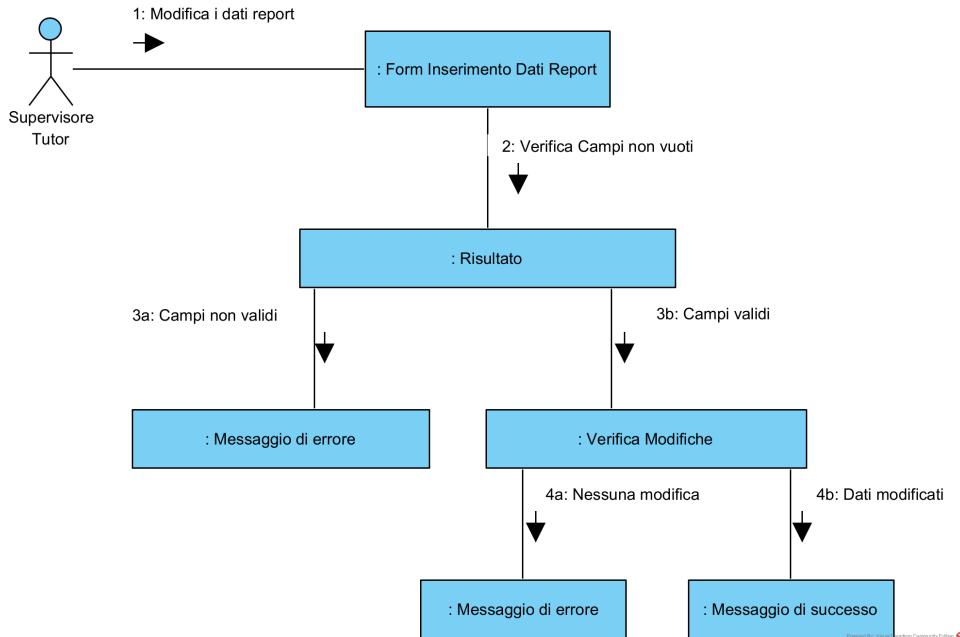


Figura 4.31: Communication Diagram – Modifica report.

- 1 Modifica dei dati del report: Il supervisore o tutor accede al modulo di modifica dei dati del report.
- 2 Verifica dei campi: Il sistema controlla che tutti i campi siano compilati correttamente.
 - 3a: Campi non validi: Se i campi non soddisfano i criteri, viene visualizzato un messaggio di errore.
 - 3b: Campi validi: In caso di validazione positiva, il sistema verifica se sono state effettuate modifiche ai dati.
- 3 Risultato della verifica delle modifiche:
 - 4a: Nessuna modifica: Se non sono state apportate modifiche, il sistema notifica l'utente con un messaggio di errore.
 - 4b: Dati modificati: Se i dati sono stati modificati correttamente, il sistema salva le modifiche e mostra un messaggio di successo.

4.9.5 Communication Diagram – Attività

Il Communication Diagram rappresenta il flusso di interazione per la visualizzazione dei log da parte del supervisore. Questo diagramma evidenzia la sequenza di messaggi tra l'attore e il sistema per ottenere e mostrare i log richiesti.

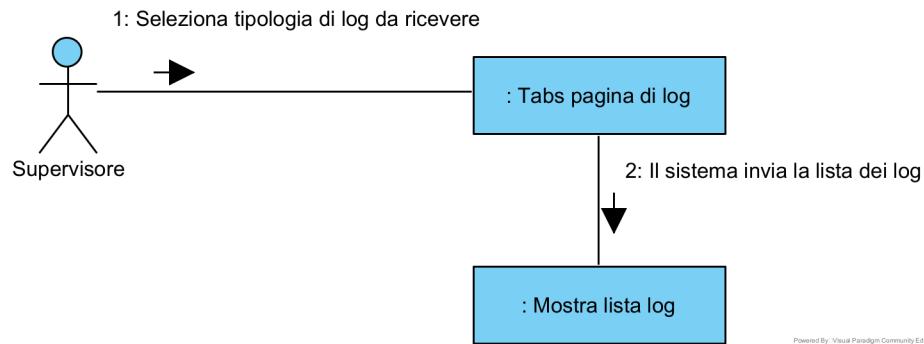


Figura 4.32: Communication Diagram – Attività.

- 1 Selezione del tipo di log: Il supervisore interagisce con la pagina dei tab dei log per selezionare la tipologia di log che desidera visualizzare.
- 2 Richiesta dei log: Il sistema elabora la richiesta e invia la lista dei log corrispondente alla selezione effettuata.
- 3 Visualizzazione dei log: La lista dei log viene mostrata al supervisore all'interno della sezione corrispondente dell'interfaccia.

4.9.6 Communication Diagram – Storage

Il Communication Diagram rappresenta il processo di aggiornamento dell'avatar da parte del supervisore o tutor.

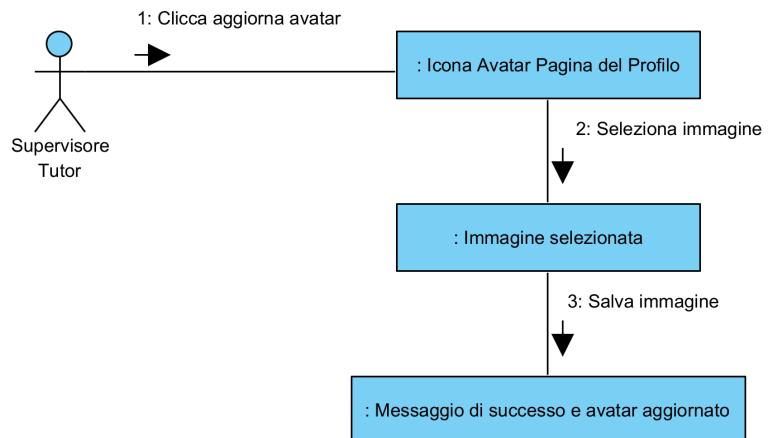


Figura 4.33: Communication Diagram – Storage.

- 1 Selezione dell'opzione di aggiornamento:
 - Il supervisore o tutor clicca sull'icona dell'avatar nella pagina del profilo per iniziare il processo di aggiornamento.

- 2 Selezione dell'immagine:
 - Viene visualizzata un'interfaccia per scegliere un'immagine dal sistema. L'utente seleziona l'immagine desiderata per l'avatar.
- 3 Salvataggio dell'immagine:
 - Una volta confermata la scelta, il sistema procede al salvataggio dell'immagine selezionata come nuovo avatar.
- 4 Conferma del completamento:
 - Il sistema notifica all'utente il successo dell'operazione con un messaggio e aggiorna l'avatar visibile.

4.10 Class Diagram MVC

Il sistema è stato progettato seguendo il modello MVC (Model-View-Controller), riflettendo la struttura modulare dell'applicazione e garantendo una separazione chiara tra le responsabilità. Il diagramma mostra tre principali pacchetti che collaborano tra loro: View, Controller, e Model.

- La View gestisce l'interfaccia utente, consentendo agli utenti di interagire con l'applicazione attraverso componenti strutturati che raccolgono input e mostrano output dinamici. Quando l'utente interagisce, le operazioni vengono inviate al Controller.
- Il Controller agisce come coordinatore, ricevendo richieste dalla View, elaborando la logica applicativa e interfacciandosi con il Model per recuperare o aggiornare i dati.
- Il Model rappresenta i dati dell'applicazione e le regole di business, fornendo un'interfaccia consistente per memorizzare, modificare e restituire le informazioni necessarie.

Il diagramma evidenzia le interazioni tra questi componenti, in cui le richieste fluiscono dalla View al Controller e, quando necessario, vengono propagate al Model. Successivamente, il Controller aggiorna la View o notifica eventuali cambiamenti per riflettere lo stato corrente del sistema.

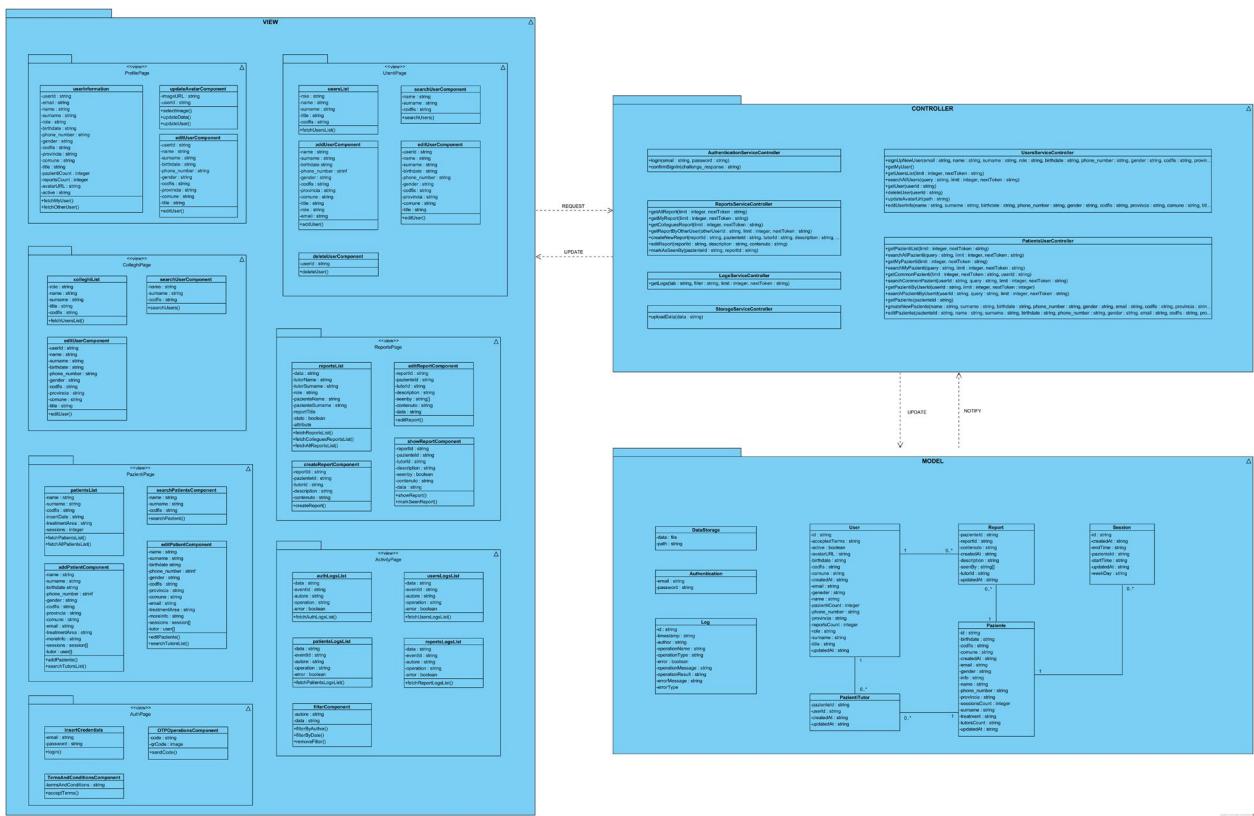


Figura 4.34: Class Diagram MVC.

5 Processo di sviluppo

Si identificano tutte le attività necessarie per lo sviluppo di un sistema software.

5.1 Framework di sviluppo

Il framework scelto per lo sviluppo del software sarà quello più adatto a gestire il cambiamento, che è un elemento inevitabile durante l'intero processo di progettazione. Durante lo sviluppo possono emergere nuove idee, errori imprevisti, o l'introduzione di nuove tecnologie, portando alla necessità di modificare i requisiti iniziali. Questo può comportare rielaborazioni del prodotto, con costi di rework che possono influire negativamente sia sui tempi di consegna che sull'efficienza generale. Ridurre questi costi è essenziale, e si può farlo seguendo due approcci: anticipando i cambiamenti con una metodologia predittiva, oppure adottando un approccio adattivo con consegne incrementali. Quest'ultimo permette di presentare al cliente versioni parziali del sistema per raccogliere feedback e valutarle, evitando modifiche inutili. In linea con questo principio, si utilizzerà il framework SCRUM, che promuove agilità, sviluppo incrementale e continua interazione con il cliente.

5.1.1 SCRUM Framework

SCRUM è un framework agile utilizzato per gestire progetti complessi, specialmente nello sviluppo software. Si basa su cicli di lavoro brevi e iterativi chiamati "sprint", durante i quali il team sviluppa incrementi funzionali del prodotto. Ogni sprint si conclude con una revisione e una retrospettiva, permettendo al team di adattarsi e migliorare continuamente. SCRUM promuove la collaborazione tra team e clienti, la flessibilità di fronte ai cambiamenti e una consegna incrementale del prodotto, favorendo la trasparenza e il miglioramento continuo.

5.1.2 Suddivisione dei ruoli

In linea con il framework SCRUM e per fini puramente didattici, durante lo sviluppo del prodotto software in esame, il team è stato suddiviso nei seguenti ruoli (Tabella 3.1):

Ruolo	Descrizione	Assegnatari
Product Owner	È il responsabile di massimizzare il valore del prodotto da sviluppare. Decide quali funzionalità implementare, stabilisce le priorità e pianifica i rilasci. Deve conoscere profondamente i bisogni dell'utente finale.	Felice Micillo
Scrum Master	Ha il compito di supportare e facilitare l'adozione di SCRUM all'interno del team, rimuovendo ostacoli e gestendo le interazioni tra il team e le parti esterne.	Francesco Scognamiglio
Development Team	Composto da professionisti incaricati di sviluppare e consegnare incrementi del prodotto, pronti per essere rilasciati.	Felice Micillo, Francesco Scognamiglio

Tabella 5.1: Suddivisione dei ruoli per il framework SCRUM.

5.1.3 Evento SCRUM

Di seguito vengono riportate gli eventi inerenti agli Sprint durante il processo di sviluppo e le tempistiche per ognuno di questi:

Evento	Descrizione	Tempo
Sprint	È l'evento principale che racchiude tutti gli altri eventi SCRUM. Durante uno sprint, il team sviluppa un incremento di prodotto utilizzabile e pronto per un eventuale rilascio.	1 settimana
Sprint Planning	Durante questa riunione, il team definisce ciò che deve essere realizzato nello sprint e pianifica come eseguire il lavoro. Viene stabilito l'obiettivo dello sprint, lo "Sprint Goal".	1 ora
Daily Scrum	Riunione giornaliera in cui il team discute il lavoro da svolgere nelle prossime 24 ore, rivede i progressi e affronta eventuali ostacoli per assicurare il completamento delle attività.	20 minuti
Sprint Review	Si tiene alla fine di ogni sprint per rivedere l'incremento del prodotto sviluppato. Il lavoro viene presentato agli stakeholder per raccogliere feedback utili per la pianificazione degli incrementi futuri.	1 ora
Sprint Retrospective	Dopo ogni sprint, il team SCRUM riflette su come è andata la collaborazione e l'esecuzione del lavoro, cercando modi per migliorare la produttività e il processo nello sprint successivo.	1 ora

Tabella 5.2: Eventi SCRUM e tempistiche.

5.2 Descrizione degli Sprint

Di seguito vengono riportati gli sprint eseguiti durante la fase di implementazione delle storie utente definite al capitolo precedente.

Per la pianificazione degli sprint e l'organizzazione del lavoro necessario allo sviluppo del software, abbiamo stabilito delle milestone all'interno di GitLab. Questa piattaforma sarà illustrata in dettaglio in una sezione successiva, dove ne verranno descritte le principali funzionalità e il ruolo nel processo di gestione del progetto.

5.2.1 Sprint 1

Nel primo Sprint sono state implementate le storie utente relative agli Epic: Gestione utente e Supervisione (parziale).

Sono state sviluppate lato Frontend le seguenti pagine e componenti:

- Pagina di autenticazione
- Pagina per la visualizzazione, creazione e rimozione degli utenti
- Pagina del profilo

Sono state sviluppate lato Backend le seguenti funzioni:

- Creazione di un utente
- Ottenimento di una lista di utenti
- Ricerca per tutti gli utenti
- Ottenimento di un utente specifico
- Registrazione di un utente
- Caricamento di un'immagine del profilo
- Modifica di un utente

The screenshot shows a GitLab Milestone card for 'Sprint 1'. At the top left, it says 'Expired' and 'Milestone Nov 19, 2024–Nov 26, 2024'. On the right, there are buttons for 'Close milestone' and three vertical dots. The main content area is titled 'Sprint 1' and contains sections for 'Tempistiche' (with a note about a one-week duration), 'Descrizione' (mentioning the implementation of frontend and backend features), 'Implementazione' (with subsections for 'Frontend' and 'Backend' listing the specific tasks completed). The 'Frontend' section lists the three pages developed, and the 'Backend' section lists the seven functions implemented.

Figura 5.1: Milestone per lo Sprint 1 in GitLab.

5.2.2 Sprint 2

Nel secondo Sprint sono state implementate le storie utente relative agli Epic: Gestione pazienti e Collaborazione (parziale).

Sono state sviluppate lato Frontend le seguenti pagine e componenti:

- Pagina per visualizzare, aggiungere, modificare ed eliminare i pazienti
- Pagina di visualizzazione dei colleghi
- Pagina per la scheda del paziente

Sono state sviluppate lato Backend le seguenti funzioni:

- Aggiungere, modificare e rimuovere i pazienti
- Visualizzare tutti i pazienti
- Ricerca su tutti i pazienti
- Visualizza tutti i colleghi
- Ricerca su tutti i colleghi
- Ottenimento di un paziente specifico

The screenshot shows a GitLab Milestone card for 'Sprint 2'. At the top left is a yellow 'Expired' button with the date 'Milestone Nov 27, 2024–Dec 3, 2024'. At the top right is a 'Close milestone' button. The main title is 'Sprint 2'. Below it is a section titled 'Tempistiche' with the note 'Il tempo per il completamento dello Sprint 2 è di una settimana.' Under 'Descrizione', it says 'Il secondo Sprint prevede l'implementazione di frontend e backend secondo le storie utente documentate.' The 'Implementazione' section is divided into 'Frontend' and 'Backend'. The 'Frontend' section lists: 'Pagina per visualizzare, aggiungere, modificare ed eliminare i pazienti', 'Pagina di visualizzazione dei colleghi', and 'Pagina per la scheda del paziente'. The 'Backend' section lists: 'Aggiungere, modificare e rimuovere i pazienti', 'Visualizzare tutti i pazienti', 'Ricerca su tutti i pazienti', 'Visualizza tutti i colleghi', 'Ricerca su tutti i colleghi', and 'Ottenimento di un paziente specifico'.

Figura 5.2: Milestone per lo Sprint 2 in GitLab.

5.2.3 Sprint 3

Nel terzo Sprint sono state implementate le storie utente relative agli Epic: Gestione report, Collaborazione e Supervisione.

Sono state sviluppate lato Frontend le seguenti pagine e componenti:

- Pagina per visualizzare, creare, modificare i report sui pazienti
- Pagina per supervisionare le attività sulla piattaforma
- Dashboard per l'accesso alle sezioni in maniera diretta

Sono state sviluppate lato Backend le seguenti funzioni:

- Creare, modificare i report
- Visualizzare i propri report
- Visualizzare i report dei colleghi
- Funzioni per la scrittura e lettura dei log
- Associazione dei log con le funzioni del sistema

The screenshot shows a GitLab Milestone card titled 'Sprint 3'. At the top left is a green 'Open' button and the text 'Milestone Dec 4, 2024–Dec 10, 2024'. At the top right are 'Close milestone' and three-dot menu buttons. Below the title is a section titled 'Tempistiche' with the note 'Il tempo per il completamento dello Sprint 3 è di una settimana.' Under 'Descrizione', it says 'Il terzo Sprint prevede l'implementazione di frontend e backend secondo le storie utente documentate.' The 'Implementazione' section is divided into 'Frontend' and 'Backend'. The 'Frontend' section lists: 'Pagina per visualizzare, creare, modificare i report sui pazienti', 'Pagina per supervisionare le attività sulla piattaforma', and 'Dashboard per l'accesso alle sezioni in maniera diretta'. The 'Backend' section lists: 'Creare, modificare i report', 'Visualizzare i propri report', 'Visualizzare i report dei colleghi', 'Funzioni per la scrittura e lettura dei log', and 'Associazione dei log con le funzioni del sistema'.

Figura 5.3: Milestone per lo Sprint 3 in GitLab.

5.3 Software utilizzati durante il processo di sviluppo

Nel processo di sviluppo di un software, l'utilizzo di strumenti collaborativi è fondamentale per garantire che il team lavori in modo sincronizzato ed efficiente. Questi software facilitano la comunicazione, la gestione delle attività e la condivisione delle informazioni, permettendo a tutti i membri del team di rimanere aggiornati e coordinati. La scelta degli strumenti giusti supporta il flusso di lavoro, aumenta la produttività e riduce i rischi di errori o incomprensioni durante lo sviluppo. Di seguito verranno descritti i principali software che abbiamo adottato per

gestire il progetto.

5.3.1 GitLab

GitLab è una piattaforma di gestione del codice sorgente basata su Git, che offre strumenti integrati per la gestione del ciclo di vita del software, dalla pianificazione allo sviluppo fino al monitoraggio. È particolarmente utile per i team che adottano metodologie agili, come Scrum, poiché fornisce funzionalità avanzate per la gestione delle attività, il controllo delle versioni e la collaborazione tra i membri del team.

Nell'ambito di Scrum, GitLab supporta l'organizzazione del lavoro tramite le milestone, che rappresentano obiettivi di alto livello da raggiungere entro specifiche scadenze, allineati con gli sprint. Ogni milestone può essere suddivisa in issue, che corrispondono alle singole attività o funzionalità da sviluppare. Le issue permettono di tracciare lo stato di avanzamento del lavoro, assegnare responsabilità e gestire le priorità.

Frontend: Implementazione della pagina per la visualizzazione, modifica, eliminazione e creazione dei pazienti

Closed Issue created 15 days ago by Francesco Scognamiglio

Tempistiche
Il tempo per il completamento della issue è di 3 giorni (massimo).

Descrizione
Si deve implementare, lato frontend, la pagina per la visualizzazione, modifica, eliminazione e creazione dei pazienti. Si deve fornire una barra di ricerca per i pazienti.

Implementazione

✓ Pagina di visualizzazione dei pazienti
✓ Dialog per l'aggiunta di un paziente
✓ Dialog per la modifica di un paziente
✓ Barra per la ricerca di un paziente

✓ 4 of 4 checklist items completed

0 0 0

Drag your designs here or click to upload.

Child items 0 Add

Assignee Francesco Scognamiglio Edit
Labels FrontEnd Edit
Milestone Sprint 2 (expired) Edit
Weight This feature is locked. Learn more Edit
Due date Nov 28, 2024 - remove due date Edit
Time tracking No estimate or time spent Edit
Confidentiality Not confidential Edit
1 Participant

Figura 5.4: Esempio di Issue in GitLab.

I commit, invece, sono strettamente legati allo sviluppo del codice. Ogni commit rappresenta un cambiamento apportato al codice e può essere associato a una specifica issue, permettendo così di mantenere traccia di quali modifiche sono state fatte per risolvere determinati problemi o implementare nuove funzionalità. Questo collegamento diretto tra commit e issue favorisce una trasparenza totale e una visione chiara del progresso del progetto, in linea con gli obiettivi di Scrum di fornire incrementi di valore a ogni sprint.

Commit dd589f15 authored 11 days ago by Francesco Scognamiglio

Implementazione della pagina per la visualizzazione, modifica, rimozione ed...
Implementazione della pagina per la visualizzazione, modifica, rimozione ed eliminazione dei pazienti e barra di ricerca - Frontend

-> parent e13094a6

Branches 7-frontend-implementazione-della-pagina-per-la-visualizzazione-modifica-eliminazione-e-creazione > Branches containing commit

No related tags found

1 merge request i7 Resolve "Frontend: Implementazione della pagina per la visualizzazione, modifica, eliminazione e creazione dei pazienti"

Changes 32

Showing 20 changed files with 3745 additions and 0 deletions

src/components/common/AddPazienteDialog/AddPazienteDialog.css 0 → 100844 +26 -0 View file @ dd589f15

```
1 + .addpaziente-dialog {  
2 +   padding: 0;  
3 +   margin: 0;  
4 + }  
5 +  
6 + .addpaziente-dialog .MuiPaper-root {  
7 +   display: flex;  
8 +   padding: 0;  
9 +   margin: 0;
```

Figura 5.5: Esempio di Commit in GitLab.

5.3.2 Microsoft Teams

Microsoft Teams è stato utilizzato principalmente come strumento di comunicazione all'interno del team, fornendo una semplice chat per lo scambio rapido di informazioni. In un contesto di sviluppo software, mantenere una comunicazione costante tra i membri del team è cruciale per risolvere rapidamente eventuali problemi, chiarire dubbi e condividere aggiornamenti sul lavoro in corso.

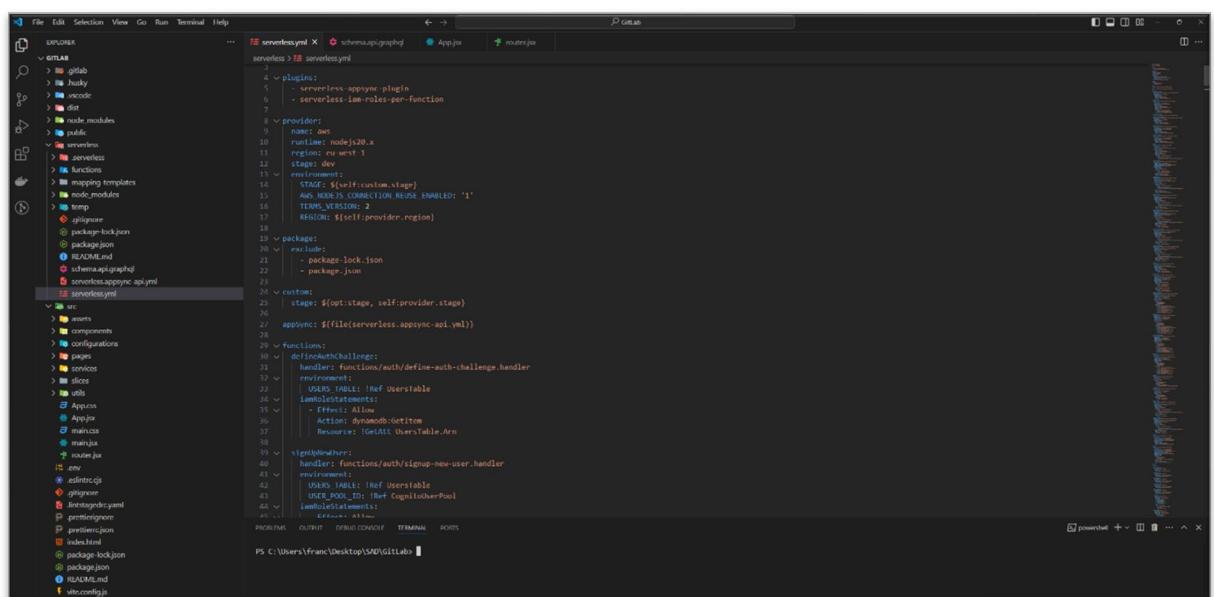
Anche senza integrazioni complesse con altre applicazioni, Teams ha permesso al team di rimanere connesso e di coordinarsi in tempo reale, riducendo i tempi di risposta e facilitando la collaborazione. La semplicità della chat ha reso più fluido il flusso di lavoro, assicurando che tutti fossero sempre allineati sugli obiettivi e sulle attività da svolgere.

5.3.3 Visual Studio Code

Visual Studio Code (VS Code) si è rivelato uno strumento estremamente utile per lo sviluppo sia del frontend che del backend all'interno dello stesso progetto, facilitando l'integrazione e il lavoro su entrambi gli aspetti dell'applicazione.

Per il frontend, VS Code ha supportato tecnologie come HTML, CSS, JavaScript e framework moderni come React. Le estensioni dedicate al frontend, come quelle per il debugging, l'autocompletamento e il linting, hanno migliorato la produttività e la qualità del codice, rendendo lo sviluppo più efficiente.

Per quanto riguarda il backend, sviluppato utilizzando il Serverless Framework con i servizi AWS, VS Code ha dimostrato la sua versatilità grazie a estensioni specifiche per la gestione di infrastrutture serverless. Le estensioni per AWS e il Serverless Framework hanno facilitato la scrittura, il deployment e il monitoraggio delle funzioni Lambda e degli altri servizi AWS come DynamoDB e S3. VS Code ha reso semplice la gestione e la configurazione dei servizi cloud direttamente dal codice, mantenendo tutto il lavoro all'interno dello stesso ambiente.



```
serverless > FB serverless.yml
serverless > FB serverless.yml
  4 < plugins:
  5   - serverless-appsync-plugin
  6     - serverless-lam-roles-per-function
  7
  8 < providers:
  9   - aws:
 10     runtime: nodejs20.x
 11     region: eu-west-1
 12     stage: dev
 13       environment:
 14         STAGE: ${self:custom.stage}
 15         AWS_NODEJS_CONNECTION_REUSE_ENABLED: '1'
 16         TERMS_VERSION: 2
 17         REGION: ${self:provider.region}
 18
 19 < package:
 20   < exclude:
 21     - package-lock.json
 22     - package.json
 23
 24 < custom:
 25   stage: ${opt:stage, self.provider.stage}
 26
 27   appSync: ${file(serverless-appsync-api.yml)}
 28
 29 < functions:
 30   authChallenge:
 31     handler: functions/auth/define-auth-challenge.handler
 32     environment:
 33       | USERS_TABLE: !Ref UserTable
 34       | IMPLEMENTATION: Lambda
 35       | Alias:
 36         Action: dynamodb:PutItem
 37         Resource: !GetAtt UserTable.Arn
 38
 39   signUpHandler:
 40     handler: functions/auth/signup-new-user.handler
 41     environment:
 42       | USERS_TABLE: !Ref UserTable
 43       | AWS_COGNITO_IDP: !Ref CognitoUserPool
 44       | IMPLEMENTATION: Lambda
 45       | Alias:
 46         Action: dynamodb:PutItem
 47         Resource: !GetAtt UserTable.Arn
```

Figura 5.6: Schermata di esempio in Visual Studio Code.

5.3.4 Word

La sua capacità di produrre documenti ben strutturati e formattati ha reso più semplice organizzare in modo chiaro e professionale tutte le informazioni necessarie, come report tecnici, manuali utente e specifiche funzionali.

Anche senza funzionalità di collaborazione in tempo reale, Word ha garantito un formato standard per la documentazione finale, rendendo agevole la distribuzione e la revisione dei contenuti da parte di tutti i membri del team. Grazie alla sua diffusione e compatibilità, Word ha assicurato che i documenti ufficiali potessero essere condivisi e consultati facilmente da tutti, mantenendo la coerenza e l'affidabilità delle informazioni.

5.3.5 Visual Paradigm

Visual Paradigm si è dimostrato uno strumento essenziale per la creazione di documentazione tecnica e per la modellazione visiva di sistemi complessi. La sua capacità di generare diagrammi ben strutturati e di mantenere la coerenza tra i modelli ha facilitato l'organizzazione chiara e professionale delle informazioni, inclusi i diagrammi dei casi d'uso, le specifiche tecniche e i flussi di processo. Sebbene non sia dotato di funzionalità di collaborazione in tempo reale, Visual Paradigm ha fornito un formato standard per la documentazione di sistema, semplificando la revisione e distribuzione dei contenuti all'interno del team. Grazie alla sua versatilità e compatibilità, Visual Paradigm ha permesso di condividere e consultare facilmente la documentazione dei progetti, garantendo la coerenza e l'affidabilità delle informazioni tra tutti i membri del team.

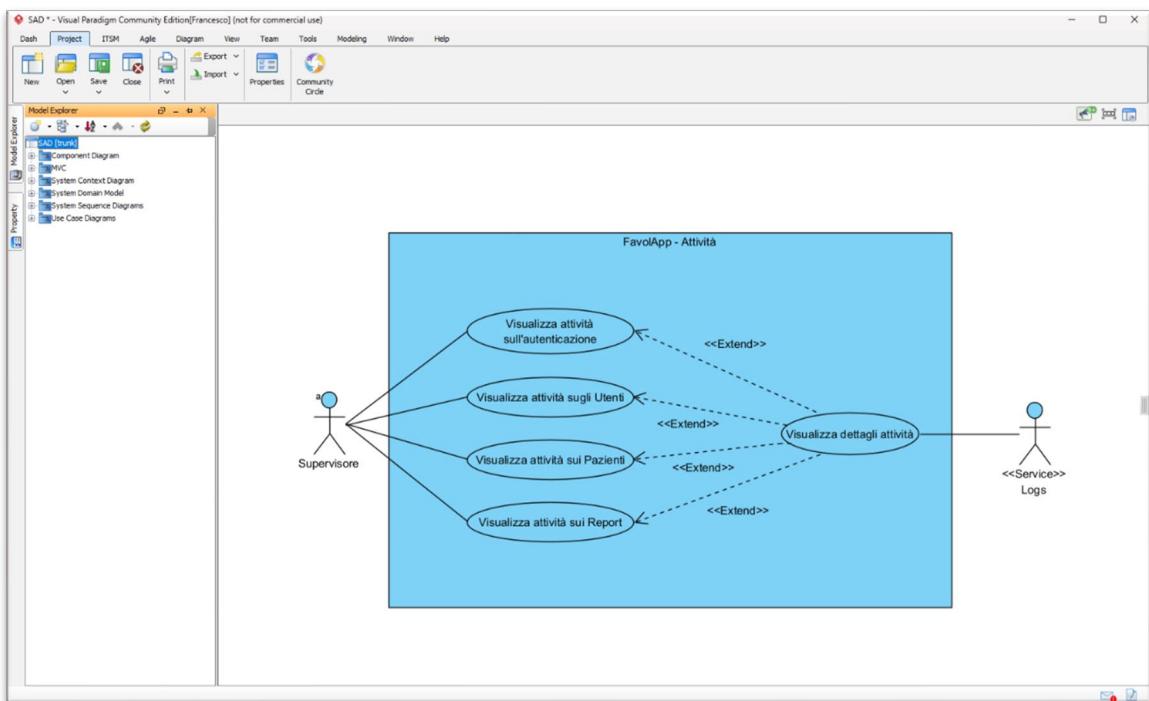


Figura 5.7: Schermata di esempio in Visual Paradigm.

6 Tecnologie utilizzate

Si determinano le tecnologie utilizzate per lo sviluppo dell'applicazione.

6.1 Introduzione alle tecnologie utilizzate

L'implementazione dell'applicazione si basa su un'architettura tecnologica moderna e scalabile, progettata per garantire elevate prestazioni, sicurezza e flessibilità. L'integrazione di strumenti avanzati sia lato frontend che backend ha permesso di creare un sistema robusto, in grado di gestire in modo efficiente le diverse funzionalità richieste. Il frontend è stato realizzato utilizzando ReactJS, una libreria JavaScript dinamica e versatile, mentre il backend si appoggia a servizi serverless di AWS, orchestrati tramite il Serverless Framework. Questa combinazione di tecnologie all'avanguardia consente non solo di ridurre i tempi di sviluppo, ma anche di ottimizzare la gestione delle risorse, garantendo un'applicazione leggera, performante e facile da manutenere.

6.2 Frontend - ReactJS

ReactJS è una libreria JavaScript ampiamente utilizzata per la creazione di interfacce utente dinamiche e reattive. Ideata da Facebook, è progettata per gestire in modo efficiente il rendering di componenti, ottimizzando le prestazioni dell'applicazione grazie al Virtual DOM, che riduce al minimo le modifiche necessarie al DOM reale. La sua struttura basata su componenti consente di costruire interfacce modulari, riutilizzabili e facili da mantenere, rendendo ReactJS una scelta ideale per progetti scalabili. Inoltre, la gestione dello stato, attraverso strumenti integrati o librerie come Redux, permette di sincronizzare facilmente i dati tra i componenti, migliorando l'esperienza utente.

6.2.1 Componenti

ReactJS è basato su un'architettura a componenti, che permette di suddividere l'interfaccia utente in blocchi modulari e riutilizzabili. Ogni componente rappresenta una parte specifica dell'interfaccia, come un pulsante, un modulo o un'intera pagina, ed è definito utilizzando JavaScript e JSX. I componenti possono essere "funzionali" o "class-based", i primi sono maggiormente utilizzati grazie alla semplicità e alle funzionalità avanzate offerte dagli hook, come useState e useEffect.

6.2.2 Virtual DOM

Una delle caratteristiche innovative di ReactJS è l'uso del Virtual DOM, un livello intermedio tra l'interfaccia utente e il DOM reale. Quando l'utente interagisce con l'applicazione, ReactJS aggiorna prima il Virtual DOM, confrontando la nuova rappresentazione con quella precedente (processo di diffing). Solo le modifiche effettive vengono poi applicate al DOM reale, riducendo al minimo le operazioni costose. Questo approccio ottimizza il rendering dell'interfaccia, rendendo ReactJS particolarmente adatto per applicazioni web che richiedono aggiornamenti frequenti, come dashboard o strumenti interattivi.

6.2.3 Single Page Application (SPA)

ReactJS consente di sviluppare applicazioni come Single Page Application (SPA), dove l'intera logica e il contenuto principale dell'applicazione risiedono nel browser. Questo significa che, una volta caricata l'applicazione iniziale, non è necessario ricaricare l'intera pagina per navigare tra le diverse sezioni o funzionalità. Grazie al routing gestito direttamente sul client, ad esempio utilizzando librerie come React Router, le SPA offrono un'esperienza utente fluida e reattiva, simile a quella delle applicazioni native. Questo approccio riduce i tempi di caricamento e ottimizza l'utilizzo delle risorse.

6.3 Backend – Amazon Web Services

Per la realizzazione del backend con i servizi AWS abbiamo utilizzato Serverless Framework. Questo, è uno strumento open-source progettato per semplificare lo sviluppo, la distribuzione e la gestione di applicazioni serverless. Questo framework consente agli sviluppatori di configurare facilmente i servizi cloud, come quelli offerti da AWS e molti altri, attraverso un approccio dichiarativo basato su file di configurazione. L'obiettivo principale del Serverless Framework è ridurre la complessità legata alla gestione manuale delle risorse cloud, automatizzando il deployment.

6.3.1 Configurazione dei servizi e delle risorse

La configurazione dei servizi avviene principalmente tramite un file `serverless.yaml`, che definisce tutte le risorse necessarie per l'applicazione. In questo file vengono specificati dettagli come le funzioni Lambda da implementare, i trigger associati (ad esempio eventi HTTP, cron job o modifiche a un database), le risorse di supporto (come bucket S3 o tabelle DynamoDB), e i permessi richiesti.

Di seguito uno screen del codice parziale del file `serverless.yaml` per la configurazione dei servizi e delle risorse AWS:

```
provider:
  name: aws
  runtime: nodejs20.x
  region: eu-west-1
  stage: dev
  environment:
    STAGE: ${self:custom.stage}
    AWS_NODEJS_CONNECTION_REUSE_ENABLED: '1'
    TERMS_VERSION: 2
    REGION: ${self:provider.region}

package:
  exclude:
    - package-lock.json
    - package.json

custom:
  stage: ${opt:stage, self:provider.stage}

appSync: ${file(serverless.apsync-api.yaml)}

functions:
  defineAuthChallenge:
    handler: functions/auth/define-auth-challenge.handler
    environment:
      USERS_TABLE: !Ref UsersTable
      iamRoleStatements:
        - Effect: Allow
          Action: dynamodb:GetItem
          Resource: !GetAtt UsersTable.Arn

    signUpNewUser:
      handler: functions/auth/signup-new-user.handler
      environment:
        USERS_TABLE: !Ref UsersTable
        USER_POOL_ID: !Ref CognitoUserPool
      iamRoleStatements:
        - Effect: Allow
          Action: dynamodb:PutItem
          Resource: !GetAtt UsersTable.Arn
        - Effect: Allow
          Action:
            - cognito-idp:AdminCreateUser
            - cognito-idp:AdminAddUserToGroup
          Resource: !GetAtt CognitoUserPool.Arn

    createAuthChallenge:
      handler: functions/auth/create-auth-challenge.handler

  verifyAuthChallengeResponse:
    handler: functions/auth/verify-auth-challenge-response.handler
    environment:
      USERS_TABLE: !Ref UsersTable
      iamRoleStatements:
```

Figura 6.1: Codice parziale del file `serverless.yaml`.

6.4 Servizi utilizzati AWS

Di seguito la lista dei servizi AWS utilizzati e le motivazioni riguardo la loro scelta.

6.4.1 AWS Cognito

AWS Cognito è un servizio di autenticazione e gestione degli utenti che consente di aggiungere rapidamente funzionalità di registrazione, accesso e gestione delle identità alle applicazioni web e mobile. Con Cognito, è possibile creare pool di utenti per gestire le credenziali in modo sicuro, supportando flussi di autenticazione standard come username e password, oltre a fornire integrazioni con provider di identità esterni come Google, Facebook e SAML. Inoltre, Cognito genera e gestisce token di accesso, ID e aggiornamento in conformità con gli standard OAuth 2.0, facilitando l'autenticazione nelle applicazioni moderne.

Uno degli aspetti distintivi di Cognito è la gestione avanzata delle sessioni e della sicurezza. Supporta l'autenticazione a più fattori (MFA) utilizzando One-Time Password (OTP), migliorando la protezione degli account utente. Inoltre, integra controlli di sicurezza come il rilevamento di comportamenti insoliti e l'automazione di policy basate sul rischio, bloccando automaticamente tentativi di accesso sospetti.

La scelta di utilizzare AWS Cognito nell'applicazione si basa sulla sua capacità di semplificare la gestione degli utenti e delle credenziali, eliminando la necessità di implementare un sistema di autenticazione personalizzato. Grazie all'integrazione nativa con altri servizi AWS, come AppSync e Lambda, Cognito consente di gestire flussi di autenticazione complessi senza introdurre overhead significativo nello sviluppo.

The screenshot shows the AWS Cognito console interface. On the left, there's a sidebar with navigation links like Panoramica,池 di utenti corrente (Visualizza tutto), Application, Gestione degli utenti (Utenti, Gruppi), Autenticazione (Metodi di autenticazione, Accessi, Registri), Fornitori sociali ed esterni, Estensioni, Sicurezza (AWS WAF, Protezione dalle minacce, Streamling di log), Branding (Dominio, Accesso gestito, Modello di messaggio), Impostazioni, and Pool di identità. The main area is titled 'Utenti [11] info' and displays a table of users. The columns are: Nome utente, Indirizzo e-mail, E-mail verificata, Stato di conferma, and Stato. Each row shows a user ID, an email address, whether the email is verified, the confirmation status, and the overall account status (Abilitato or Disabilitato). Below the table is an 'Importa utenti [0] info' section with a table for importing users from a CSV file, showing columns for Nome del processo, Stato, Utenti importati, Utenti ignorati, Utenti con errori, Registrati di CloudWatch, and Ora di creazione. A note at the bottom says 'Nessun processo di importazione degli utenti trovato' (No import processes found).

Figura 6.2: Pool di utenti nella console di AWS.

6.4.2 AWS AppSync

AWS AppSync è un servizio fully managed che consente di creare API GraphQL scalabili, sicure e flessibili. Questo servizio permette di semplificare la gestione dei dati e la comunicazione tra il frontend e il backend di un'applicazione, fornendo un'unica interfaccia per interrogare, modificare e sottoscrivere i dati in tempo reale. AppSync è in grado di orchestrare le richieste verso varie risorse AWS come DynamoDB, S3 e Lambda, gestendo i resolver direttamente dal backend o eseguendo funzioni serverless per operazioni più complesse.

Un aspetto chiave di AppSync è il supporto nativo a GraphQL, un linguaggio di query che permette ai client di specificare esattamente quali dati necessitano, riducendo il sovraccarico di dati non necessari rispetto alle tradizionali API REST.

Nella nostra applicazione, AppSync è stato scelto per gestire in modo efficiente le query e le mutation GraphQL dal frontend. La sua capacità di collegarsi direttamente alle funzioni Lambda consente di eseguire logiche complesse quando necessario, mentre le interazioni dirette con DynamoDB permettono di gestire i dati in modo veloce e scalabile. L'integrazione con AWS Cognito aggiunge un ulteriore livello di sicurezza, garantendo che solo gli utenti autenticati possano accedere ai dati o eseguire operazioni.

```

schema {
  query: Query
  mutation: Mutation
}

type Query {
  getMyUser: User!
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getUsersList(limit: Int, nextToken: String): UserConnection
  @aws_auth(cognito_groups: ["admin"])
  searchAllUsers(query: String!, limit: Int, nextToken: String): UserConnection
  @aws_auth(cognito_groups: ["admin"])
  getPazientiList(limit: Int, nextToken: String): PazienteConnection
  @aws_auth(cognito_groups: ["admin"])
  searchAllPazienti(query: String!, limit: Int, nextToken: String): PazienteConnection
  @aws_auth(cognito_groups: ["admin"])
  getMyPazienti(limit: Int, nextToken: String): PazienteHydrate
  @aws_auth(cognito_groups: ["admin", "tutor"])
  searchMyPazienti(query: String!, limit: Int, nextToken: String): PazienteConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getCommonPazienti(userId: ID!, limit: Int, nextToken: String): PazienteHydrate
  @aws_auth(cognito_groups: ["admin", "tutor"])
  searchCommonPazienti(userID: ID!, query: String!, limit: Int, nextToken: String): PazienteConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getPaientiByUserId(userID: ID!, limit: Int, nextToken: String): PazienteHydrate
  @aws_auth(cognito_groups: ["admin"])
  searchPaientiByUserId(userID: ID!, query: String!, limit: Int, nextToken: String): PazienteConnection
  @aws_auth(cognito_groups: ["admin"])
  getUser(userID: ID!): User
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getPatient(patientId: ID!): Paziente
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getAllReports(limit: Int, nextToken: String): ReportsConnection
  @aws_auth(cognito_groups: ["admin"])
  getMyReport(limit: Int, nextToken: String): ReportsConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getColleaguesReports(limit: Int, nextToken: String): ReportsConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getMyColleagues(limit: Int, nextToken: String): UserConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  searchMyColleagues(query: String!, limit: Int, nextToken: String): UserConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
  getLogs(tab: String!, filter: FilterLogsInput!, limit: Int, nextToken: String): LogConnection
  @aws_auth(cognito_groups: ["admin"])
  getReportsByOtherUser(otherUserId: ID!, limit: Int, nextToken: String): ReportsConnection
  @aws_auth(cognito_groups: ["admin", "tutor"])
}

type Mutation {
  signUpNewUser(newUser: NewUser!): Boolean!
  @aws_auth(cognito_groups: ["admin"])
  deleteUser(userID: ID!): Boolean!
  @aws_auth(cognito_groups: ["admin"])
  updateAvatarURL(path: String!): String!
  @aws_auth(cognito_groups: ["admin", "tutor"])
}

```

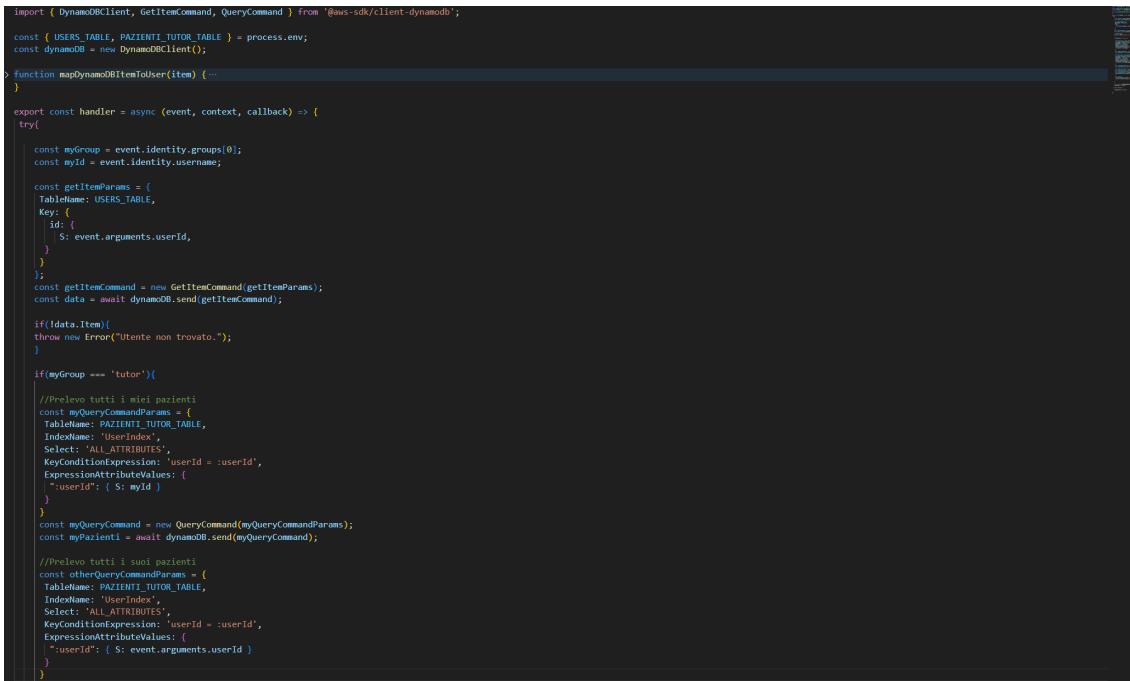
Figura 6.3: Esempio parziale di schema GraphQL per AppSync in Servless Framework.

6.4.3 AWS Lambda

AWS Lambda è un servizio serverless che consente di eseguire codice in risposta a eventi senza dover gestire o configurare server. È progettato per eseguire funzioni che rispondono a trigger, come modifiche nei database o eventi generati da altri servizi AWS. Lambda si basa su un modello "pay-per-use", addebitando solo per il tempo di esecuzione del codice.

Le funzioni Lambda sono stateless e progettate per essere eseguite in modo rapido ed efficiente, scalando automaticamente in base al carico. Questo le rende ideali per una vasta gamma di utilizzi, come l'elaborazione di dati in tempo reale, la gestione di backend di applicazioni, o l'esecuzione di logiche complesse che richiedono solo risorse temporanee. Lambda supporta una varietà di linguaggi di programmazione, tra cui Python, Node.js, Java, e altri, offrendo una grande flessibilità agli sviluppatori.

Nella nostra applicazione, AWS Lambda viene utilizzato per eseguire operazioni che non possono essere direttamente risolte da AppSync, come la gestione avanzata della logica applicativa o l'integrazione con sistemi esterni. Inoltre, Lambda comunica con CloudWatch per raccogliere log e monitorare le esecuzioni. Grazie alla sua integrazione nativa con altri servizi AWS, come DynamoDB e S3, Lambda si adatta perfettamente alla nostra architettura serverless, permettendo di mantenere un'infrastruttura leggera e flessibile. La scelta di AWS Lambda si basa sulla sua capacità di fornire un ambiente di esecuzione reattivo e scalabile, eliminando la necessità di gestire server dedicati.



```

import { DynamoDBClient, GetItemCommand, QueryCommand } from '@aws-sdk/client-dynamodb';
const [USERS_TABLE, PAZIENTI_TUTOR_TABLE] = process.env;
const dynamoDB = new DynamoDBClient({});

function mapDynamoDBItemToUser(item) {
}

export const handler = async (event, context, callback) => {
  try {
    const myGroup = event.identity.groups[0];
    const myId = event.identity.username;

    const getItemParams = {
      TableName: USERS_TABLE,
      Key: {
        id: {
          $: event.arguments.userId,
        }
      }
    };
    const getItemCommand = new GetItemCommand(getItemParams);
    const data = await dynamoDB.send(getItemCommand);

    if(!data.Item){
      throw new Error("Utente non trovato.");
    }

    if(myGroup === 'tutor'){
      //Prelevo tutti i miei pazienti
      const myQueryCommandParams = {
        TableName: PAZIENTI_TUTOR_TABLE,
        IndexName: 'UserIndex',
        Select: 'ALL_ATTRIBUTES',
        KeyConditionExpression: 'userId = :userId',
        ExpressionAttributeValues: {
          ':userId': { $: myId }
        }
      };
      const myQueryCommand = new QueryCommand(myQueryCommandParams);
      const myPazienti = await dynamoDB.send(myQueryCommand);

      //Prelevo tutti i suoi pazienti
      const otherQueryCommandParams = {
        TableName: PAZIENTI_TUTOR_TABLE,
        IndexName: 'UserIndex',
        Select: 'ALL_ATTRIBUTES',
        KeyConditionExpression: 'userId = :userId',
        ExpressionAttributeValues: {
          '-userId': { $: event.arguments.userId }
        }
      };
    }
  }
}

```

Figura 6.4: Esempio parziale di una funzione Lambda in Servless Framework.

6.4.4 AWS DynamoDB

AWS DynamoDB è un database NoSQL completamente gestito, progettato per offrire prestazioni rapide e scalabilità automatica. Grazie alla sua architettura serverless, DynamoDB elimina la necessità di configurare o gestire infrastrutture. È particolarmente adatto per gestire grandi volumi di dati con accessi a bassa latenza.

DynamoDB utilizza una struttura a tabelle con chiavi primarie per identificare univocamente gli elementi, offrendo un'ampia flessibilità nella definizione degli attributi. Supporta modelli di accesso flessibili tramite indici secondari e offre funzionalità avanzate come il backup continuo.

Nella nostra applicazione, DynamoDB viene utilizzato per gestire i dati strutturati, come le informazioni sugli utenti, i pazienti, i report e le attività. Grazie alla sua integrazione nativa con AWS AppSync, DynamoDB consente di eseguire query e mutation in modo rapido ed efficiente, garantendo una sincronizzazione immediata tra il frontend e il database.

La scelta di DynamoDB si basa sulla sua capacità di scalare automaticamente per gestire picchi di traffico, mantenendo al contempo una bassa latenza. Questo lo rende particolarmente adatto per applicazioni che richiedono prestazioni elevate e affidabilità, come nel nostro caso, dove i dati devono essere sempre disponibili e aggiornati.

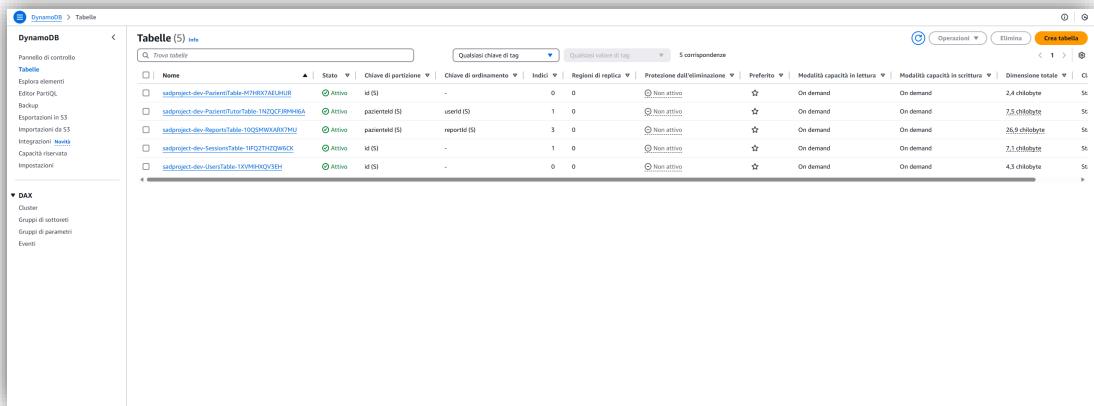


Figura 6.5: Pannello di DynamoDB nella console di AWS.

```

UsersTable:
Type: AWS::DynamoDB::Table
Properties:
  BillingMode: PAY_PER_REQUEST
  KeySchema:
    - AttributeName: id
      KeyType: HASH
  AttributeDefinitions:
    - AttributeName: id
      AttributeType: S
  Tags:
    - Key: Environment
      Value: ${self:custom.stage}
    - Key: Name
      Value: users-table

PazientiTable:
Type: AWS::DynamoDB::Table
Properties:
  BillingMode: PAY_PER_REQUEST
  KeySchema:
    - AttributeName: id
      KeyType: HASH
  AttributeDefinitions:
    - AttributeName: id
      AttributeType: S
  Tags:
    - Key: Environment
      Value: ${self:custom.stage}
    - Key: Name
      Value: pazienti-table

```

Figura 6.6: Esempio di configurazione di una tabella in Serverless Framework.

6.4.5 AWS CloudWatch

AWS CloudWatch è un servizio di monitoraggio completamente gestito che consente di raccogliere, analizzare e visualizzare metriche e log in tempo reale per applicazioni e risorse AWS. Questo strumento è fondamentale per garantire il funzionamento ottimale delle applicazioni, offrendo una panoramica completa delle prestazioni e dello stato dell'infrastruttura.

Il servizio raccoglie dati da una vasta gamma di fonti, tra cui funzioni Lambda, bucket S3, database DynamoDB e molte altre risorse AWS. Oltre a tracciare le prestazioni, CloudWatch consente di analizzare i log generati dalle applicazioni per identificare problemi o colli di bottiglia, facilitando il processo di debugging e ottimizzazione.

Nella nostra applicazione, AWS CloudWatch viene utilizzato principalmente per raccogliere i log generati dalle funzioni Lambda e dai resolver AppSync. Questo consente di monitorare l'esecuzione delle operazioni, identificare errori e ottimizzare il codice per garantire prestazioni elevate. Inoltre, fornisce avvisi tempestivi in caso di anomalie o superamento di soglie predefinite.

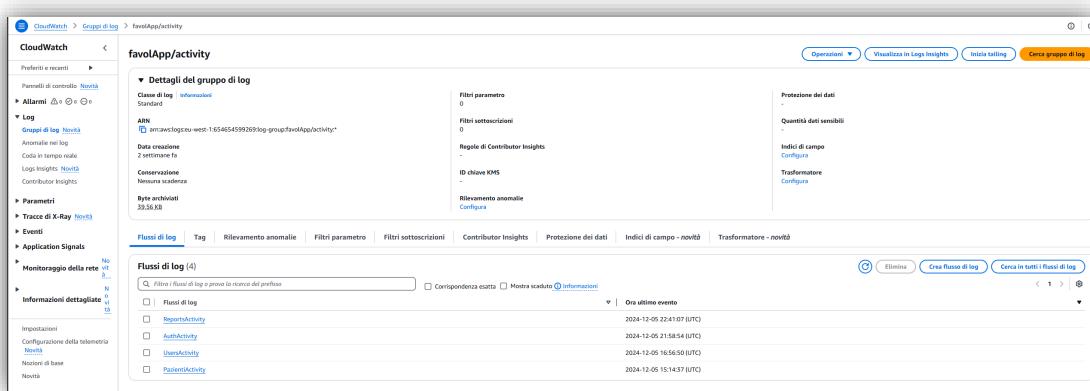


Figura 6.7: Pannello di CloudWatch nella console di AWS.

6.4.6 AWS S3 (Simple Storage Service)

AWS S3 è un servizio di storage scalabile, sicuro e altamente disponibile, progettato per memorizzare e recuperare dati in qualsiasi momento e da qualsiasi posizione. Grazie alla sua architettura serverless, S3 è ideale per gestire file statici e dinamici, offrendo una vasta gamma di utilizzi, tra cui backup, archiviazione di oggetti e distribuzione di contenuti statici.

Nella nostra applicazione, due bucket S3 sono stati configurati per scopi specifici. Il primo è utilizzato per lo storage delle immagini avatar, consentendo agli utenti di caricare e gestire facilmente le immagini del proprio profilo. L'archiviazione avviene in modo sicuro, garantendo l'accesso controllato tramite politiche RBAC. Questo approccio consente di separare i file statici dal database principale, ottimizzando le prestazioni e la gestione dello spazio.

Il secondo bucket S3 è stato configurato per ospitare il frontend dell'applicazione React, che viene distribuito tramite Amazon CloudFront, una Content Delivery Network (CDN). Questo bucket contiene i file statici generati dal processo di build di React, come index.js e altri file statici. La scelta di utilizzare S3 si basa sulla sua scalabilità automatica e sul costo contenuto, che lo rendono una soluzione ideale per gestire sia file statici come gli avatar che l'hosting del frontend.

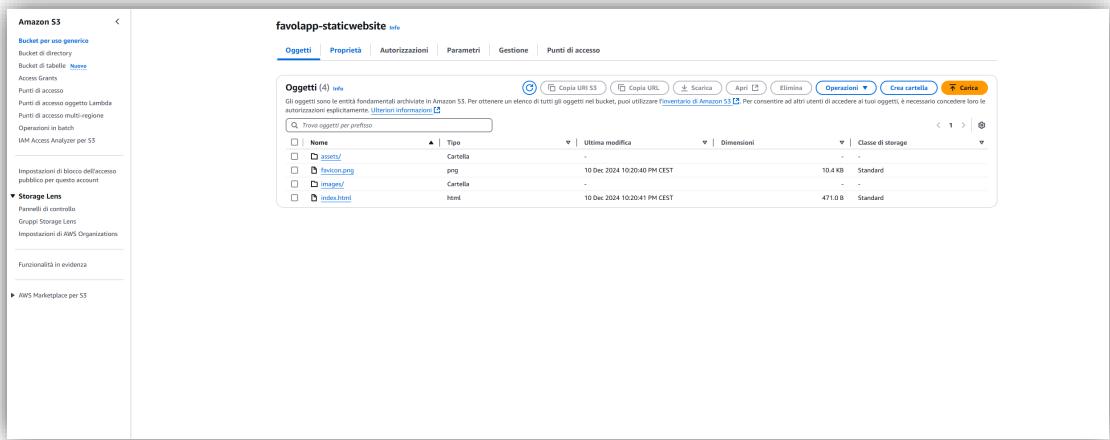


Figura 6.8: Pannello di S3 nella console di AWS.

6.4.7 CloudFront

AWS CloudFront è un servizio di Content Delivery Network (CDN) progettato per distribuire contenuti in modo rapido, sicuro ed efficiente a livello globale. Questo servizio memorizza nella cache i contenuti statici e dinamici presso edge locations distribuite in tutto il mondo, riducendo la latenza e migliorando l'esperienza utente. CloudFront è ideale per applicazioni che necessitano di alte prestazioni e disponibilità, supportando il bilanciamento del carico e l'integrazione con altri servizi AWS per garantire una distribuzione sicura e affidabile.

Nella nostra applicazione, CloudFront è stato configurato per distribuire il frontend React ospitato in un bucket S3. I file statici dell'applicazione, come HTML, JavaScript, CSS e immagini, vengono memorizzati nelle edge locations di CloudFront, consentendo agli utenti di accedere rapidamente ai contenuti da qualsiasi parte del mondo. Questo migliora notevolmente i tempi di caricamento e riduce il carico sul server backend.

CloudFront offre un livello aggiuntivo di sicurezza grazie all'uso di HTTPS per crittografare la comunicazione tra gli utenti e le edge locations. Inoltre, è possibile associare un certificato SSL/TLS tramite AWS Certificate Manager, che consente di gestire in modo semplice e sicuro i certificati necessari per supportare HTTPS.

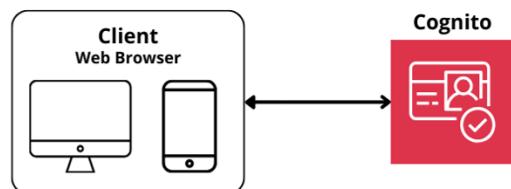
6.5 Architettura AWS

L'architettura AWS implementata per l'applicazione è suddivisa in rami principali, ognuno dei quali svolge un ruolo specifico per garantire efficienza, sicurezza e scalabilità. Questa suddivisione riflette un approccio modulare, in cui ogni componente è responsabile di una funzione chiave, consentendo una gestione ottimale delle risorse e un'integrazione fluida tra i servizi. I rami principali includono lo storage, gestito tramite S3 per i file statici e avatar; l'orchestrazione dei dati, affidata ad AppSync per le operazioni GraphQL e l'integrazione con Lambda e DynamoDB; l'autenticazione, gestita tramite Cognito per assicurare un accesso sicuro e la gestione delle sessioni; e infine la distribuzione dell'applicazione, che utilizza S3 e CloudFront per garantire una consegna rapida e affidabile dei contenuti statici.

6.5.1 Ramo di autenticazione

Nel ramo di autenticazione viene coinvolto solo il servizio AWS Cognito.

Alto livello



Basso livello

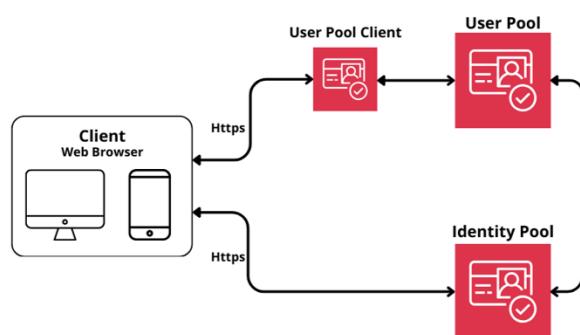


Figura 6.9: Ramo di autenticazione ad alto e basso livello.

Il flusso di autenticazione rappresenta una delle componenti fondamentali del sistema, garantendo un accesso sicuro e controllato agli utenti. Nell'immagine ad alto livello, viene illustrata la comunicazione tra il client, rappresentato dal browser web utilizzato dagli utenti, e il servizio di autenticazione AWS Cognito. Quando un utente tenta di autenticarsi, il client invia le credenziali al servizio Cognito, il quale verifica i dati e risponde con un token di accesso. Questo schema semplice evidenzia l'interazione diretta tra il browser e Cognito, mostrando come le richieste e le risposte avvengano tramite HTTPS, garantendo la sicurezza dei dati in transito.

Nell'immagine a basso livello, il flusso di autenticazione viene ulteriormente suddiviso per includere le due componenti principali di AWS Cognito: il User Pool e l'Identity Pool. Il User Pool gestisce le operazioni di autenticazione, verifica le credenziali degli utenti, e genera i token di sessione per gli utenti autenticati. Una volta autenticato, il client può utilizzare questi token per accedere a risorse specifiche. D'altro canto, l'Identity Pool entra in gioco per fornire credenziali temporanee che consentono agli utenti autenticati di accedere ad altri servizi AWS, come S3 o DynamoDB. La combinazione di questi servizi non solo migliora la sicurezza del sistema, ma rende anche il processo di autenticazione scalabile e facile da integrare con altre risorse AWS.

Di seguito una parte della configurazione della risorsa di Cognito tramite il file serverless.yaml:

```

CognitoUserPool:
  Type: AWS::Cognito::UserPool
  Properties:
    AccountRecoverySetting:
      RecoveryMechanisms:
        - Name: verified_email
          Priority: 1
    AdminCreateUserConfig:
      AllowAdminCreateUserOnly: true
      InviteMessageTemplate: ...
    AutoVerifiedAttributes:
      - email
    DeletionProtection: ACTIVE
    EmailConfiguration:
      EmailSendingAccount: COGNITO_DEFAULT
    MfaConfiguration: 'ON'
    EnabledMfas:
      - SOFTWARE_TOKEN_MFA
    Policies:
      PasswordPolicy:
        MinimumLength: 8
        RequireLowerCase: true
        RequireNumbers: true
        RequireUpperCase: true
        RequireSymbols: true
        TemporaryPasswordValidityDays: 7
      UserAttributeUpdateSettings:
        AttributesRequireVerificationBeforeUpdate:
          - email
    UsernameAttributes:
      - email
    UsernameConfiguration:
      CaseSensitive: false

```

Attivazione MFA con app di autenticazione

Configurazione della policy per la password

```

VerificationMessageTemplate:
  DefaultEmailOption: CONFIRM_WITH_CODE
  EmailMessage: |
    EmailSubject: favolApp - Codice di verifica
  Schema:
    - AttributeDataType: String
      DeveloperOnlyAttribute: false
      Mutable: false
      Name: sub
      Required: true
    - AttributeDataType: String
      DeveloperOnlyAttribute: false
      Mutable: true
      Name: email_verified
      Required: true
    - AttributeDataType: Boolean
      DeveloperOnlyAttribute: false
      Mutable: true
      Name: email_verified
      Required: true
  LambdaConfig:
    DefineAuthChallenge: !GetAtt DefineAuthChallengeLambdaFunction.Arn
    CreateAuthChallenge: !GetAtt CreateAuthChallengeLambdaFunction.Arn
    VerifyAuthChallengeResponse: !GetAtt VerifyAuthChallengeResponseLambdaFunction.Arn
    PreAuthentication: !GetAtt PreAuthenticationLambdaFunction.Arn
    PostAuthentication: !GetAtt PostAuthenticationLambdaFunction.Arn

```

Schema degli attributi dell'utente

Configurazione delle lambda function invocate da Cognito

Figura 6.10: Configurazione parziale di Cognito in Serverless Framework.

Di seguito un sequence diagram di come avviene il flusso di autenticazione tra i servizi aws:

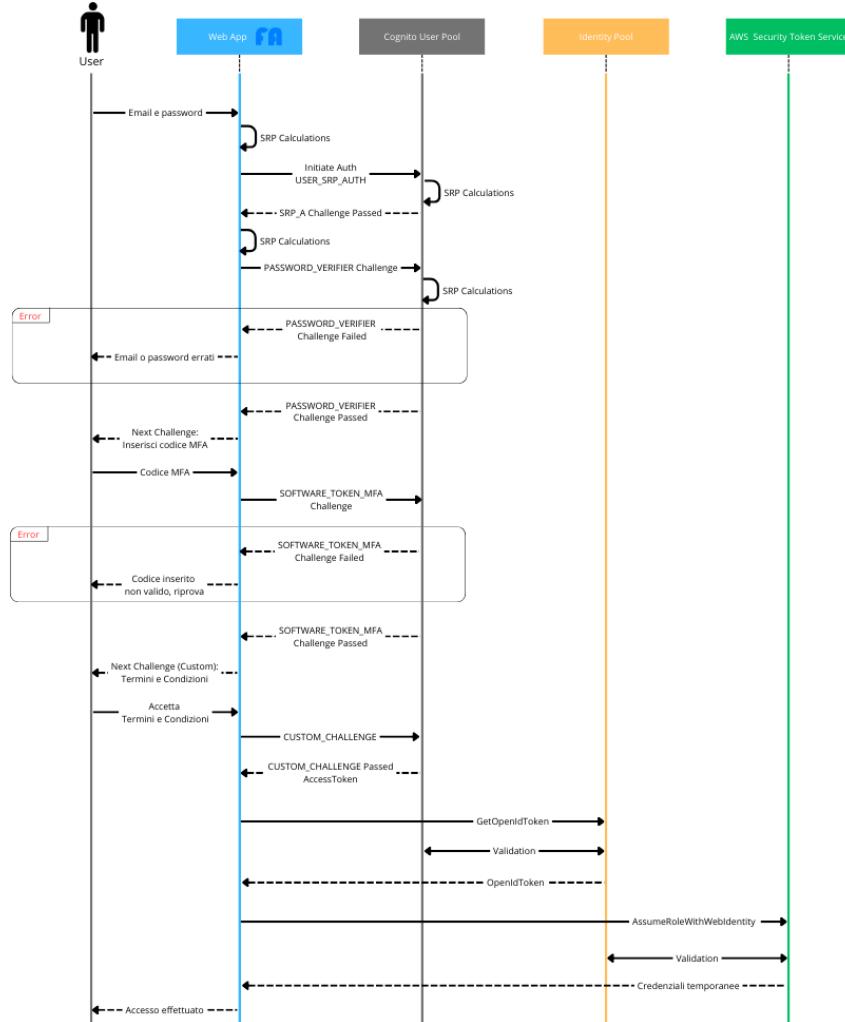
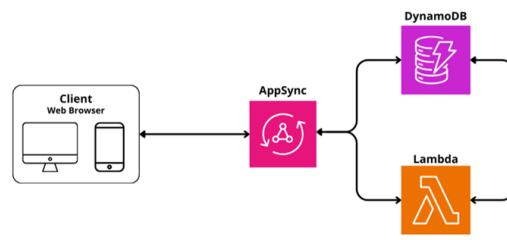


Figura 6.11: Sequence diagram per l'autenticazione.

6.5.2 Ramo di orchestrazione delle richieste

Nel ramo di orchestrazione delle richieste per operare con il database.

Alto livello



Basso livello

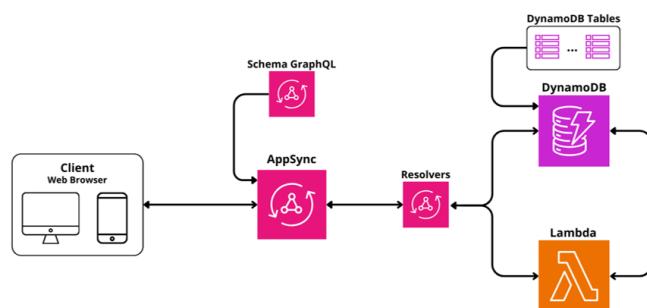


Figura 6.12: Ramo di orchestrazione delle richieste ad alto e basso livello.

L'architettura di AppSync si suddivide in un flusso ad alto livello e un dettaglio più specifico che illustra come le query e le mutation GraphQL vengano orchestrate e collegate ai componenti principali dell'applicazione.

Nell'immagine ad alto livello, il client (che accede tramite un browser web) comunica con AWS AppSync, il servizio che gestisce l'orchestrazione delle query e mutation GraphQL. AppSync agisce come intermediario tra il frontend e i componenti del backend, come DynamoDB per la gestione dei dati persistenti e le funzioni Lambda per le operazioni logiche o complesse. Questa interazione assicura che il frontend possa inviare richieste e ricevere risposte in modo efficiente.

Nel dettaglio rappresentato dalla seconda immagine, il flusso inizia con il client che invia una query o una mutation a uno schema GraphQL definito su AppSync. Questo schema definisce la struttura delle richieste supportate dal sistema. Le richieste vengono inoltrate ai resolver di AppSync, che decidono come e dove instradare i dati richiesti. I resolver possono comunicare direttamente con DynamoDB per operazioni di lettura o scrittura semplici oppure con funzioni Lambda per eseguire logiche più avanzate. Le funzioni Lambda, a loro volta, possono interagire ulteriormente con DynamoDB per operazioni personalizzate, completando così il ciclo di richiesta.

Di seguito la configurazione di un resolver per AppSync in Serverless Framework:

```

getMyUser:
  type: Query
  field: getMyUser
  kind: UNIT
  dataSource: usersTable
  request: mapping-templates/Query.getMyUser.request.vtl
  response: mapping-templates/Query.getMyUser.response.vtl
  
```

Definizione di un resolver


```

dataSources:
  usersTable:
    type: AMAZON_DYNAMODB
    config:
      tableName: !Ref UsersTable
  
```

Definizione di un Data Source


```

{
  "version": "2018-05-29",
  "operation": "GetItem",
  "key": {
    "id": $util.dynamodb.toDynamoDBJson($context.identity.username)
  }
}
  
```

Request Template per DynamoDB


```

$util.toJson($context.result)
  
```

Response Template per DynamoDB

Figura 6.13: Esempio di configurazione di un resolver per AppSync in SF.

Di seguito un sequence diagram di come avviene il flusso di orchestrazione di una richiesta tra i servizi aws:

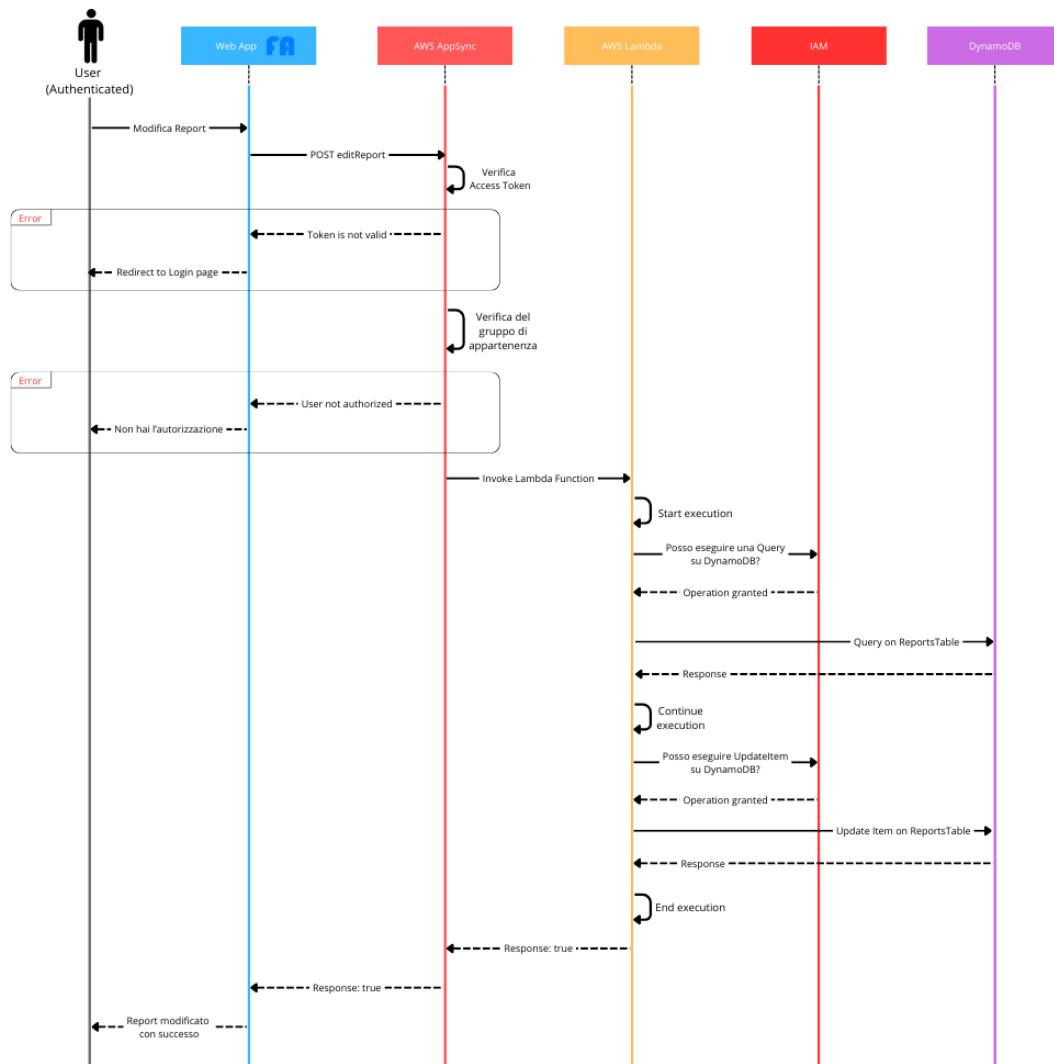


Figura 6.14: Sequence diagram di dettaglio per l'orchestrazione delle richieste con AppSync.

6.5.3 Ramo per lo storage delle immagini

Nel ramo per lo storage delle immagini che permette agli utenti il caricamento e la modifica degli avatar.

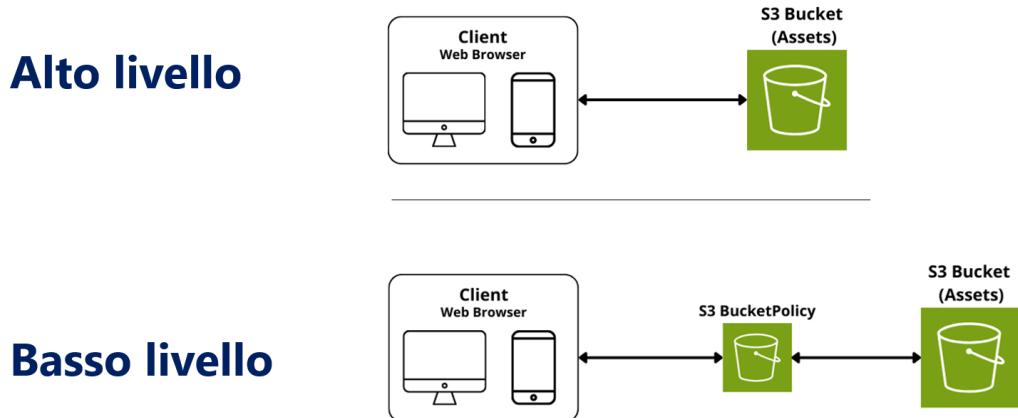


Figura 6.15: Ramo per lo storage delle immagini ad alto e basso livello.

L'architettura per la gestione del caricamento delle immagini, come mostrato, si basa sull'uso di Amazon S3. L'immagine ad alto livello evidenzia il flusso diretto tra il client e il bucket S3, rappresentando un'interazione essenziale per la memorizzazione degli asset, come le immagini degli avatar. In questa configurazione, il client comunica con S3 tramite HTTP, permettendo di caricare direttamente i file nel bucket specifico.

L'immagine a basso livello approfondisce questo processo, mostrando come il caricamento sia regolato da una Bucket Policy di S3. Questa politica stabilisce le regole di accesso al bucket, garantendo che solo i client autorizzati possano eseguire operazioni come il caricamento delle immagini.

In questo flusso, il client effettua il caricamento diretto dell'immagine utilizzando un'API che si interfaccia con S3. Il bucket S3 memorizza quindi le immagini, rendendole disponibili per la visualizzazione o l'aggiornamento in futuro.

Di seguito la configurazione del bucket S3 in Serverless Framework:

Definizione della risorsa Bucket S3

```

AssetsBucket:
  Type: AWS::S3::Bucket
  Properties:
    AccelerationConfiguration:
      AccelerationStatus: Enabled
    CorsConfiguration:
      CorsRules:
        - AllowedMethods:
          - GET
          - PUT
        AllowedOrigins:
        - "*"
        AllowedHeaders:
        - "*"
  
```

AssetBucketPolicy

```

Type: AWS::S3::BucketPolicy
Properties:
  Bucket: !Ref AssetsBucket
  PolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Effect: "Allow"
        Principal:
          AWS: !GetAtt AdminGroupRole.Arn
        Action:
          - "s3:PutObject"
          - "s3:DeleteObject"
          - "s3:GetObject"
        Resource:
          !Sub
            ${!AssetBucketArn}/*
            !GetAtt AssetsBucket.Arn
      - Effect: "Allow"
        Principal:
          AWS: !GetAtt TutorGroupRole.Arn
        Action:
          - "s3:PutObject"
          - "s3:DeleteObject"
        Resource:
          !Sub
            ${!AssetBucketArn}/public/${!CognitoIdentityId}/*
            !GetAtt AssetsBucket.Arn
            CognitoIdentityId: {"cognito-identity.amazonaws.com:sub": "!Sub"}
            !Sub
              ${!AssetBucketArn}/private/${!CognitoIdentityId}/*
              !GetAtt AssetsBucket.Arn
              CognitoIdentityId: {"cognito-identity.amazonaws.com:sub": "!Sub"}
      - Effect: "Allow"
        Principal:
          AWS: !GetAtt TutorGroupRole.Arn
        Action:
          - "s3:GetObject"
        Resource:
          !Sub
            ${!AssetBucketArn}/*
            !GetAtt AssetsBucket.Arn
            !Sub
              ${!AssetBucketArn}/private/${!CognitoIdentityId}/*
              !GetAtt AssetsBucket.Arn
              CognitoIdentityId: {"cognito-identity.amazonaws.com:sub": "!Sub"}*
            !GetAtt AssetsBucket.Arn
            CognitoIdentityId: {"cognito-identity.amazonaws.com:sub": "!Sub"}*
  
```

Ruolo associato
Operazioni ammesse
Path e risorsa
Ruolo associato
Operazioni ammesse
Path e risorsa
Ruolo associato
Operazione ammessa
Path e risorsa

Figura 6.16: Esempio di configurazione del bucket S3 in Serverless Framework.

Di seguito un sequence diagram di come avviene il flusso di caricamento di un'immagine tra i servizi aws:

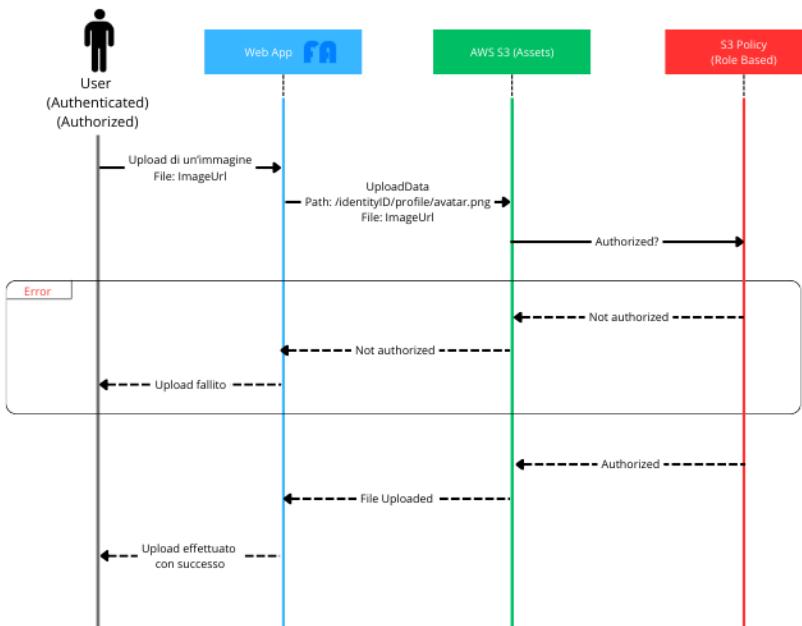
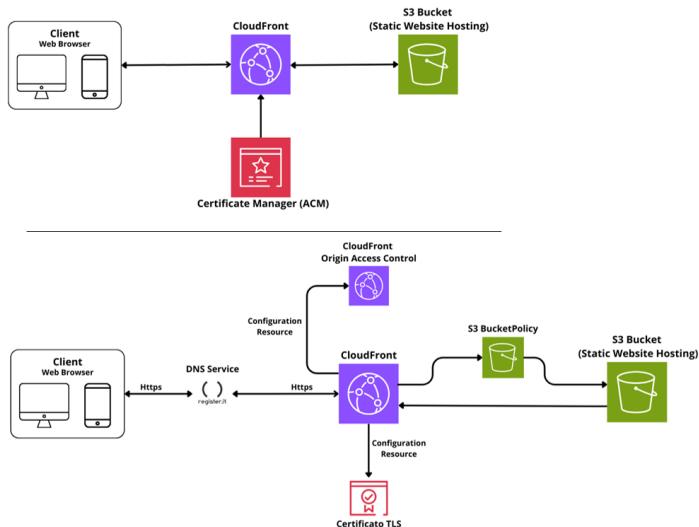


Figura 6.17: Sequence diagram di dettaglio per il caricamento di un'immagine su S3.

6.5.4 Ramo per la distribuzione dell'applicazione

Nel ramo per la distribuzione dell'applicazione avviene il processo di ottenimento dell'applicazione da un bucket S3 dove si trova il frontend attraverso il servizio CloudFront.

Alto livello



Basso livello

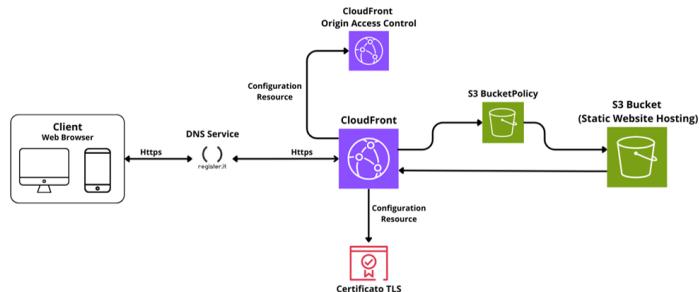


Figura 6.18: Ramo di distribuzione dell'applicazione ad alto e basso livello.

Il primo diagramma rappresenta una visione semplificata del flusso. L'utente interagisce con CloudFront, che si connette al bucket S3 per servire i contenuti statici dell'applicazione. La comunicazione sicura è garantita dal certificato TLS gestito tramite ACM.

Il secondo diagramma illustra in dettaglio il flusso di distribuzione della nostra applicazione web tramite AWS CloudFront e S3, evidenziando la configurazione avanzata necessaria per un hosting sicuro e performante. Quando un utente accede alla nostra applicazione tramite un browser, la richiesta passa attraverso un servizio DNS (ad esempio Register.it), che instrada il traffico verso CloudFront. CloudFront è configurato con un Origin Access Control per garantire che solo esso possa accedere direttamente al bucket S3 contenente i file dell'applicazione. Questo approccio aumenta la sicurezza, impedendo accessi non autorizzati al bucket.

CloudFront utilizza un certificato TLS, gestito tramite AWS Certificate Manager (ACM), per abilitare la comunicazione HTTPS. Questo assicura che tutte le interazioni tra l'utente e l'applicazione siano crittografate e sicure. I contenuti statici dell'applicazione, ospitati in un bucket S3 configurato per il website hosting statico, vengono memorizzati nella cache in più edge locations di CloudFront, migliorando la velocità di distribuzione per gli utenti di tutto il mondo.

Di seguito la configurazione di CloudFront nella console di AWS:

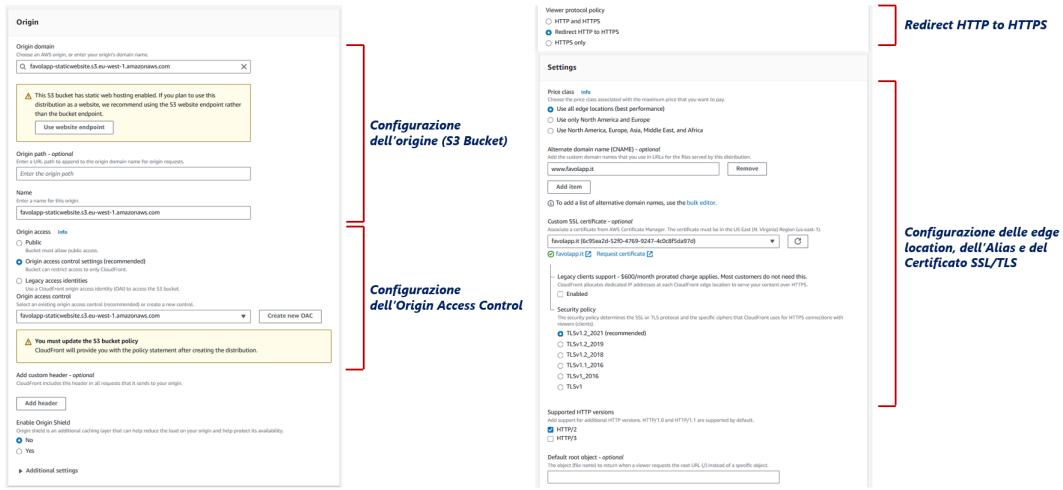


Figura 6.19: Esempio di configurazione di CloudFront nella console di AWS.

Di seguito un sequence diagram di come avviene il flusso di richiesta dell'applicazione tra i servizi aws:

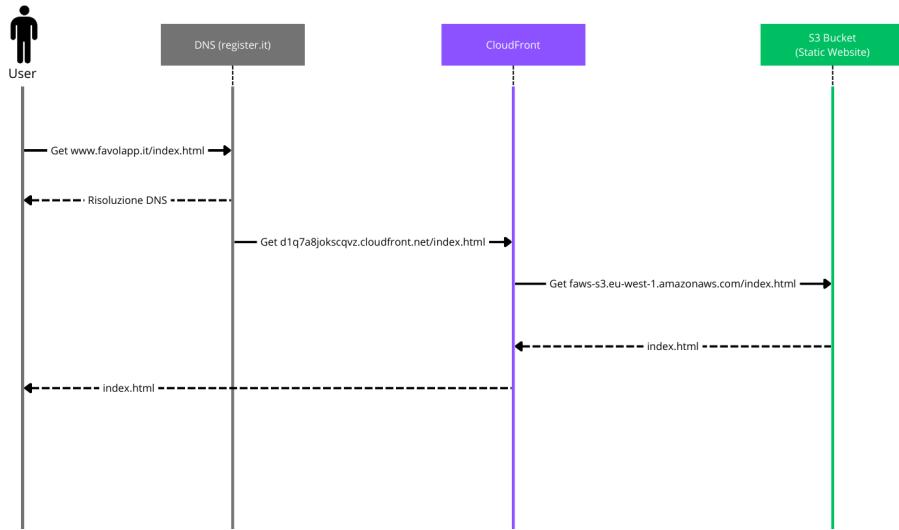


Figura 6.20: Sequence diagram del flusso di richiesta dell'applicazione.

6.5.5 AWS Deployment Diagram

Il diagramma rappresenta l'architettura complessiva dell'applicazione distribuita su AWS, evidenziando l'integrazione tra i servizi chiave.

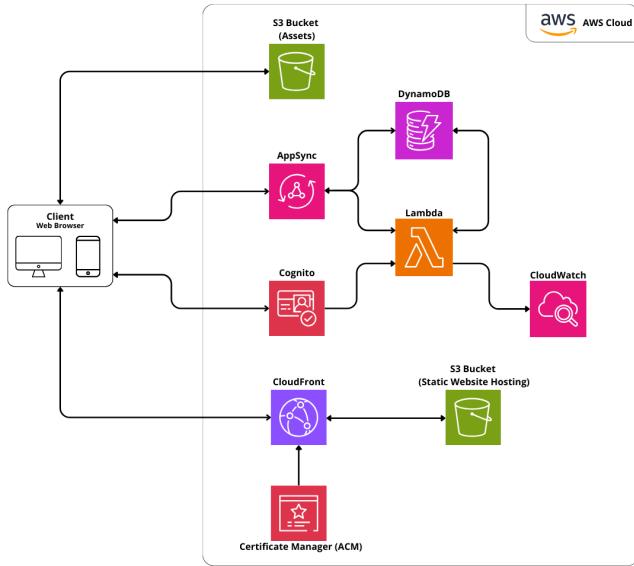


Figura 6.21: AWS Deployment Diagram.

6.6 Comunicazione tra ReactJS e AWS

La comunicazione tra il client React e i servizi AWS nella nostra applicazione è realizzata dall'uso del pacchetto npm aws-amplify. Con questo pacchetto, le richieste vengono inviate dal client React ai servizi AWS configurati attraverso il supporto di API GraphQL e REST. Per la gestione delle query e mutation verso AWS AppSync, aws-amplify utilizza lo schema GraphQL definito sia sul backend che sul frontend, consentendo di eseguire operazioni come l'aggiunta, la modifica o l'eliminazione di dati. Amplify gestisce automaticamente l'autenticazione delle richieste utilizzando i token di sessione forniti da AWS Cognito, garantendo che solo gli utenti autorizzati possano accedere alle risorse.

Per il caricamento di file, come le immagini degli avatar, aws-amplify facilita l'interazione con Amazon S3, fornendo metodi per caricare, recuperare e gestire i file direttamente dai bucket configurati. Questo processo avviene attraverso le policy predefinite che regolano i permessi di accesso, assicurando la sicurezza dei dati.

L'autenticazione degli utenti avviene tramite Cognito, e aws-amplify semplifica l'intero flusso, gestendo le sessioni, i token di accesso e le operazioni di login e logout.

7 Specifiche di implementazione

Si determinano le scelte che hanno determinato la realizzazione del prodotto software.

7.1 Package Diagram

Il Package Diagram è un tipo di diagramma UML che consente di rappresentare la struttura organizzativa e i pacchetti principali di un sistema software. Questo diagramma è particolarmente utile per fornire una visione di alto livello dell'architettura di un'applicazione, evidenziando i moduli principali e le loro dipendenze. L'obiettivo è quello di semplificare la comprensione della struttura complessiva del progetto e facilitare la comunicazione tra i membri del team. I pacchetti rappresentano raggruppamenti logici di funzionalità o componenti, che possono includere file sorgente, configurazioni, risorse o altri elementi pertinenti.

7.1.1 Package Diagram – FavolApp

Nel package diagram mostrato, viene rappresentata l'architettura dell'applicazione FavolApp, suddivisa in due macro-sezioni principali: il Frontend e il Backend. Questa suddivisione riflette la separazione logica tra l'interfaccia utente, realizzata in ReactJS, e i servizi backend, implementati utilizzando AWS tramite il framework Serverless.

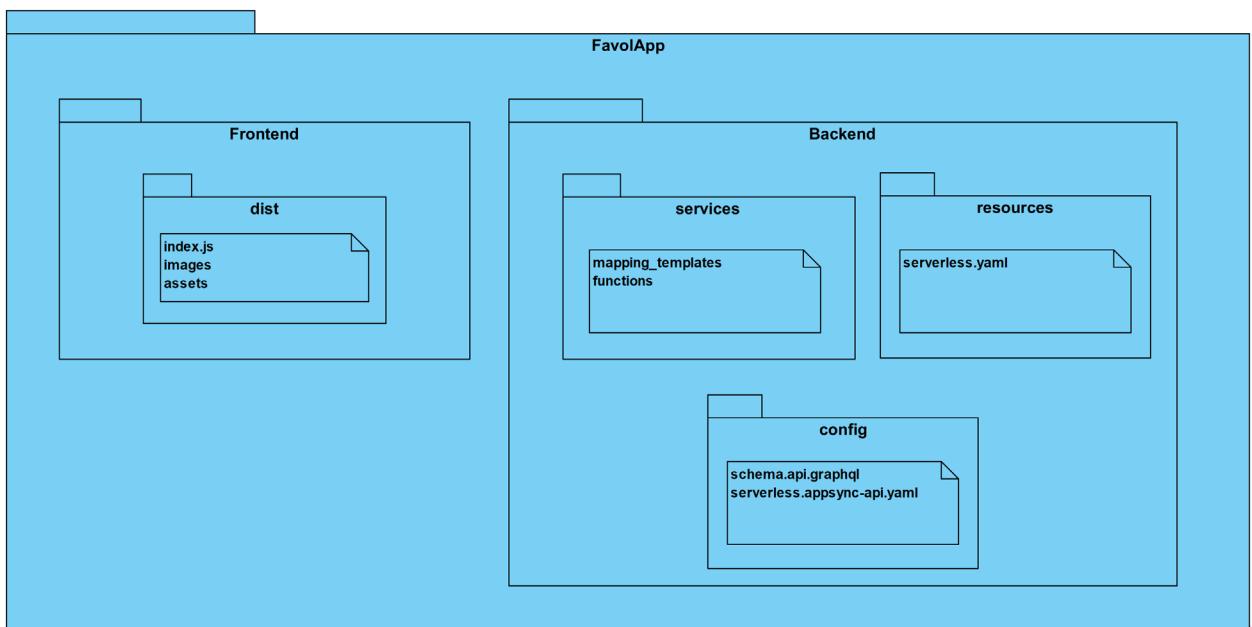


Figura 7.1: Package Diagram - FavolApp

Frontend

Il package relativo al frontend contiene la cartella dist, che rappresenta la versione di produzione dell'applicazione ReactJS. Al suo interno troviamo:

- **index.js**: Il file principale che rappresenta il punto di ingresso del frontend.
- **images**: Una cartella che raccoglie le immagini utilizzate nell'applicazione.
- **assets**: Un contenitore per file statici e risorse aggiuntive necessarie al funzionamento dell'interfaccia utente.

Questa struttura evidenzia come il frontend sia progettato per essere autonomo, ma strettamente connesso al backend per la gestione delle operazioni dinamiche.

Backend

Il backend è suddiviso in tre pacchetti principali:

- **services**: Contiene sottocartelle come mapping_templates e functions, che rappresentano i template di mapping per le query AppSync e le funzioni lambda implementate per gestire la logica dell'applicazione.
- **resources**: Include il file serverless.yaml, che definisce la configurazione del framework Serverless, specificando le risorse AWS necessarie come DynamoDB, AppSync, e S3.
- **config**: Comprende file come schema.api.graphql, che descrive lo schema GraphQL per le query e le mutazioni, e serverless.appsync-api.yaml, che configura le interazioni con AppSync.

Questa organizzazione riflette un'architettura modulare, dove ciascun pacchetto è responsabile di una parte specifica del sistema, garantendo scalabilità e facilità di manutenzione. Inoltre, il MainHandler, integrato tramite AppSync, funge da punto centrale per l'orchestrazione delle richieste, gestendo la comunicazione tra frontend e backend.

7.2 Design Pattern

I design pattern rappresentano soluzioni consolidate a problemi ricorrenti nella progettazione software. Più che semplici regole, essi costituiscono delle linee guida che aiutano i progettisti a strutturare il codice in modo efficace, facilitando la comprensione, la manutenzione e l'estendibilità del sistema. I pattern non offrono un codice pronto, ma descrivono una struttura o un approccio per risolvere problematiche specifiche che emergono durante lo sviluppo, indipendentemente dal linguaggio di programmazione utilizzato.

La loro utilità risiede nel promuovere pratiche di sviluppo basate su concetti come modularità, riusabilità e flessibilità. Grazie ai design pattern, gli sviluppatori possono adottare soluzioni che sono state già validate nel tempo, riducendo i rischi legati alla progettazione e migliorando la qualità del software.

7.2.1 Pattern Grasp

Nel contesto dell'applicazione, le responsabilità delle classi sono state definite seguendo i principi del pattern GRASP (General Responsibility Assignment Software Patterns). Questo approccio permette di distribuire in modo chiaro e logico le responsabilità tra le diverse componenti del sistema, garantendo una progettazione più comprensibile e scalabile.

- Creator: Il principio del Creator è stato applicato per identificare le classi responsabili della creazione di determinati oggetti.
 - La classe Supervisore è considerata il Creator degli utenti, in quanto gestisce la creazione di nuovi profili utente, garantendo che ogni nuovo utente sia correttamente definito e inserito nel sistema.
 - Allo stesso modo, il Supervisore è responsabile della creazione dei pazienti, assicurandosi che i dati anagrafici, i tutor e le sessioni siano correttamente associati.
 - Per i report, è il Supervisore o il Tutor a essere identificato come Creator, dato che si occupano di inizializzare e gestire l'inserimento di nuovi report nel sistema.
- Information Expert: Il principio di Information Expert è stato utilizzato per assegnare alle classi la responsabilità di conoscere e gestire determinate informazioni.
 - La classe Utente funge da Information Expert nel gestire i dettagli del proprio profilo, come avatar, dati personali e credenziali.
 - La classe Paziente è l'Information Expert per i dati relativi alle proprie sessioni, tutor e informazioni mediche associate.
 - Per quanto riguarda i report, la classe Report è l'Information Expert che gestisce i dati dei report, comprese descrizioni, contenuti e associazioni con i pazienti.
- Controller: Il principio del Controller è stato utilizzato per identificare le classi responsabili di ricevere, elaborare e indirizzare le richieste provenienti dalla View.
 - La classe MainHandler funge da Controller centrale, gestendo le richieste inviate dagli utenti attraverso l'interfaccia.
 - Per la gestione degli utenti, il UsersControllerController riceve le richieste per la creazione, modifica ed eliminazione degli utenti.
 - Per i pazienti, il PatientsServiceController è responsabile della gestione delle richieste legate alla registrazione, modifica e visualizzazione delle informazioni dei pazienti.
 - Nel caso dei report, il ReportsServiceController gestisce le operazioni di creazione, modifica e recupero dei dati dei report.
 - Infine, il StorageServiceController è stato implementato come Controller per le operazioni legate al caricamento e all'aggiornamento dell'avatar.

L'applicazione di questi principi ha permesso di organizzare in modo chiaro le responsabilità delle classi, migliorando la modularità e favorendo una separazione delle preoccupazioni. Ogni componente del sistema ha un ruolo ben definito, contribuendo alla robustezza e alla manutenibilità dell'intera applicazione.

7.2.2 Pattern Proxy

Nel contesto dell'applicazione, il pattern Proxy viene implicitamente implementato attraverso l'uso di AWS AppSync, che abbiamo associato al MainHandler. Questo componente funge da intermediario tra la View (realizzata in ReactJS) e il back-end, ricevendo richieste dalla View sotto forma di query e mutation e inoltrandole ai resolver configurati. Il MainHandler, tramite AppSync, si occupa di orchestrare l'accesso al database o ad altri servizi necessari, incapsulando i dettagli implementativi e nascondendoli alla View. Questo approccio garantisce che la View possa interagire con il sistema senza preoccuparsi della complessità sottostante, mantenendo una chiara separazione tra interfaccia utente e logica del sistema, migliorando così modularità, scalabilità e manutenibilità.

7.2.3 Pattern Observer

Nel contesto dell'applicazione, il pattern Observer si applica in specifiche situazioni in cui è necessario notificare determinati eventi a più componenti del sistema. Ad esempio, quando un nuovo paziente viene registrato sulla piattaforma, viene inviata automaticamente un'email contenente le credenziali di accesso al paziente. Questa notifica è una rappresentazione classica del meccanismo di osservazione, dove l'evento di registrazione funge da trigger per l'invio dell'email.

Inoltre, è possibile estendere l'implementazione del pattern Observer affinché ogni operazione significativa, come la registrazione o modifica di dati, venga automaticamente registrata nella pagina delle attività visibile al supervisore. In questo caso, il supervisore agirebbe come osservatore degli eventi generati nel sistema, ricevendo notifiche visibili in tempo reale che riflettono lo stato delle operazioni. Attualmente, il sistema registra le operazioni principali eseguite sul sistema ma non in tempo reale. Questo approccio migliorerebbe la tracciabilità delle azioni all'interno della piattaforma, garantendo un monitoraggio efficace e aggiornato.

7.3 Sequence diagram di dettaglio

Di seguito vengono presentati i diagrammi di dettaglio che illustrano in maniera approfondita le dinamiche di implementazione delle principali funzionalità offerte dal sistema, seguendo l'architettura basata sul design pattern MVC. In particolare, vengono analizzati nel dettaglio tecnico i Sequence Diagram, evidenziando i flussi di interazione tra i diversi componenti e le loro responsabilità operative.

7.3.1 Sequence diagram di dettaglio - Autenticazione

Il Sequence Diagram di dettaglio di seguito descrive il processo di autenticazione dell'utente non autenticato, illustrando i flussi di comunicazione tra i principali componenti del sistema: la View (AuthPage), il Controller (AuthenticationServiceController) e il Model (AuthenticationDB).

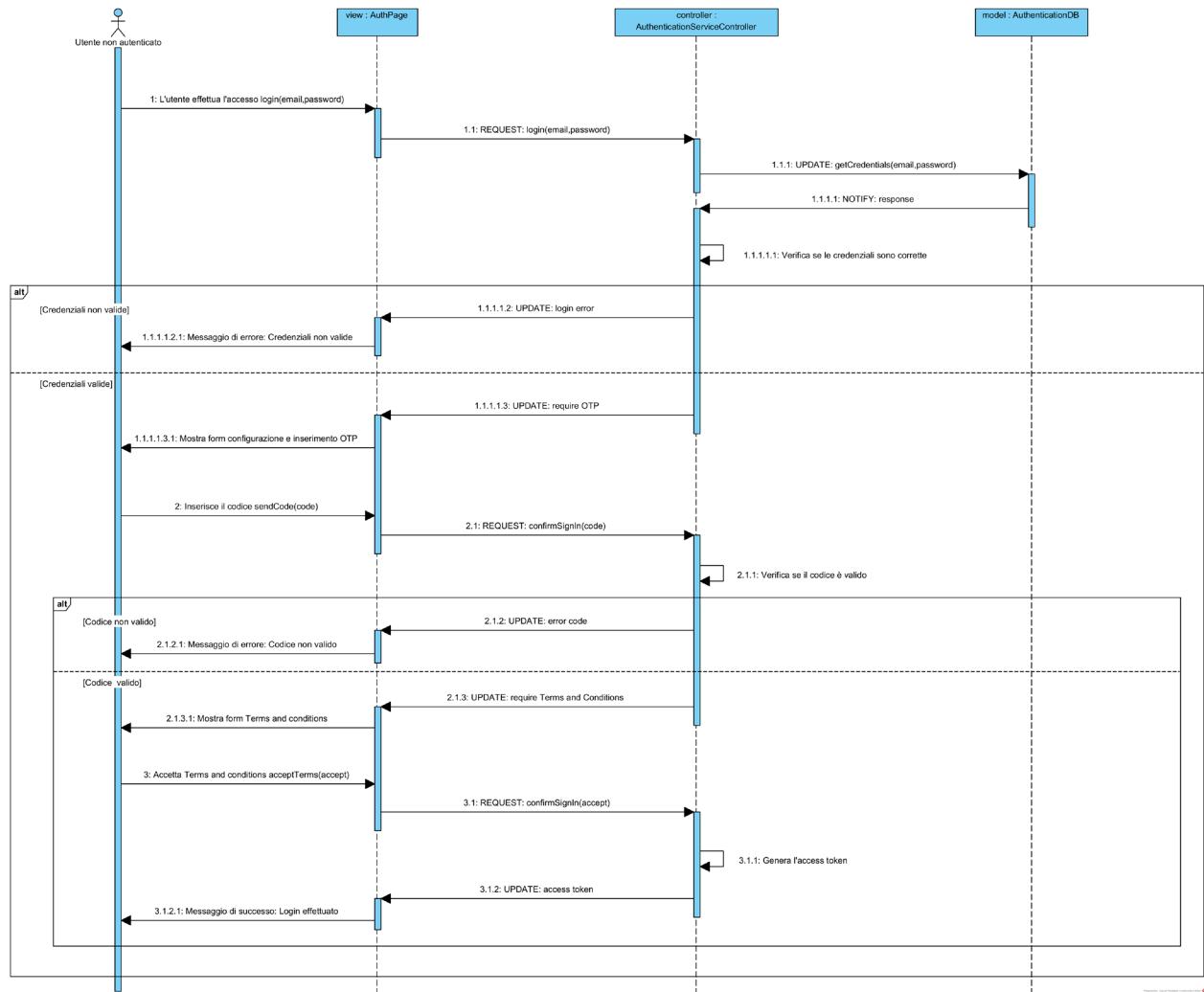


Figura 7.2: Sequence diagram di dettaglio – Autenticazione.

L'intero processo inizia quando l'utente inserisce le proprie credenziali, come email e password, nella View. Successivamente, la View invia una richiesta al Controller per verificare l'autenticità delle credenziali. Quest'ultimo interagisce con il Model per controllare se i dati forniti sono corretti.

Se le credenziali risultano non valide, il Controller aggiorna la View con un messaggio di errore, informando l'utente che l'autenticazione non è andata a buon fine. Al contrario, se le credenziali sono corrette, il Controller richiede un secondo livello di verifica, ossia l'inserimento di un codice OTP. La View si aggiorna per consentire all'utente di immettere il codice ricevuto, il quale viene poi inoltrato al Controller per ulteriori verifiche.

La validità del codice OTP rappresenta un altro punto decisionale. Qualora il codice fosse errato, il Controller aggiorna la View con un messaggio di errore che comunica il problema all'utente. Tuttavia, se il codice OTP risulta valido, il sistema prosegue con la presentazione dei Termini e Condizioni, mostrati nella View. In questa fase, l'utente ha la possibilità di accettare i Termini e, una volta confermata l'azione, la View invia la conferma al Controller. Quest'ultimo conclude il processo generando un token di accesso che viene associato all'utente, consentendogli di accedere al sistema.

Infine, il Controller aggiorna la View con un messaggio di successo, segnalando che l'autenticazione è stata completata con successo. Questo Sequence Diagram evidenzia come il sistema gestisca in modo chiaro e sequenziale le diverse fasi del processo di autenticazione, garantendo al contempo sicurezza e conformità al design pattern MVC.

7.3.2 Sequence diagram di dettaglio - Utenti

Il Sequence Diagram di dettaglio di seguito descrive i processi relativi alla gestione degli utenti, evidenziando i flussi di comunicazione tra i principali componenti del sistema: la View, che rappresenta l'interfaccia utente e le interazioni principali, il Controller, che gestisce le logiche applicative, e il Model, responsabile della gestione e archiviazione dei dati.

7.3.2.1 Registrazione utente

Il Sequence Diagram di dettaglio presentato illustra il processo di registrazione di un nuovo utente, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View, rappresentata da UtentiPage, il Controller, denominato UsersServiceController e AuthenticationServiceController, e il Model, identificato come UsersDB e AuthenticationDB.

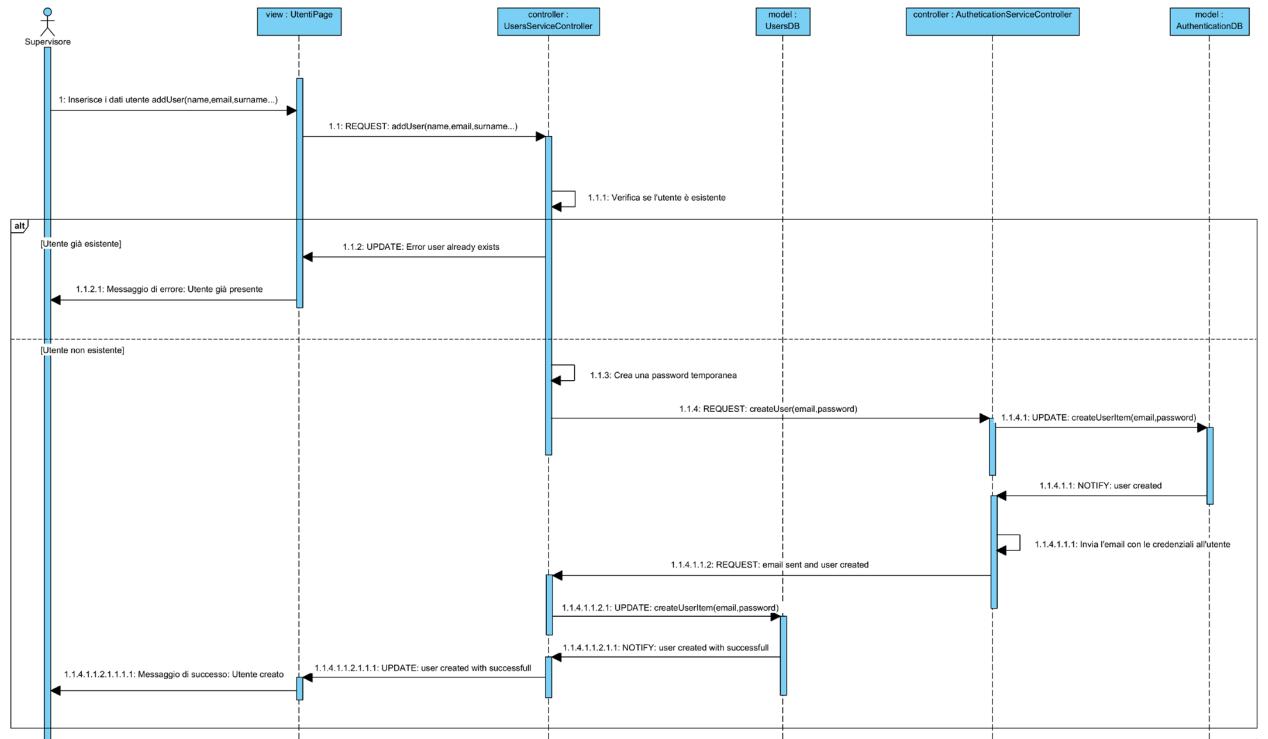


Figura 7.3: Sequence diagram di dettaglio – Registrazione utente.

Il processo di registrazione di un nuovo utente inizia quando il supervisore inserisce i dati richiesti nella View, inclusi nome, cognome, email e altre informazioni necessarie. La View invia una richiesta al Controller principale, il UsersServiceController, per verificare se l'utente da registrare esiste già nel sistema. Il Controller interroga il Model (UsersDB) per effettuare questa verifica.

Se il Controller rileva che l'utente è già presente nel sistema, invia un messaggio di errore alla View, informando il supervisore dell'impossibilità di completare l'operazione. Tuttavia, se l'utente non è presente, il Controller genera una password temporanea per il nuovo utente e procede alla creazione del record utente nel UsersDB.

Successivamente, il UsersServiceController delega la creazione delle credenziali utente al AuthenticationServiceController, che si occupa di gestire i dettagli relativi all'autenticazione. Questo Controller comunica con il Model (AuthenticationDB) per creare l'utente nel sistema di autenticazione, inviando anche una notifica alla View per confermare che le credenziali sono state generate correttamente.

Parallelamente, un'email contenente le credenziali di accesso viene inviata automaticamente all'indirizzo email fornito dal supervisore. Infine, il Controller aggiorna la View con un messaggio di successo, indicando che l'utente è stato registrato con successo nel sistema.

7.3.2.2 **Ottenimento utente**

Il Sequence Diagram di dettaglio presentato illustra il processo di ottenimento del profilo di un utente, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View, rappresentata da ProfilePage, il Controller, denominato UsersServiceController, e i Model, identificati come UsersDB e PatientTutorDB.

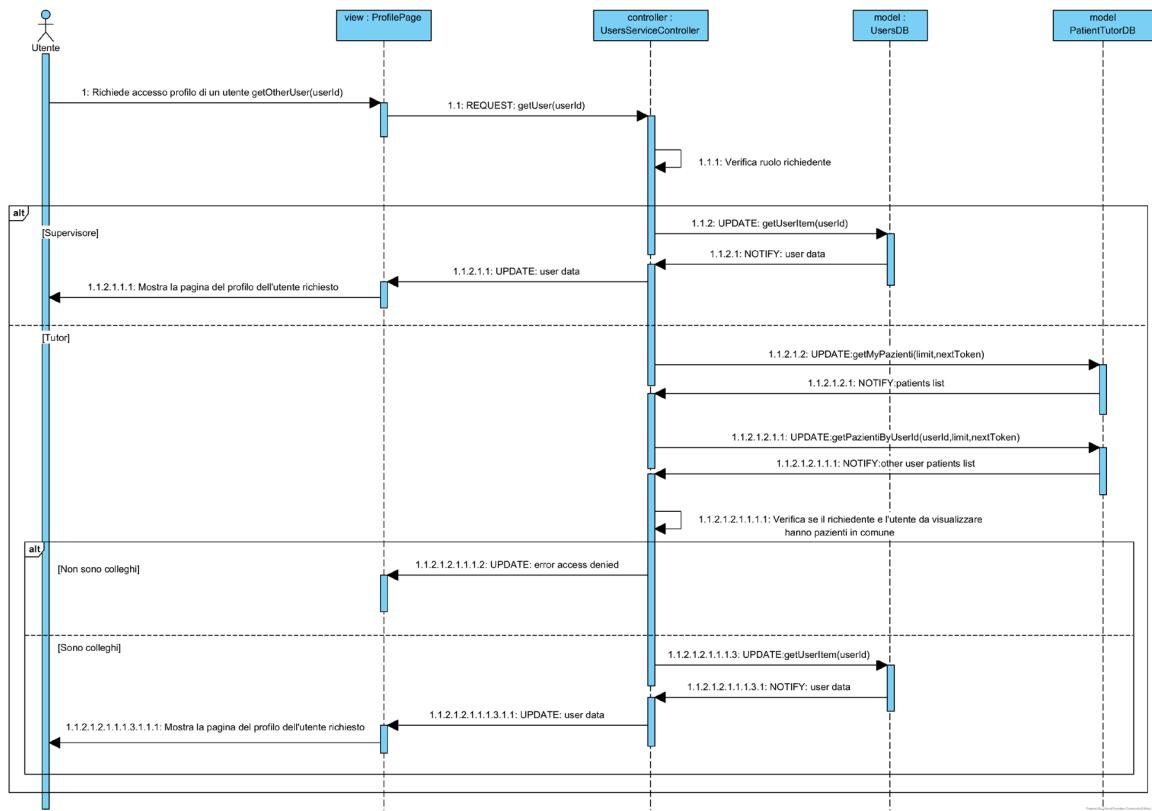


Figura 7.4: Sequence diagram di dettaglio – Ottenimento utente.

Il processo di registrazione di un nuovo utente inizia quando il supervisore inserisce i dati richiesti nella View, inclusi nome, cognome, email e altre informazioni necessarie. La View invia una richiesta al Controller principale, il UserServiceController, per verificare se l'utente da registrare esiste già nel sistema. Il Controller interroga il Model (UsersDB) per effettuare questa verifica.

Se il Controller rileva che l'utente è già presente nel sistema, invia un messaggio di errore alla View, informando il supervisore dell'impossibilità di completare l'operazione. Tuttavia, se l'utente non è presente, il Controller genera una password temporanea per il nuovo utente e procede alla creazione del record utente nel UsersDB.

Successivamente, il UserServiceController delega la creazione delle credenziali utente al AuthenticationServiceController, che si occupa di gestire i dettagli relativi all'autenticazione. Questo Controller comunica con il Model (AuthenticationDB) per creare l'utente nel sistema di autenticazione, inviando anche una notifica alla View per confermare che le credenziali sono state generate correttamente.

Parallelamente, un'email contenente le credenziali di accesso viene inviata automaticamente all'indirizzo email fornito dal supervisore. Infine, il Controller aggiorna la View con un messaggio di successo, indicando che l'utente è stato registrato con successo nel sistema.

7.3.2.3 Eliminazione utente

Il Sequence Diagram di dettaglio presentato illustra il processo di eliminazione di un utente da parte del supervisore, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View, rappresentata da UentiPage, i Controller, denominati UsersServiceController e AuthenticationServiceController, e i Model, identificati come UsersDB e AuthenticationDB.

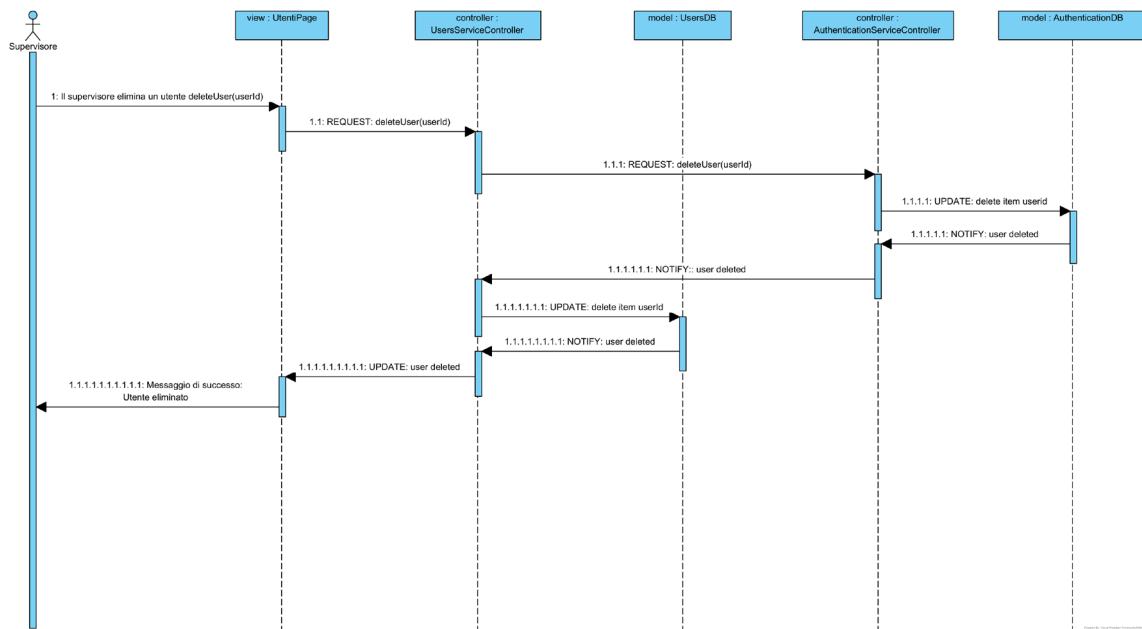


Figura 7.5: Sequence diagram di dettaglio – Eliminazione utente.

Il flusso inizia quando il supervisore seleziona l'opzione per eliminare un utente dalla View (UentiPage), specificando l'identificativo dell'utente (userId). La View invia una richiesta al UsersServiceController per avviare la procedura di eliminazione.

Il UsersServiceController, una volta ricevuta la richiesta, interagisce con il Model (UsersDB) per eliminare il record dell'utente specificato. Al termine di questa operazione, il Model invia una notifica al Controller confermando che l'eliminazione del record è avvenuta con successo.

Dopo la conferma del UsersDB, il UsersServiceController inoltra una richiesta al AuthenticationServiceController, delegandogli la responsabilità di gestire la rimozione delle credenziali di autenticazione dell'utente. L'AuthenticationServiceController a sua volta invia una richiesta al Model (AuthenticationDB) per eliminare le credenziali associate. Una volta completata questa operazione, il Model invia una notifica al AuthenticationServiceController, che a sua volta informa il UsersServiceController del successo dell'operazione.

Infine, il UsersServiceController aggiorna la View (UentiPage) inviando un messaggio di successo che conferma l'avvenuta eliminazione dell'utente. La View riflette questo aggiornamento, informando il supervisore che l'utente è stato eliminato correttamente.

7.3.3 Sequence diagram di dettaglio - Pazienti

Il Sequence Diagram di dettaglio di seguito descrive i processi relativi alla gestione dei pazienti, evidenziando i flussi di comunicazione tra i principali componenti del sistema: la View, che rappresenta l'interfaccia utente e le interazioni principali, il Controller, che gestisce le logiche applicative, e il Model, responsabile della gestione e archiviazione dei dati.

7.3.3.1 Creazione paziente

Il Sequence Diagram di dettaglio presentato illustra il processo di creazione di un nuovo paziente da parte del supervisore, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (PazientiPage), il Controller (PatientsUserController) e i Model (PatientsDB, SessionsDB, e PazientiTutorDB).

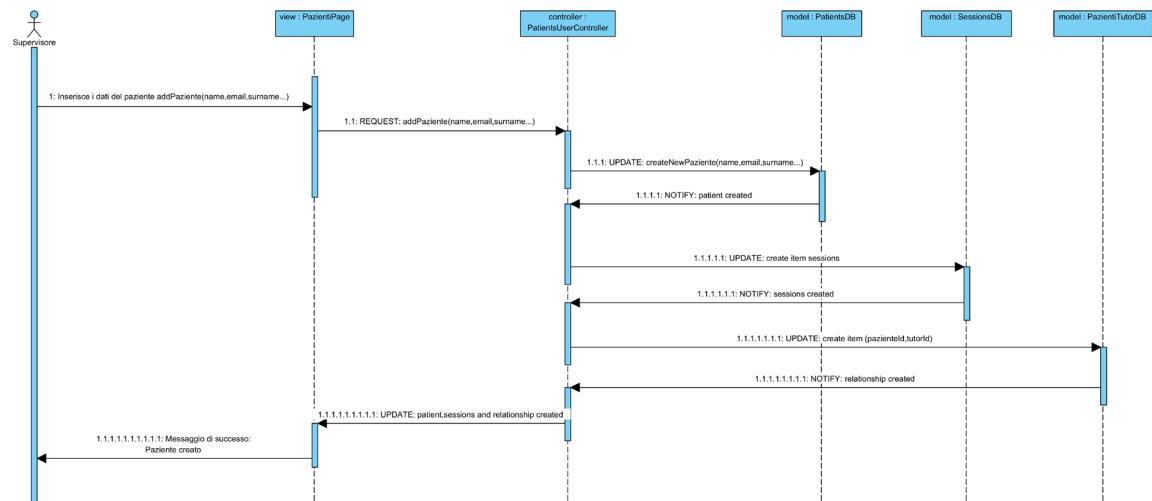


Figura 6.6: Sequence diagram di dettaglio – Creazione paziente.

Il flusso ha inizio quando il supervisore, attraverso la View (PazientiPage), inserisce i dati del paziente, come nome, email e cognome. La View inoltra questi dati al Controller (PatientsUserController) con una richiesta specifica, indicando l'intenzione di aggiungere un nuovo paziente.

Il Controller (PatientsUserController), ricevuta la richiesta, interagisce con il Model (PatientsDB) per creare un nuovo record per il paziente. Una volta che il record è stato generato con successo, il PatientsDB invia una notifica al Controller confermando la creazione del paziente.

Successivamente, il PatientsUserController procede con un secondo step, inviando una richiesta al Model (SessionsDB) per creare le sessioni associate al nuovo paziente. Anche in questo caso, una volta completata l'operazione, il SessionsDB notifica il Controller del successo della creazione.

In un terzo step, il Controller interagisce con il Model (PazientiTutorDB) per definire la relazione tra il paziente appena creato e il tutor responsabile. Una volta registrata questa associazione, il PazientiTutorDB invia una notifica al Controller indicando che la relazione è stata creata correttamente.

Infine, il PatientsUserController comunica con la View (PazientiPage), aggiornandola con un messaggio di successo che informa il supervisore dell'avvenuta creazione del paziente e delle sue associazioni. Questo messaggio conferma che tutte le operazioni, inclusa la registrazione del paziente, la creazione delle sessioni e l'associazione tutor-paziente, sono state completate con successo.

7.3.3.2 Modifica paziente

Il Sequence Diagram di dettaglio presentato illustra il processo di modifica delle informazioni di un paziente da parte del supervisore, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (PazientiPage), il Controller (PatientsUserController) e i Model (PatientsDB, SessionsDB e PazientiTutorDB).

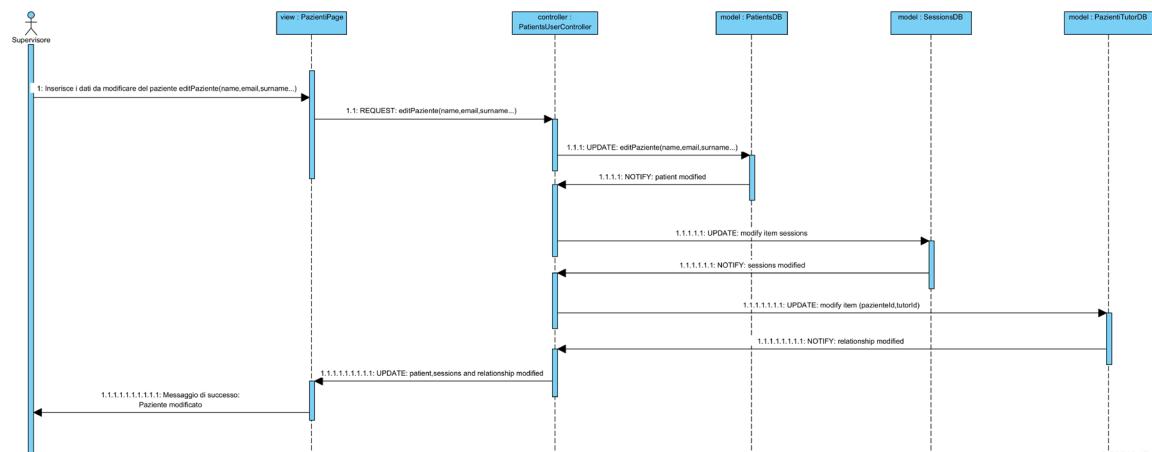


Figura 6.7: Sequence diagram di dettaglio – Modifica paziente.

Il flusso inizia quando il supervisore accede alla View (PazientiPage) e inserisce i dati aggiornati del paziente da modificare, come nome, email e cognome. La View invia una richiesta al Controller (PatientsUserController) per elaborare le modifiche.

Il Controller (PatientsUserController) riceve la richiesta e procede con la modifica del record corrispondente nel Model (PatientsDB). Una volta che il database aggiorna con successo i dati del paziente, il PatientsDB invia una notifica al Controller confermando che la modifica è stata completata.

Successivamente, il Controller si occupa di aggiornare le informazioni relative alle sessioni associate al paziente. A tal fine, invia una richiesta al Model (SessionsDB) per modificare i dettagli delle sessioni. Dopo che queste modifiche sono state effettuate, il SessionsDB invia una notifica al Controller per confermare il successo dell'operazione.

Il passo finale consiste nell'aggiornamento delle relazioni tra il paziente e il tutor responsabile. Il Controller invia una richiesta al Model (PazientiTutorDB) per modificare tali relazioni. Una volta completata questa operazione, il PazientiTutorDB invia una notifica al Controller confermando il completamento dell'aggiornamento.

Infine, il PatientsUserController comunica con la View (PazientiPage), aggiornandola con un messaggio di successo che informa il supervisore dell'avvenuta modifica del paziente, delle sessioni associate e delle relazioni con il tutor. Questo messaggio conferma che tutte le operazioni richieste sono state completate con successo.

7.3.4 Sequence diagram di dettaglio - Report

Il Sequence Diagram di dettaglio di seguito descrive i processi relativi alla gestione dei report, evidenziando i flussi di comunicazione tra i principali componenti del sistema: la View, che rappresenta l'interfaccia utente e le interazioni principali, il Controller, che gestisce le logiche applicative, e il Model, responsabile della gestione e archiviazione dei dati.

7.3.4.1 Creazione report

Il Sequence Diagram di dettaglio presentato illustra il processo di creazione di un report da parte dell'utente, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (ReportsPage), il Controller (ReportsServiceController) e il Model (ReportsDB).

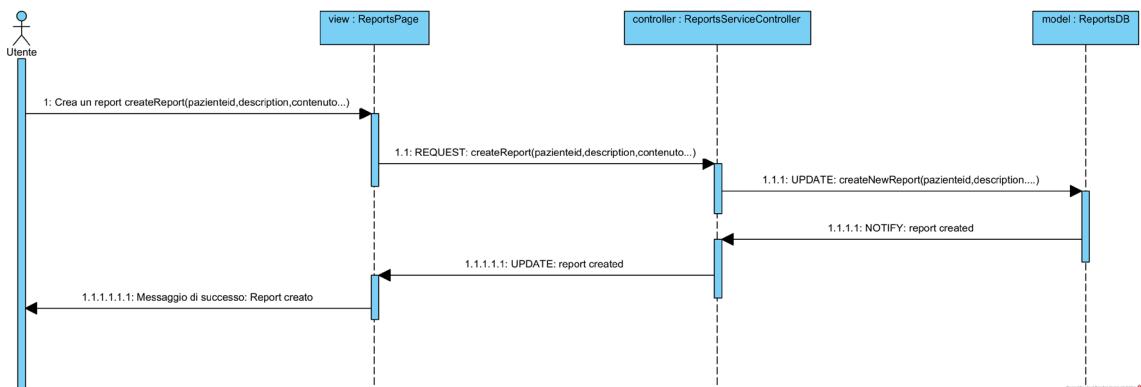


Figura 7.8: Sequence diagram di dettaglio – Creazione report.

Il flusso ha inizio quando l'utente accede alla View (ReportsPage) e inserisce i dettagli necessari per la creazione del report, come il paziente associato, una descrizione e il contenuto del report. Una volta completati i dati, la View invia una richiesta al Controller (ReportsServiceController) per gestire l'operazione di creazione del report.

Il Controller (ReportsServiceController) riceve la richiesta e la processa, interagendo con il Model (ReportsDB) per salvare le informazioni del nuovo report nel database. Il Model si occupa di registrare i dati forniti, e una volta completata l'operazione, invia una notifica al Controller per confermare che il report è stato creato con successo.

In seguito alla conferma del Model, il Controller (ReportsServiceController) aggiorna la View (ReportsPage) con un messaggio di successo che informa l'utente che il report è stato creato correttamente.

7.3.4.2 Modifica report

Il Sequence Diagram di dettaglio presentato illustra il processo di modifica di un report da parte di un utente, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (ReportsPage), il Controller (ReportsServiceController) e il Model (ReportsDB).

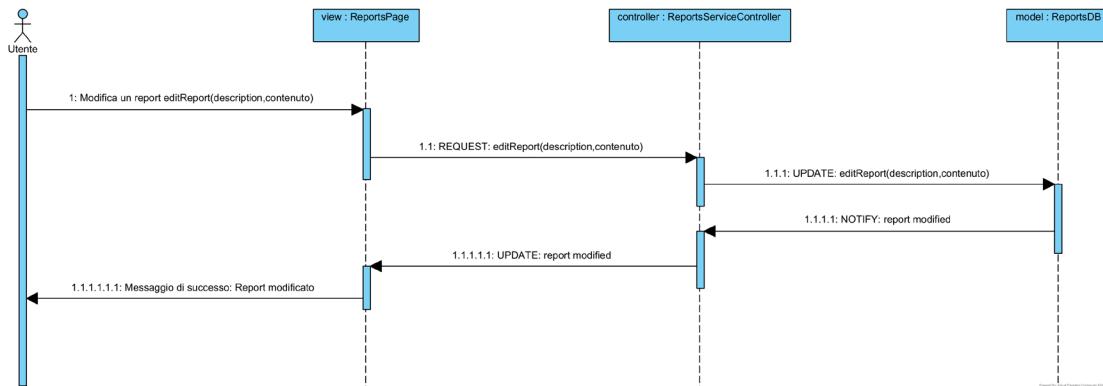


Figura 7.9: Sequence diagram di dettaglio – Modifica report.

L'interazione ha inizio quando l'utente accede alla View (ReportsPage) e invia una richiesta per modificare un report esistente, fornendo i nuovi dettagli, come una descrizione o un contenuto aggiornato. La View trasmette questa richiesta al Controller (ReportsServiceController) per gestire la modifica.

Il Controller (ReportsServiceController) riceve la richiesta e la inoltra al Model (ReportsDB), che si occupa di applicare le modifiche al report corrispondente nel database. Una volta che il Model ha completato l'operazione di aggiornamento, invia una notifica al Controller per confermare che il report è stato modificato correttamente.

Il Controller (ReportsServiceController), dopo aver ricevuto la conferma dal Model, aggiorna la View (ReportsPage) con un messaggio di successo. Questo messaggio informa l'utente che la modifica del report è stata eseguita con successo.

7.3.5 Sequence diagram di dettaglio - Attività

Il Sequence Diagram di dettaglio illustra il processo di recupero dei log delle attività da parte del supervisore, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (ActivityPage), il Controller (LogsServiceController) e il Model (LogsDB).

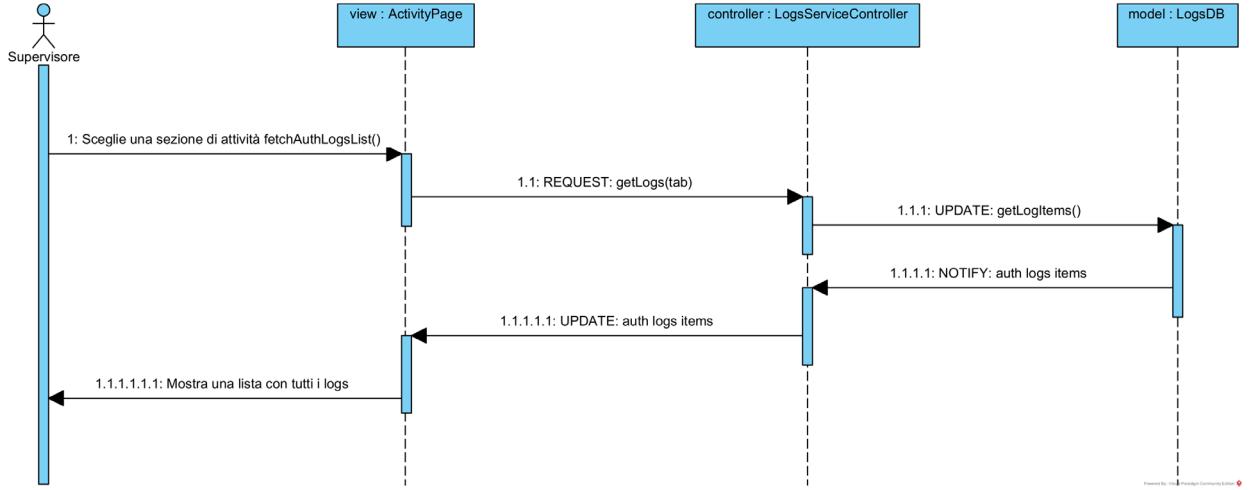


Figura 7.10: Sequence diagram di dettaglio – Attività.

Il processo inizia quando il supervisore, attraverso la View (ActivityPage), seleziona una specifica sezione di log da consultare e invia una richiesta per il recupero delle informazioni pertinenti. La richiesta, contenente il tipo di log desiderato, viene inoltrata al Controller (LogsServiceController).

Il Controller (LogsServiceController), dopo aver ricevuto la richiesta, comunica con il Model (LogsDB) per accedere ai dati dei log richiesti. Il Model esegue la query necessaria e restituisce gli elementi dei log corrispondenti al Controller, notificando il completamento dell'operazione.

Una volta ricevuti i dati, il Controller (LogsServiceController) aggiorna la View (ActivityPage) con le informazioni recuperate. La View, a sua volta, mostra al supervisore una lista dettagliata di tutti i log recuperati, consentendogli di analizzare le informazioni.

7.3.6 Sequence diagram di dettaglio - Storage

Il Sequence Diagram di dettaglio illustra il processo di aggiornamento dell'avatar di un utente, evidenziando il flusso di comunicazione tra i principali componenti del sistema: la View (ProfilePage), i Controller (StorageServiceController e UsersServiceController), e i Model (DataStorage e UsersDB).

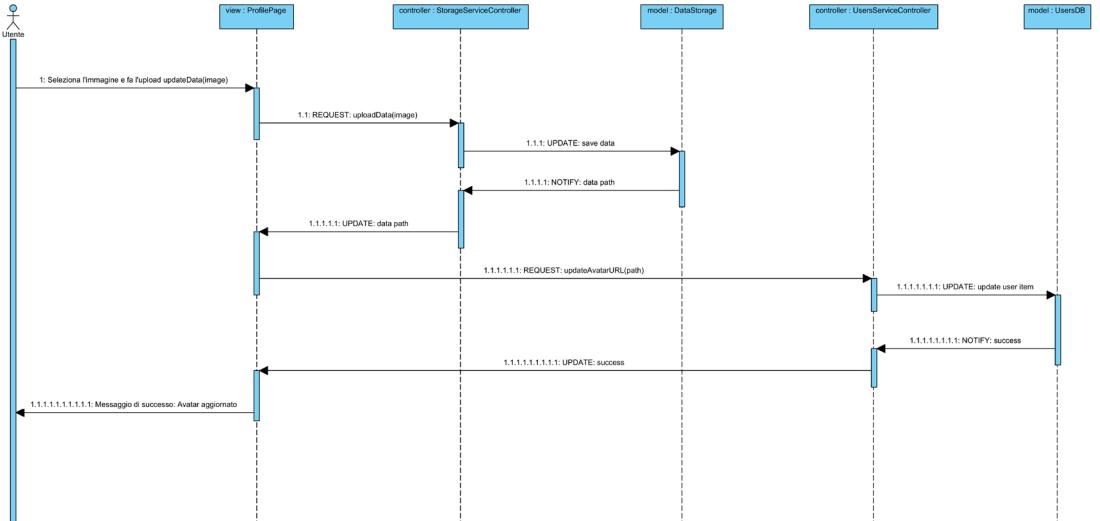


Figura 7.11: Sequence diagram di dettaglio – Storage.

Il flusso ha inizio quando l'utente, attraverso la View (ProfilePage), seleziona un'immagine da caricare come avatar e avvia il processo tramite l'azione updateData(image). La View invia questa richiesta al Controller (StorageServiceController).

Il StorageServiceController, una volta ricevuta la richiesta, inoltra l'immagine al Model (DataStorage), chiedendogli di salvare il file nei suoi archivi. Una volta completata l'operazione, il DataStorage notifica il Controller restituendo il percorso dell'immagine appena salvata.

Ricevuto il percorso del file, il StorageServiceController invia una risposta alla View la quale invia una richiesta al Controller (UserServiceController) per aggiornare l'URL dell'avatar dell'utente nel database degli utenti. Il UserServiceController interagisce con il Model (UsersDB) per aggiornare le informazioni relative all'utente.

Una volta completato l'aggiornamento, il UsersDB notifica il successo dell'operazione al UserServiceController, che a sua volta aggiorna il StorageServiceController e la View (ProfilePage). Infine, la View informa l'utente che l'avatar è stato aggiornato con successo, mostrando un messaggio di conferma.

7.4 Deployment Diagram

Il Deployment Diagram è uno strumento utilizzato per rappresentare la distribuzione fisica dei componenti software su vari nodi di rete o dispositivi. Questo tipo di diagramma offre una visione d'insieme su come i diversi elementi di un'applicazione sono distribuiti in un sistema, evidenziando le interazioni tra hardware e software e i protocolli utilizzati per la comunicazione. È particolarmente utile per comprendere l'architettura del sistema e ottimizzare le risorse.

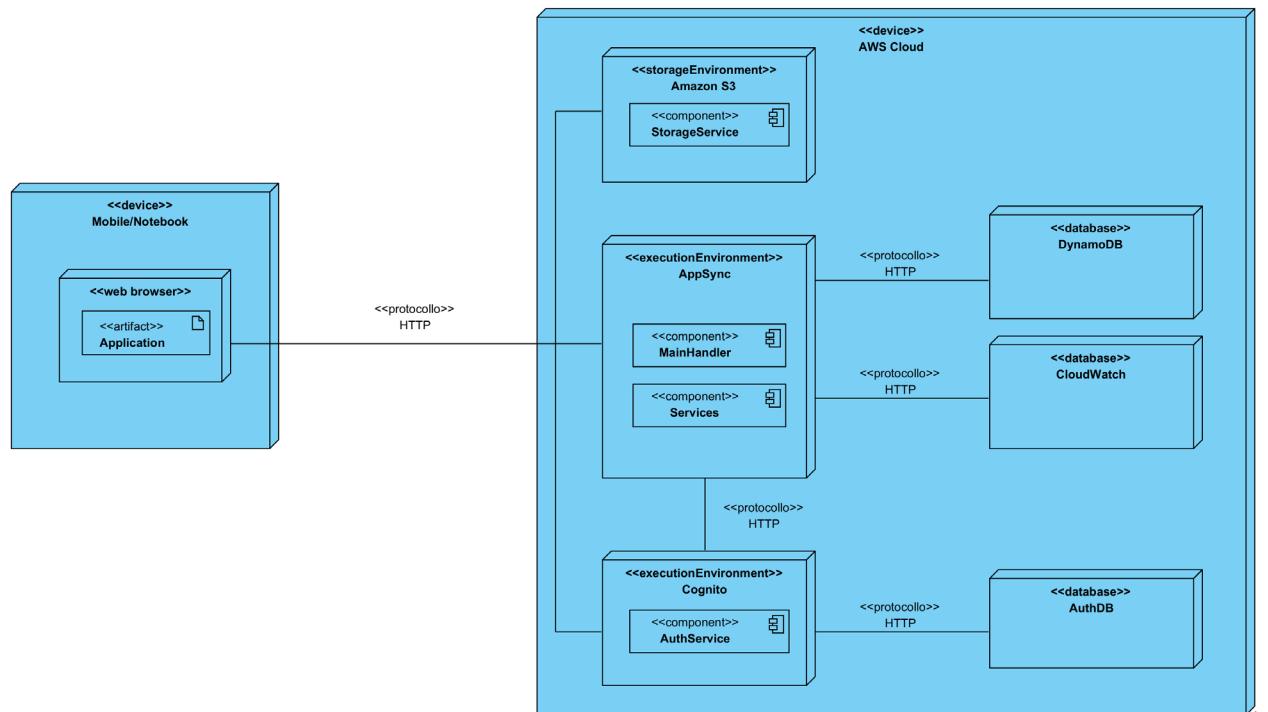


Figura 7.12: Deployment Diagram.

Nel diagramma, viene rappresentata l'architettura di deployment dell'applicazione web. L'applicazione viene eseguita su un device client, come un computer o un dispositivo mobile, attraverso un web browser. La comunicazione tra il dispositivo client e il backend avviene tramite il protocollo HTTP.

Sul lato server, l'applicazione sfrutta vari servizi forniti da AWS Cloud. Il backend è orchestrato tramite AppSync, che è costituito dal MainHandler, responsabile delle richieste provenienti dal frontend, e i Services. I dati sono archiviati in diversi database, come DynamoDB per le informazioni applicative e AuthDB per la gestione delle credenziali degli utenti, CloudWatch per i log.

Il diagramma mostra anche uno storage environment, rappresentato da Amazon S3, utilizzato per archiviare file statici come le immagini caricate dagli utenti. La gestione dell'autenticazione è invece affidata a Cognito, che si occupa di verificare e proteggere l'accesso al sistema.

8 Testing

Si prova a dimostrare con la fase di testing, se un sistema soddisfa i suoi requisiti funzionali e non-funzionali.

8.1 Test funzionali e non funzionali

Il testing rappresenta un passaggio cruciale nello sviluppo del software, poiché consente di individuare difetti e verificare che il sistema soddisfi pienamente i requisiti sia funzionali che non funzionali. Questa attività è essenziale per assicurare l'affidabilità, la qualità e la conformità dell'applicazione rispetto alle aspettative progettuali. Durante lo sviluppo, il testing è stato integrato come processo continuo, eseguito a diversi livelli di dettaglio e in momenti distinti, con l'obiettivo di validare progressivamente ogni componente e funzionalità.

La strategia adottata nella nostra applicazione prevede un testing suddiviso in due principali fasi. La prima riguarda il testing a livello di singolo componente, in cui vengono analizzati separatamente i microservizi per verificarne la correttezza e l'integrità. La seconda fase si concentra sul testing funzionale, volto a garantire che l'interazione tra il frontend e i vari servizi lato server sia priva di errori e completamente operativa.

8.2 Testing dei servizi

Per garantire la qualità e l'affidabilità dei servizi della nostra applicazione, sono stati implementati test di unità e test di integrazione, entrambi realizzati utilizzando il framework JEST. Questo strumento offre una soluzione completa per simulare richieste, analizzare le risposte del sistema e verificare che il comportamento dei servizi sia conforme alle aspettative. È importante sottolineare che, per ottimizzare i tempi e le risorse, non sono stati eseguiti test su tutte le funzioni implementate, ma solo su un insieme rappresentativo, scelto per coprire le casistiche principali e i flussi più rilevanti del sistema.

I test di unità si concentrano sulla verifica di singole funzionalità, simulando richieste isolate e verificando che il sistema risponda con i risultati previsti. I test di integrazione, invece, mirano a verificare il corretto funzionamento delle interazioni tra i diversi servizi AWS utilizzati, come Lambda, DynamoDB e AppSync.

8.2.1 Testing microservizio AuthenticationService

Per il testing del microservizio di autenticazione si prevedono i seguenti casi di test:

1 Login con credenziali corrette:

- **Descrizione:** Verifica che il login con credenziali valide restituisca i token e le informazioni richieste in un formato corretto.
- **Tempo di esecuzione:** 2382 ms.
- **Risultato atteso:** I token (idToken, accessToken, refreshToken) sono restituiti correttamente e nel formato atteso (JWT), con altre proprietà (username, expireAt) valide.
- **Passed:**

2 Login con email non registrata:

- **Descrizione:** Verifica che il tentativo di login con un'email non registrata restituisca un errore.
- **Tempo di esecuzione:** 623 ms.
- **Risultato atteso:** L'errore "Incorrect username or password." viene lanciato.
- **Passed:**

3 Login con password non corretta:

- **Descrizione:** Verifica che il tentativo di login con una password non corretta restituisca un errore.
- **Tempo di esecuzione:** 615 ms.
- **Risultato atteso:** L'errore "Incorrect username or password." viene lanciato.
- **Passed:**

4 Verifica della validità del token corretto:

- **Descrizione:** Verifica che un access token valido sia decodificato correttamente e contenga le proprietà richieste.
- **Tempo di esecuzione:** 969 ms.
- **Risultato atteso:** Il token contiene le proprietà come sub, cognito:groups, iss, client_id, ecc., con valori validi.
- **Passed:**

5 Verifica della validità del token errato:

- **Descrizione:** Verifica che un token non valido restituisca un errore durante la verifica.
- **Tempo di esecuzione:** 71 ms.
- **Risultato atteso:** L'errore "invalid signature" viene lanciato.
- **Passed:**

```
PASS __tests__/_test_cases/unit/authenticationService.mjs (5.351 s)
  ✓ Login con credenziali corrette. (2382 ms)
  ✓ Login con email non registrata. (623 ms)
  ✓ Login con password non corretta. (615 ms)
  ✓ Verifica della validità del token corretto. (969 ms)
  ✓ Verifica della validità del token errato. (71 ms)

-----|-----|-----|-----|-----|-----|
File    | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 94.73 | 75 | 100 | 94.54 |
lib       | 94.11 | 100 | 100 | 93.33 |
lib.mjs   | 94.11 | 100 | 100 | 93.33 | 21
steps     | 95 | 75 | 100 | 95 |
given.mjs | 100 | 100 | 100 | 100 |
then.mjs  | 94.44 | 75 | 100 | 94.44 | 18
when.mjs | 87.5 | 100 | 100 | 87.5 | 23
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests:      5 passed, 5 total
Snapshots:  0 total
Time:       5.553 s
Ran all test suites.
```

Figura 8.1: Jest unit test microservizio AuthenticationService.

8.2.2 Testing microservizio UsersService

Per il testing del microservizio degli utenti si prevedono i seguenti casi di test:

1 Un Supervisore registra un nuovo utente:

- **Descrizione:** Verifica che un supervisore autenticato possa registrare un nuovo utente nel sistema.
- **Tempo di esecuzione:** 1826 ms.
- **Risultato atteso:** Il supervisore registra correttamente l'utente, e la funzione restituisce true.
- **Passed:**

2 Un Tutor prova a registrare un nuovo utente:

- **Descrizione:** Verifica che un tutor non autorizzato non possa registrare un nuovo utente.
- **Tempo di esecuzione:** 623 ms.
- **Risultato atteso:** L'operazione fallisce con l'errore: Not Authorized to access signUpNewUser on type Mutation.
- **Passed:** 

3 Un Supervisore registra un utente già esistente:

- **Descrizione:** Verifica che il tentativo di registrare un utente già esistente restituisca un errore.
- **Tempo di esecuzione:** 615 ms.
- **Risultato atteso:** L'operazione fallisce con l'errore: User account already exists.
- **Passed:** 

4 Un utente vuole ottenere la pagina del profilo di un collega:

- **Descrizione:** Verifica che un tutor autenticato possa visualizzare il profilo di un collega.
- **Tempo di esecuzione:** 969 ms.
- **Risultato atteso:** La funzione restituisce un oggetto che rappresenta il profilo del collega con le proprietà richieste (ad esempio, id, email, name, ecc.).
- **Passed:** 

5 Un utente vuole ottenere la pagina del profilo di un utente non collega:

- **Descrizione:** Verifica che un tutor non possa accedere al profilo di un utente non collegato.
- **Tempo di esecuzione:** 71 ms.
- **Risultato atteso:** L'operazione fallisce con l'errore: Unauthorized.
- **Passed:** 

6 Un Supervisore elimina un utente dal sistema:

- **Descrizione:** Verifica che un supervisore autenticato possa eliminare un utente esistente dal sistema.
- **Tempo di esecuzione:** 1565 ms.

- **Risultato atteso:** L'utente viene eliminato correttamente, e la funzione restituisce true.
- **Passed:**

7 Un Supervisore prova ad eliminare un utente non esistente sul sistema:

- **Descrizione:** Verifica che il tentativo di eliminare un utente non esistente restituisca un errore.
- **Tempo di esecuzione:** 71 ms.
- **Risultato atteso:** L'operazione fallisce con l'errore: User does not exist.
- **Passed:**

```
PASS __tests__/test_cases/unit/usersService.mjs (10.008 s)
✓ Un Supervisore registra un nuovo utente. (2595 ms)
✓ Un Tutor prova a registrare un nuovo utente. (868 ms)
✓ Un Supervisore registra un utente già esistente. (1221 ms)
✓ Un utente vuole ottenere la pagina del profilo di un collega. (1459 ms)
✓ Un utente vuole ottenere la pagina del profilo di un utente non collega. (890 ms)
✓ Un Supervisore elimina un utente dal sistema. (1275 ms)
✓ Un Supervisore prova ad eliminare un utente non esistente sul sistema. (1026 ms)

-----|-----|-----|-----|-----|-----|
File      | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #s
-----|-----|-----|-----|-----|-----|
All files | 71.07 |      50 |      65 |    72.64 |
lib        |   55 |  41.66 |      50 |  57.14 |
graphql.mjs | 68.96 |      40 |      60 |  71.42 | 22-30,44
lib.mjs    | 41.93 |      50 |      40 |  42.85 | 8-40,81
steps      | 86.88 |     100 |      80 |  86.88 |
given.mjs  |    80 |     100 |  66.66 |      80 | 21-24,33-53
when.mjs   | 96.15 |     100 |     100 |  96.15 |  24
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        10.19 s
Ran all test suites matching /usersService.mjs/i.
```

Figura 8.2: Jest unit testing microservizio UsersService.

Sono stati eseguiti anche i seguenti test di integrazione:

1 Verifica che l'utente esiste in DynamoDB e in Cognito:

- **Descrizione:** Testa la corretta registrazione di un nuovo utente. Verifica che l'utente sia stato creato in DynamoDB e Cognito.
- **Tempo di esecuzione:** 1826 ms.
- **Risultato atteso:**
 - **result** restituisce true.
 - L'utente esiste sia in DynamoDB che in Cognito.
- **Passed:**

2 Verifica che l'utente è stato eliminato da DynamoDB e da Cognito:

- **Descrizione:** Testa l'eliminazione di un utente dal sistema. Verifica che l'utente sia disattivato in DynamoDB e rimosso da Cognito.
- **Tempo di esecuzione:** 1565 ms.
- **Risultato atteso:**
 - **responseDelete** restituisce true.
 - L'utente risulta disattivato (active: "true") in DynamoDB.
 - L'utente non esiste più in Cognito.
- **Passed:**

```
PASS  __tests__/_test_cases/integration/usersService.test.mjs
  ✓ Verifica che l'utente esiste in DynamoDB e in Cognito. (1826 ms)
  ✓ Verifica che l'utente è stato eliminato da DynamoDB e da Cognito (1565 ms)

-----|-----|-----|-----|-----|-----|
File      | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 53.91 |     20 | 35.29 | 54.02 |
data/functions | 94.33 |    100 | 100 | 94.11 |
delete-user.mjs | 87.5 |    100 | 100 | 87.5 | 37,49 |
signup-new-user.mjs | 97.29 |    100 | 100 | 97.14 | 114 |
lib | 28.33 |    8.33 | 20 | 28.57 |
graphql.mjs | 13.79 |      0 | 0 | 14.28 | 8-30,37-63 |
lib.mjs | 41.93 |      50 | 40 | 42.85 | 8-40,81 |
steps | 48.07 |    37.5 | 26.31 | 48.07 |
given.mjs | 48.57 |    100 | 16.66 | 48.57 | 9-12,21-24,33-53,59-64,70-75 |
then.mjs | 54.28 |    16.66 | 28.57 | 54.28 | 15-43,70 |
when.mjs | 41.17 |    100 | 33.33 | 41.17 | 11-24,30-41,46-78,83-96 |

Test Suites: 1 passed, 1 total
Tests:       2 passed, 2 total
Snapshots:   0 total
Time:        4.715 s
Ran all test suites matching /__tests__\\_test_cases\\integration\\usersService.test.mjs/i.
```

Figura 8.3: Jest integration testing microservizio UsersService.

8.2.3 Testing microservizio PatientsService

Per il testing del microservizio dei pazienti si prevedono i seguenti casi di test:

1 Un Supervisore crea un nuovo paziente:

- **Descrizione:** Testa la creazione di un nuovo paziente da parte di un supervisore autenticato.
- **Tempo di esecuzione:** 459 ms.
- **Risultato atteso:**
 - La creazione restituisce true.
- **Passed:** 

2 Un Tutor prova a creare un nuovo paziente:

- **Descrizione:** Testa il tentativo di un tutor di creare un nuovo paziente, che non dovrebbe essere autorizzato.
- **Tempo di esecuzione:** 1087 ms.
- **Risultato atteso:**
 - Genera un errore con il messaggio: "Not Authorized to access createNewPaziente on type Mutation".
- **Passed:** 

3 Un Supervisore modifica un paziente:

- **Descrizione:** Testa la modifica di un paziente esistente da parte di un supervisore autenticato.
- **Tempo di esecuzione:** 1993 ms.
- **Risultato atteso:**
 - La modifica restituisce true.
- **Passed:** 

4 Un tutor tenta di modificare un paziente:

- **Descrizione:** Testa il tentativo di un tutor di modificare un paziente, che non dovrebbe essere autorizzato.
- **Tempo di esecuzione:** 917 ms.
- **Risultato atteso:**
 - Genera un errore con il messaggio: "Not Authorized to access editPaziente on type Mutation".
- **Passed:**

5 Viene modificato un paziente senza apportare nuove modifiche:

- **Descrizione:** Testa la gestione di una richiesta di modifica senza effettive modifiche al paziente.
- **Tempo di esecuzione:** 787 ms.
- **Risultato atteso:**
 - Genera un errore con il messaggio: "Non sono state apportate modifiche".
- **Passed:**

```
PASS __tests__/_test_cases/unit/patientsService.test.mjs (10.712 s)
✓ Un Supervisore crea un nuovo paziente. (4599 ms)
✓ Un Tutor prova a creare un nuovo paziente. (1087 ms)
✓ Un Supervisore modifica un paziente. (1993 ms)
✓ Un Tutor tenta di modificare un paziente. (917 ms)
✓ Viene modificato un paziente senza apportare nuove modifiche. (787 ms)

-----|-----|-----|-----|-----|-----|
File    | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 62.2 | 62.5 | 54.54 | 62.56 |
lib      | 63.01 | 56.25 | 54.54 | 64.7  |
graphql.mjs | 68.96 | 40 | 60 | 71.42 | 22-30,44
lib.mjs   | 59.09 | 83.33 | 50 | 60 | 8-40,81
steps     | 61.76 | 68.75 | 54.54 | 61.48 |
given.mjs | 69.62 | 68.75 | 66.66 | 69.23 | 15-18,27-30,39-59,87-101,175,185,203-204
when.mjs   | 50.87 | 100 | 40 | 50.87 | 24,30-41,46-78,83-96,101-105,111-115,166-169,174-178
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests:      5 passed, 5 total
Snapshots:  0 total
Time:       10.894 s
Ran all test suites matching __tests__\\_test_cases\\unit\\patientsService.test.mjs.
```

Figura 8.4: Jest unit testing microservizio PatientsService.

Sono stati eseguiti anche i seguenti test di integrazione:

1 Verifica che il paziente, le sessioni e l'associazione tutor-paziente esistono su DynamoDB:

- **Descrizione:** Verifica che un paziente, le sessioni associate e il legame tutor-paziente siano correttamente presenti nelle rispettive tabelle su DynamoDB.
- **Tempo di esecuzione:** 991 ms.
- **Risultato atteso:**
 - Il paziente deve esistere nella tabella PazientiTable (pazienteExists.result === true).
 - Le sessioni devono esistere nella tabella SessionsTable (sessionsExists.result === true).
 - L'associazione tutor-paziente deve esistere nella tabella PazientiTutorTable (tutorPazienteExists.result === true).
- **Passed:**

```
PASS _tests_\test_cases\integration\patientsService.test.mjs
✓ Verifica che il paziente, le sessioni e l'associazione tutor-paziente esistono su DynamoDB (991 ms)

-----|-----|-----|-----|-----|-----|
File      | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 50.47 | 39.28 | 41.5   | 50       | 
data/functions | 92.1 | 50 | 100 | 97.14 | 
create-new-paziente.mjs | 92.1 | 50 | 100 | 97.14 | 122
lib | 41.09 | 18.75 | 27.27 | 41.17 | 
graphql.mjs | 13.79 | 0 | 0 | 14.28 | 8-30,37-63
lib.mjs | 59.09 | 50 | 50 | 60 | 8-40,81
steps | 46.03 | 46.42 | 36.11 | 44.67 | 
given.mjs | 55.69 | 62.5 | 41.66 | 55.12 | 15-18,27-30,39-59,65-70,76-81,87-101,123,162-164,175,185,203-204
then.mjs | 51.51 | 25 | 50 | 48.38 | 15-43,50-69,79-94,124,159-165,200-201
when.mjs | 26.31 | 100 | 10 | 26.31 | 11-24,30-41,46-78,83-96,101-105,111-115,121-133,138-150,155-169
-----|-----|-----|-----|-----|-----|
Test Suites: 1 passed, 1 total
Tests: 1 passed, 1 total
Snapshots: 0 total
Time: 2.472 s, estimated 3 s
Ran all test suites matching _\tests_\test_cases\integration\patientsService.test.mjs.i.
```

Figura 8.5: Jest integration testing microservizio PatientsService.

8.2.4 Testing microservizio ReportsService

Per il testing del microservizio dei report si prevedono i seguenti casi di test:

1 Un utente crea un nuovo report:

- **Descrizione:** Verifica che un utente autenticato possa creare un nuovo report per un paziente di sua responsabilità.
- **Tempo di esecuzione:** 1923 ms.
- **Risultato atteso:** La creazione del report restituisce true.
- **Passed:**

2 Un utente prova a creare un report su un paziente non preso in carico:

- **Descrizione:** Verifica che un utente non possa creare un report per un paziente che non gli è stato assegnato.
- **Tempo di esecuzione:** 1331 ms.
- **Risultato atteso:** La creazione del report lancia un'eccezione con il messaggio Is not your patient.
- **Passed:** 

3 Un utente modifica un proprio report:

- **Descrizione:** Verifica che un utente possa modificare i dettagli di un report di sua proprietà.
- **Tempo di esecuzione:** 1153 ms.
- **Risultato atteso:** Il report restituisce i valori modificati per reportId, description e contenuto.
- **Passed:** 

4 Un utente prova a modificare un report di un paziente non preso in carico:

- **Descrizione:** Verifica che un utente non possa modificare un report che non appartiene a lui.
- **Tempo di esecuzione:** 1187 ms.
- **Risultato atteso:** La modifica del report lancia un'eccezione con il messaggio Unauthorized: The report does not belong to the specified tutor.
- **Passed:** 

5 Un utente prova a modificare un proprio report con campi non validi:

- **Descrizione:** Verifica che un utente non possa modificare un report con campi obbligatori mancanti o invalidi.
- **Tempo di esecuzione:** 1046 ms.
- **Risultato atteso:** La modifica del report lancia un'eccezione con il messaggio Missing required fields: reportId, description, contenuto.
- **Passed:** 

```

PASS __tests__/test_cases/unit/reportsService.test.mjs (7.865 s)
✓ Un utente crea un nuovo report. (1923 ms)
✓ Un utente prova a creare un report su un paziente non preso in carico. (1331 ms)
✓ Un utente modifica un proprio report. (1153 ms)
✓ Un utente prova a modificare un report non suo. (1187 ms)
✓ Un utente prova a modificare un proprio report con campi non validi. (1046 ms)

-----|-----|-----|-----|-----|-----|
File      | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 39.89 | 20.96 | 40.9   | 40.17 |
lib        | 31.5   | 25      | 27.27  | 33.82 |
graphql.mjs | 68.96 | 40      | 60     | 71.42 | 22-30,44
lib.mjs    | 6.81   | 0       | 0      | 7.5   | 8-40,74-107,115-143
steps      | 41.97 | 19.56  | 43.63 | 41.69 |
given.mjs  | 50     | 26.92  | 61.53 | 48.48 | 15-18,27-30,39-59,87-101,118-141,162
then.mjs   | 30.66 | 14.28  | 33.33 | 32.39 | 22,34,43,50-69,79-94,105-124,134-165
when.mjs   | 38.75 | 0       | 21.42 | 38.75 | 24,30-41,46-78,83-96,101-105,111-115

Test Suites: 1 passed, 1 total
Tests:      5 passed, 5 total
Snapshots:  0 total
Time:       8.046 s, estimated 13 s
Ran all test suites matching __tests__\test_cases\unit\reportsService.test.mjs/i.

```

Figura 8.6: Jest unit testing microservizio ReportsService.

Sono stati eseguiti anche i seguenti test di integrazione:

1 Verifica che il report esiste su DynamoDB:

- **Descrizione:** Verifica che un report creato da un supervisore per un paziente a lui associato venga correttamente salvato nella tabella DynamoDB.
- **Tempo di esecuzione:** 2083 ms.
- **Risultato atteso:** La funzione di verifica restituisce true, confermando che il report esiste in DynamoDB.
- **Passed:**

```

PASS __tests__/test_cases/integration/reportsService.test.mjs
✓ Verifica che il report esiste su DynamoDB (2083 ms)

-----|-----|-----|-----|-----|-----|
File      | %Stmts | %Branch | %Funcs | %Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 28.2   | 11.76   | 21.73  | 28.95 |
data/functions | 87.5   | 66.66   | 100    | 86.36 | 12,32,67
create-new-report.mjs | 87.5   | 66.66   | 100    | 86.36 | 12,32,67
lib        | 9.58   | 0       | 0      | 10.29 | 8-30,37-63
graphql.mjs | 13.79  | 0       | 0      | 14.28 | 8-30,37-63
lib.mjs    | 6.81   | 0       | 0      | 7.5   | 8-40,74-107,115-143
steps      | 27.98  | 8.69   | 21.81 | 28.97 |
given.mjs  | 22.46  | 3.84   | 15.38 | 23.48 | 15-18,27-30,39-59,76-81,87-1
then.mjs   | 37.33  | 21.42  | 40    | 39.43 | 22,34,43,50-69,79-94,105-124
when.mjs   | 28.75  | 0       | 14.28 | 28.75 | 24,30-41,46-78,83-96,101-105

Test Suites: 1 passed, 1 total
Tests:      1 passed, 1 total
Snapshots:  0 total
Time:       3.564 s
Ran all test suites matching __tests__\test_cases\integration\reportsService.test.mjs/i.

```

Figura 8.7: Jest integration testing microservizio ReportsService.

8.2.5 Testing microservizio Attività

Per il testing del microservizio delle attività si prevedono i seguenti casi di test:

1 Un Supervisore ottiene le attività sull'autenticazione:

- **Descrizione:** Un supervisore richiede i log delle attività relative all'autenticazione.
- **Tempo di Esecuzione:** 4045 ms.
- **Risultato Atteso:** I log delle attività contengono proprietà come id, timestamp, operationName, author, ecc.
- **Passed:**

2 Un Supervisore ottiene le attività sugli utenti:

- **Descrizione:** Un supervisore richiede i log delle attività relative agli utenti.
- **Tempo di Esecuzione:** 1680 ms.
- **Risultato Atteso:** I log delle attività contengono proprietà come id, timestamp, operationName, author, ecc.
- **Passed:**

3 Un Supervisore ottiene le attività sui pazienti:

- **Descrizione:** Un supervisore richiede i log delle attività relative ai pazienti.
- **Tempo di Esecuzione:** 1582 ms.
- **Risultato Atteso:** I log delle attività contengono proprietà come id, timestamp, operationName, author, ecc.
- **Passed:**

4 Un Supervisore ottiene le attività sui report:

- **Descrizione:** Un supervisore richiede i log delle attività relative ai report.
- **Tempo di Esecuzione:** 1856 ms.
- **Risultato Atteso:** I log delle attività contengono proprietà come id, timestamp, operationName, author, ecc.
- **Passed:**

5 Un Tutor prova ad ottenere le attività:

- **Descrizione:** Un tutor tenta di accedere ai log delle attività ma non ha l'autorizzazione necessaria.

- **Tempo di Esecuzione:** 917 ms.
- **Risultato Atteso:** La richiesta genera un errore con messaggio: "Not Authorized to access getLogs on type Query".
- **Passed:**

```
PASS __tests__/_test_cases/unit/logsService.test.mjs (11.348 s)
✓ Un Supervisore ottiene le attività sull'autenticazione. (4045 ms)
✓ Un Supervisore ottiene le attività sugli utenti. (1680 ms)
✓ Un Supervisore ottiene le attività sui pazienti. (1582 ms)
✓ Un Supervisore ottiene le attività sui report. (1856 ms)
✓ Un Tutor prova ad ottenere le attività. (917 ms)

-----|-----|-----|-----|-----|-----|
File    | % Stmt | % Branch | % Funcs | % Lines | Uncovered Line #
-----|-----|-----|-----|-----|-----|
All files | 26.5 | 16.12 | 18.18 | 27.63 |
lib | 31.5 | 25 | 27.27 | 33.82 |
graphql.mjs | 68.96 | 40 | 60 | 71.42 | 22-30,44
lib.mjs | 6.81 | 0 | 0 | 7.5 | 8-40,74-107,115-143
steps | 25.25 | 13.04 | 16.36 | 26.14 |
given.mjs | 18.84 | 0 | 7.69 | 19.69 | 15-18,27-30,39-59,87-101,118-141,162-165
then.mjs | 30.66 | 14.28 | 33.33 | 32.39 | 22,34,43,50-69,79-94,105-124,134-165,166
when.mjs | 31.25 | 66.66 | 14.28 | 31.25 | 24,30-41,46-78,83-96,101-105,111-115,116

Test Suites: 1 passed, 1 total
Tests: 5 passed, 5 total
Snapshots: 0 total
Time: 11.595 s, estimated 14 s
Ran all test suites matching __tests__/_test_cases/unit/logsService.test.mjs/i.
```

Figura 8.8: Jest unit testing microservizio ReportsService.

8.3 Test funzionali

Il testing funzionale ha l'obiettivo di verificare che l'intero sistema soddisfi i requisiti previsti, simulando flussi di utilizzo realistici dal punto di vista dell'utente. Questo tipo di testing si concentra sull'assicurare che ogni funzionalità dell'applicazione operi correttamente all'interno del contesto completo. Nel nostro caso, il testing funzionale mira a validare i principali flussi di utilizzo dell'applicazione, come il login e la visualizzazione dei dati. Per garantire una copertura ottimale e rappresentativa, il testing si concentra sui casi più rilevanti e sui flussi principali, analizzando il comportamento del sistema in scenari realistici.

8.3.1 Testing operazione di login

- **Tentativo di login con un account non registrato:**
 - **Condizioni di validità**
Un utente vuole che accedere al sistema deve essere stato registrato da un Supervisore in precedenza.
 - **Precondizioni**
1) L'utente deve essere stato registrato da un Supervisore.

TC	Descrizione	Input	Output
TC1	Inserisce l'email di un account non registrato.	email: "accountnonregistrato@gmail.com"	error: "L'utente non esiste"

- **Tentativo di login con un account registrato ma password errata:**

 - **Condizioni di validità**

Un utente che vuole accedere al sistema deve inserire le credenziali corrette.

 - **Precondizioni**

1) L'utente deve essere stato registrato da un Supervisore.

TC	Descrizione	Input	Output
TC2	Inserisce una password non valida.	password: "Pippo123!"	error: "Verifica che l'email o la password siano validi."

- **Tentativo di login con credenziali corrette:**

 - **Condizioni di validità**

Un utente che vuole accedere al sistema deve inserire le credenziali ottenute tramite l'email dopo la registrazione da parte di un Supervisore.

 - **Precondizioni**

1) L'utente deve essere stato registrato da un Supervisore.

TC	Descrizione	Input	Output
TC3	Inserisce le credenziali d'accesso ricevute per email: email e password.	email: "utenteregistrato@gmail.com" password: "Pluto123!"	success: "Login effettuato con successo."

8.3.2 Testing operazione di creazione utente

- **Tentativo di creazione utente con dati non validi:**

 - **Condizioni di validità**

Un Supervisore che vuole registrare un utente nel sistema deve inserire correttamente i dati dell'utente nell'apposito form.

 - **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC4	Inserisce un codice fiscale non valido.	codfis: “SCGFNC97PABCDEFG”	error: “Verifica che tutti i campi siano compilati correttamente.”

- **Tentativo di creazione utente già esistente:**

- **Condizioni di validità**

Un Supervisore che vuole registrare un utente nel sistema deve inserire correttamente i dati dell’utente nell’apposito form e l’email inserita non deve essere già registrata.

- **Precondizioni**

1) L’utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC5	Inserisce un email già registrata.	email: “accountregistrato@gmail.com”	error: “Utente già registrato.”

- **Tentativo di creazione utente con dati corretti:**

- **Condizioni di validità**

Un Supervisore che vuole registrare un utente nel sistema deve inserire correttamente i dati dell’utente nell’apposito form e l’email inserita non deve essere già registrata.

- **Precondizioni**

1) L’utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC6	Inserisce le informazioni corrette.	Informazioni utente (nome, cognome, etc.)	success: “Utente registrato con successo.”

8.3.3 Testing operazione di ottenimento utente

- Tentativo di ottenimento di utente collega:

- **Condizioni di validità**

Un Tutor vuole ottenere la pagina del profilo di un collega, cioè, che presentino almeno un paziente in comune.

- **Precondizioni**

1) Il tutor e l'utente da visualizzare devono essere colleghi.

TC	Descrizione	Input	Output
TC7	Accede alla pagina del profilo del collega.		Ottenimento della pagina del profilo

- Tentativo di ottenimento di utente non collega:

- **Condizioni di validità**

Un Tutor vuole ottenere la pagina del profilo di un collega, cioè, che presentino almeno un paziente in comune.

- **Precondizioni**

1) Il tutor e l'utente da visualizzare devono essere colleghi.

TC	Descrizione	Input	Output
TC8	Accede alla pagina del profilo dell'utente.		Reindirizzamento alla dashboard.

8.3.4 Testing operazione di eliminazione utente

- Tentativo di eliminazione utente:

- **Condizioni di validità**

Un Supervisore che vuole eliminare un utente dal sistema deve effettuare la conferma dell'eliminazione scrivendo “conferma”.

- **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC9	Nella finestra di eliminazione deve inserire una parola per confermare l'eliminazione.	testo: “conferma”.	success: “L'utente è stato eliminato con successo.”

8.3.5 Testing operazione di creazione paziente

- Tentativo di creazione di un paziente con informazioni non valide:

- **Condizioni di validità**

Un Supervisore che vuole registrare un paziente nel sistema deve inserire correttamente i dati nell'apposito form.

- **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC10	Inserisce un codice fiscale non valido.	codfis: "MCLFLC98BABCDEFG"	error: "Verifica che tutti i campi siano compilati correttamente."

- Tentativo di creazione di un paziente con informazioni valide:

- **Condizioni di validità**

Un Supervisore che vuole registrare un paziente nel sistema deve inserire correttamente i dati nell'apposito form.

- **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC11	Inserisce le informazioni del paziente.	Informazioni del paziente: (nome, cognome, etc.)	error: "Verifica che tutti i campi siano compilati correttamente."

8.3.6 Testing operazione di modifica paziente

- Tentativo di modifica di un paziente con campi non validi:

- **Condizioni di validità**

Un Supervisore che vuole modificare un paziente nel sistema deve compilare l'apposito form con campi validi e modificati.

- **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC12	Inserisce un codice fiscale non valido.	codfis: "MCLFLC98BABCDEFG"	error: "Verifica che tutti i campi siano compilati correttamente."

- **Tentativo di modifica di un paziente con campi validi:**

- **Condizioni di validità**

Un Supervisore che vuole modificare un paziente nel sistema deve compilare l'apposito form con campi validi e modificati.

- **Precondizioni**

1) L'utente deve essere un Supervisore.

TC	Descrizione	Input	Output
TC13	Inserisce le informazioni del paziente.	Informazioni del paziente: (nome, cognome, etc.)	success: "Il paziente è stato modificato con successo."

8.3.7 Testing operazione di creazione report

- **Tentativo di creazione di un report con informazioni non valide:**

- **Condizioni di validità**

Un utente che vuole creare un report nel sistema deve inserire correttamente i dati nell'apposito form.

- **Precondizioni**

Nessuna.

TC	Descrizione	Input	Output
TC14	Non inserisce un contenuto nel report.		error: "Verifica che tutti i campi siano compilati correttamente."

- **Tentativo di creazione di un report con informazioni valide:**

- **Condizioni di validità**

Un utente che vuole creare un report nel sistema deve inserire correttamente i dati nell'apposito form.

- **Precondizioni**

Nessuna.

TC	Descrizione	Input	Output
TC15	Inserisce tutte le informazioni del report.	Informazioni del report: (titolo, contenuto, paziente)	success: "Report creato con successo."

8.3.8 Testing operazione di modifica report

- Tentativo di modifica di un report con campi non validi:

- **Condizioni di validità**

Un Supervisore che vuole modificare un report nel sistema deve essere proprietario del report e compilare l'apposito form con campi validi e modificati.

- **Precondizioni**

1) L'utente deve essere il proprietario del report.

TC	Descrizione	Input	Output
TC16	Non effettua modifiche.		warn: "Non sono state rilevate modifiche al report".

- Tentativo di modifica di un report con campi validi:

- **Condizioni di validità**

Un Supervisore che vuole modificare un report nel sistema deve essere proprietario del report e compilare l'apposito form con campi validi e modificati.

- **Precondizioni**

1) L'utente deve essere il proprietario del report.

TC	Descrizione	Input	Output
TC17	Inserisce le modifiche sul report.	Informazioni del report modificate: (titolo, contenuto, paziente)	success: "Il report è stato modificato con successo."

8.4 Test non funzionali

I test non funzionali rivestono un ruolo cruciale nel garantire che un sistema soddisfi non solo le sue funzionalità di base, ma anche le aspettative relative a qualità, prestazioni e affidabilità. Questi test si concentrano su aspetti come la sicurezza, la scalabilità, l'usabilità, l'affidabilità e la manutenibilità, che sono fondamentali per assicurare un'esperienza utente efficiente e affidabile. Diversamente dai test funzionali, i test non funzionali valutano il "come" il sistema opera piuttosto che il "cosa" fa, aiutando a identificare eventuali criticità o limiti in condizioni operative reali.

Nella nostra applicazione, i test non funzionali sono stati pianificati e condotti seguendo scenari di qualità predefiniti, descritti nei capitoli precedenti. Tali scenari delineano stimoli, risposte attese e parametri di misura, consentendo di verificare in modo strutturato la conformità del sistema ai requisiti non funzionali.

8.4.1 Sicurezza

RNF01 - L'applicazione deve impedire l'accesso a utenti non autenticati e proteggere i dati trasmessi.

Verifica: Per testare questo requisito, possiamo simulare richieste non autorizzate agli endpoint protetti del backend utilizzando il tool Postman. Inviando richieste senza fornire credenziali valide o senza token di autenticazione, possiamo osservare la risposta del sistema e verificare che restituisca un errore appropriato senza divulgare informazioni sensibili.

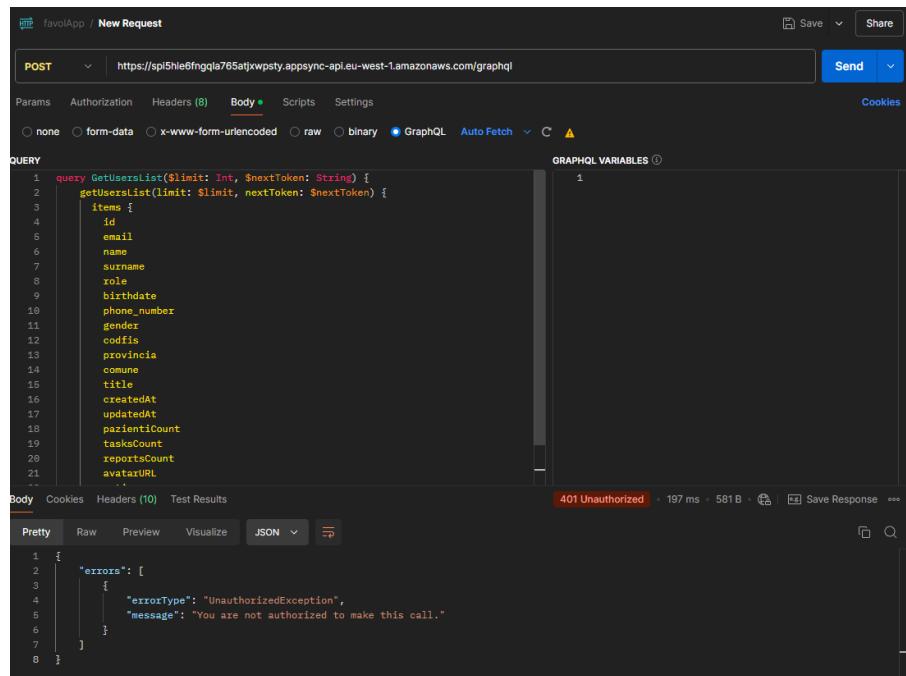


Figura 8.9 Richiesta negata ad una funzione GraphQL senza token di autenticazione.

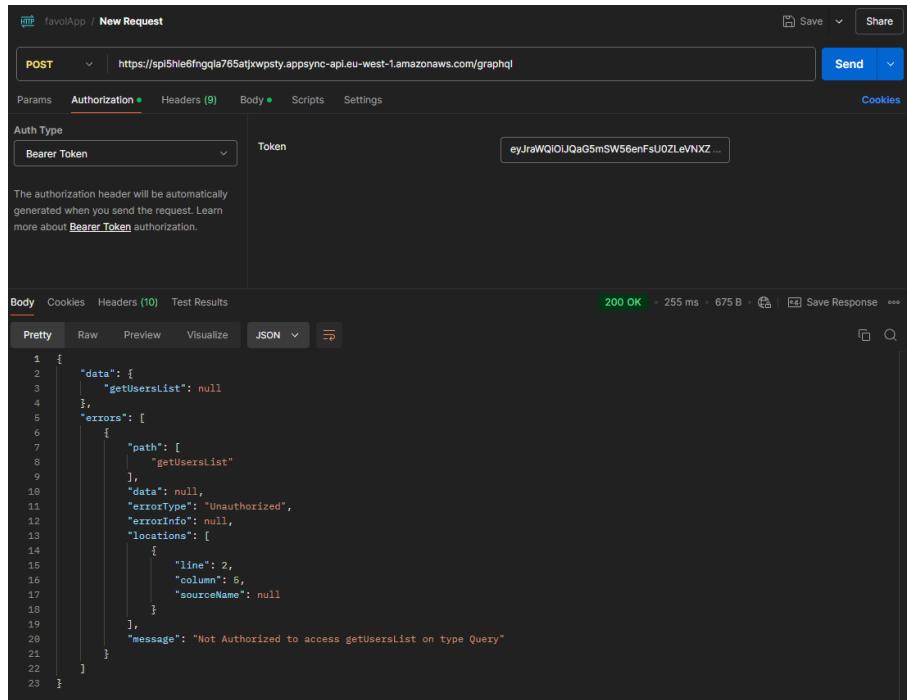


Figura 8.10 Richiesta negata ad una funzione GraphQL riservata ai supervisori da parte di un tutor.

RNF02 - I dati trasmessi devono essere protetti da accessi non autorizzati.

Verifica: Possiamo analizzare il traffico di rete tra il client e il server utilizzando il tool Wireshark. Durante una sessione di comunicazione tra il frontend e il backend, verifichiamo che i dati siano cifrati tramite protocolli come TLS e che non siano leggibili in chiaro.

- **Identificazione dell'IP del backend**
 - Utilizzando il comando ping www.favolapp.it, è stato determinato che il dominio dell'applicazione è risolto nell'indirizzo IP 18.65.64.29, un endpoint gestito tramite Amazon CloudFront.
 - Questo passaggio è stato necessario per identificare l'IP del server distribuito da CloudFront per filtrare il traffico di rete relativo all'applicazione utilizzando Wireshark.
- **Monitoraggio del traffico con Wireshark**
 - Una volta identificato l'IP del server, è stato avviato Wireshark e applicato il filtro:Questo filtro ha permesso di isolare il traffico tra il client (frontend) e il server CloudFront.
- **Analisi del traffico TLS**
 - Durante l'analisi, è stato confermato che la comunicazione utilizza il protocollo TLSv1.2, come indicato nei pacchetti catturati.
 - I pacchetti contenenti dati applicativi erano marcati come Encrypted Application Data nella sezione Transport Layer Security (TLS), evidenziando che i dati trasmessi sono cifrati.
 - Il contenuto dei pacchetti è risultato incomprensibile nel riquadro inferiore di Wireshark, dimostrando che i dati non sono trasmessi in chiaro e che un attaccante non può intercettarli senza la chiave di decriptazione.

• Conferma della cifratura

- La porta utilizzata per la comunicazione è la 443, standard per HTTPS, il che rafforza ulteriormente la validità del protocollo TLS utilizzato.
- L'assenza di dati in chiaro durante la cattura evidenzia che il requisito RNF02 è stato soddisfatto: i dati intercettati risultano cifrati, rendendoli inutilizzabili da eventuali attacchi man-in-the-middle.

```
C:\Users\Felice>ping www.favolapp.it

Esecuzione di Ping d1q7a8jokscqvz.cloudfront.net [18.65.64.45] con 32 byte di dati:
Risposta da 18.65.64.45: byte=32 durata=11ms TTL=247
Risposta da 18.65.64.45: byte=32 durata=11ms TTL=247
Risposta da 18.65.64.45: byte=32 durata=11ms TTL=247
Risposta da 18.65.64.45: byte=32 durata=12ms TTL=247

Statistiche Ping per 18.65.64.45:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 11ms, Massimo = 12ms, Medio = 11ms
```

Figura 8.11: Rilevamento dell'ip di CloudFront tramite ping.

No.	Time	Source	Destination	Protocol	Length	Info
1138	4.334480	192.168.1.13	18.65.64.29	TCP	54	59791 > +443 [FIN, ACK] Seq=1 Ack=2 Win=145 Len=0
1144	4.345857	192.168.1.13	18.65.64.29	TCP	54	59790 > +59790 [FIN, ACK] Seq=1 Ack=2 Win=145 Len=0
1144	4.345882	192.168.1.13	18.65.64.29	TCP	54	59790 > +443 [ACK] Seq=2 Ack=2 Win=145 Len=0
3655	15.664533	192.168.1.13	18.65.64.29	TCP	55	59791 > +443 [ACK] Seq=1 Ack=1 Win=145 Len=1
3668	15.675377	18.65.64.29	192.168.1.13	TCP	66	443 > 59791 [ACK] Seq=1 Ack=2 Win=145 Len=1 SLE=1 SRE=2
11909	60.694189	192.168.1.13	18.65.64.29	TCP	55	59791 > +443 [ACK] Seq=1 Ack=2 Win=145 Len=1 SLE=1 SRE=2
11995	60.694189	192.168.1.13	18.65.64.29	TCP	66	[TCP keep-alive ACK] 443 > 59791 [ACK] Seq=1 Ack=2 Win=145 Len=0 SLE=1 SRE=2
20173	185.694695	192.168.1.13	18.65.64.29	TCP	55	[TCP keep-alive] 59791 > +443 [ACK] Seq=1 Ack=1 Win=145 Len=1
20201	185.694695	192.168.1.13	18.65.64.29	TCP	66	[TCP keep-alive ACK] 443 > 59791 [ACK] Seq=1 Ack=2 Win=145 Len=0 SLE=1 SRE=2
27449	158.718989	192.168.1.13	18.65.64.29	TCP	55	[TCP keep-alive] 59791 > +443 [ACK] Seq=1 Ack=1 Win=145 Len=1
27449	158.718994	18.65.64.29	192.168.1.13	TCP	66	[TCP keep-alive ACK] 443 > 59791 [ACK] Seq=1 Ack=2 Win=145 Len=0 SLE=1 SRE=2
30119	165.631144	18.65.64.29	192.168.1.13	TLSv1.2	93	Application Data
30123	165.631144	192.168.1.13	18.65.64.29	TCP	54	59791 > +443 [FIN, ACK] Seq=48 Ack=2 Win=145 Len=0
30122	165.631250	192.168.1.13	18.65.64.29	TCP	54	59791 > +443 [ACK] Seq=2 Ack=41 Win=145 Len=0
30122	165.631642	192.168.1.13	18.65.64.29	TCP	54	59791 > +443 [FIN, ACK] Seq=41 Ack=41 Win=145 Len=0
-	30125	165.643088	18.65.64.29	TCP	54	443 > 59791 [ACK] Seq=41 Ack=3 Win=145 Len=0

.... .0 = SYN: Absent [Completeness Flags: -FO-] [TCP Segments Len: 99]	00000 34 2e b7 fe b9 d8 08 06 05 16 4e 80 00 00 45 00 4... N - E: 00100 00 4f ab 4a 00 00 f7 86 04 40 12 41 40 1d c8 a8 0... K AB: 00200 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0... P 00300 00 91 49 b3 00 00 17 03 03 00 22 23 00 43 00 00 00 0... I - - - - - 00400 0f 91 49 3b 7d 50 10 42 a8 5b ff 63 05 15 c5 e5 0... 16:17 18:19 00500 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0... 19:20 20:21
> Flags: B(RST) (PSH, ACK) Window: 145 [Calculated window size: 145] [Window size scaling factor: -1 (unknown)] Checksum: 0x0000 (verified) [Checksum Status: Unverified] Urgent Pointer: 0 [Urgent pointer: 0] > [SQNACK analysis] TCP payload (39 bytes) ▼ Transport Layer Security ▼ TLSv1.2/TLSv1.3 Layer Application Data Protocol: Hypertext Transfer Protocol Content Type: Application Data (23) Version: TLS 1.2 (0x0003) Length: 39 Encrypted Application Data: 22ec489467ef91d41b7d381042aa3b8ff0105f3c5e6d5f8eb94e0f2e49675ed0eb2321 [Application Data Protocol: Hypertext Transfer Protocol]	

Figura 8.12: Wireshark mostra i dati criptati tramite certificato TLS.

8.4.2 Scalabilità

RNF03 - Il sistema deve gestire un aumento delle richieste senza degradare le prestazioni.

Verifica: Per simulare un carico elevato utilizziamo il tool k6. Questo strumento permette di generare un carico simulato e monitorare il tempo medio di risposta e la percentuale di richieste servite senza timeout. Abbiamo configurato un test per raddoppiare il numero di richieste rispetto al normale utilizzo e abbiamo osservato i risultati.

Abbiamo eseguito un test di carico sulla nostra applicazione utilizzando k6 per verificare le prestazioni effettuando varie richieste GraphQL. Lo script ha simulato fino a 50 utenti virtuali simultanei, che hanno eseguito richieste ripetute per un totale di 4 minuti, suddivisi in tre fasi: un aumento graduale del carico, un periodo di carico costante e una riduzione graduale. Durante il test, sono state inviate richieste al server e sono stati raccolti dati sulle prestazioni, come il tempo medio di risposta e il tasso di successo.

Metrica	Valore	Descrizione
Checks	100.00% (16694/16694)	Percentuale di controlli superati (tutti i check sono stati soddisfatti).
Dati Ricevuti (data_received)	31 MB (128 kB/s)	Volume totale di dati ricevuti dal server durante il test.
Dati Inviati (data_sent)	5.1 MB (21 kB/s)	Volume totale di dati inviati al server.
http_req_duration	avg=83.97ms, p(95)=157.69ms	Tempo medio e massimo al 95° percentile per completare una richiesta HTTP.
http_req_failed	0.00%	Percentuale di richieste HTTP fallite (nessuna richiesta è fallita).
http_reqs	8347	Numero totale di richieste HTTP eseguite.
iteration_duration	avg=1.08s, max=1.33s	Durata media e massima per completare un'iterazione dello script.
Virtual Users (VUs)	1 min, 50 max	Numero di utenti virtuali attivi durante il test (fino a 50 simultanei).
Durata Test	4m0s	Durata totale del test, inclusa la fase di ramp-up e ramp-down.

Tabella 8.1: Metriche e valori del load test eseguito sull'applicazione.



```

execution: local
script: loadtest.js
output: InfluxDBv1 (http://localhost:8086)

scenarios: (100.00%) 1 scenario, 50 max VUs, 4m30s max duration (incl. graceful stop):
  * default: Up to 50 looping VUs for 4m0s over 3 stages (gracefulRampDown: 30s, gracefulStop: 30s)

✓ is status 200
✓ response time < 3s

checks.....: 100.00% 16694 out of 16694
data_received.....: 31 MB 128 kB/s
data_sent.....: 5.1 MB 21 kB/s
http_req_blocked.....: avg=99.71µs min=0s med=0s max=60.33ms p(90)=0s p(95)=0s
http_req_connecting.....: avg=36.79µs min=0s med=0s max=9.51ms p(90)=0s p(95)=0s
✓ http_req_duration.....: avg=83.97ms min=60.17ms med=75.46ms max=303.53ms p(90)=111.86ms p(95)=157.69ms
  { expected_response:true }.....: avg=83.97ms min=60.17ms med=75.46ms max=303.53ms p(90)=111.86ms p(95)=157.69ms
✓ http_req_failed.....: 0.00% 0 out of 8347
http_req_receiving.....: avg=147.33µs min=0s med=0s max=5.61ms p(90)=639.62µs p(95)=866.51µs
http_req_sending.....: avg=145.72µs min=0s med=0s max=7.64ms p(90)=509.29µs p(95)=627.04µs
http_req_tls_handshaking.....: avg=59.55µs min=0s med=0s max=33.77ms p(90)=0s p(95)=0s
http_req_waiting.....: avg=83.68ms min=60.17ms med=75.18ms max=303.53ms p(90)=111.7ms p(95)=157.28ms
http_reqs.....: 8347 34.630559/s
iteration_duration.....: avg=1.08s min=1.06s med=1.07s max=1.33s p(90)=1.11s p(95)=1.15s
iterations.....: 8347 34.630559/s
vus.....: 1 min=1 max=50
vus_max.....: 50 min=50 max=50

running (4m01.0s), 00/50 VUs, 8347 complete and 0 interrupted iterations
default ✓ [=====] 00/50 VUs 4m0s

```

Figura 8.14 Screen dei risultati del load test ottenuti da k6.

8.4.3 Usabilità

RNF04 - L'applicazione deve essere facilmente comprensibile e navigabile per gli utenti.

Verifica: Per testare l'usabilità dell'applicazione, utilizziamo Lighthouse, un tool integrato nei browser come Google Chrome, che permette di analizzare automaticamente aspetti legati all'usabilità e all'accessibilità. Lighthouse fornisce un report dettagliato sulla navigabilità, evidenziando eventuali criticità come problemi di layout, tempi di caricamento non ottimali o difficoltà nell'interazione con l'interfaccia utente. Questo approccio ci consente di ottenere metriche oggettive e riproducibili, senza necessità di coinvolgere utenti esterni.

Controlli superati:

- Il report evidenzia che 16 controlli di accessibilità sono stati superati, indicando una buona conformità con i requisiti di base per un'interfaccia accessibile.

Ulteriori verifiche manuali richieste:

- Sono presenti 10 elementi che necessitano di verifiche manuali, poiché i test automatici non possono coprire tutte le aree dell'accessibilità, come ad esempio la qualità dei contenuti testuali o la validità delle descrizioni vocali.

Elementi non applicabili:

- 40 elementi non sono stati considerati rilevanti per l'attuale contesto applicativo, riducendo così il campo delle aree testabili.

Il punteggio di 91/100 dimostra che l'applicazione è progettata con una buona attenzione all'accessibilità.

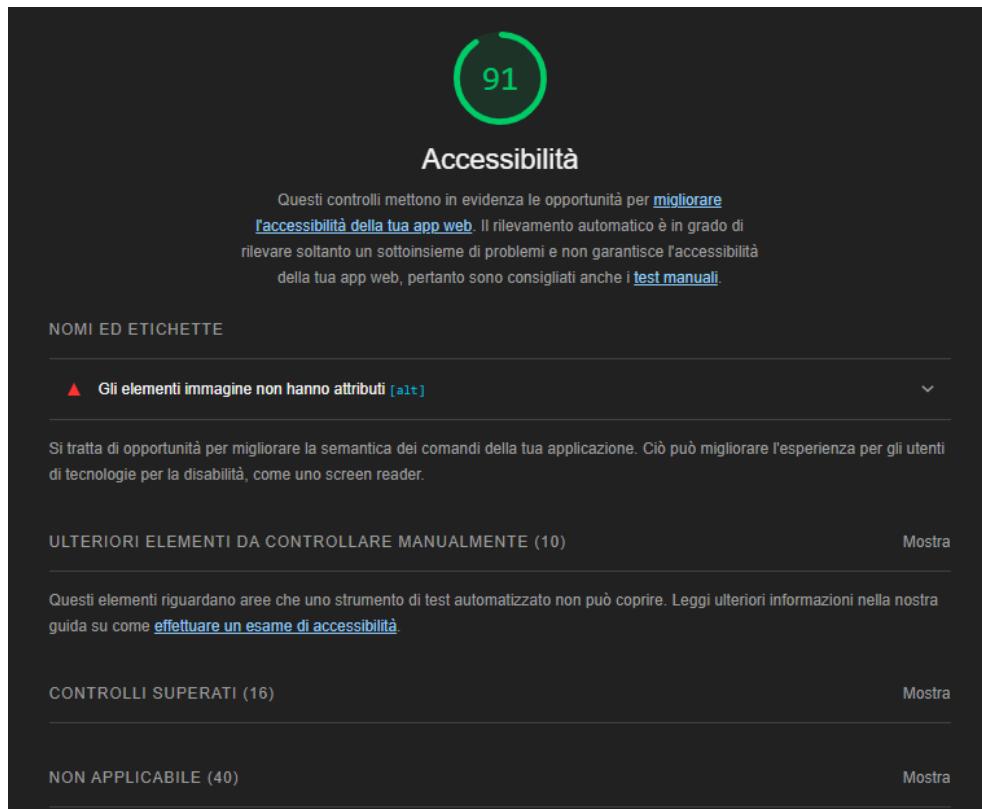


Figura 8.15 Risultato del test automatico con il tool Lighthouse.

8.4.4 Affidabilità

RNF05 - Il sistema deve garantire una disponibilità continua durante le ore di utilizzo principali.

Verifica: La nostra applicazione è ospitata su un'infrastruttura AWS, che garantisce elevati livelli di affidabilità e un uptime superiore al 99,99% grazie ai suoi servizi completamente gestiti. Poiché AWS assicura la disponibilità attraverso meccanismi integrati come il bilanciamento del carico, la replica multi-region e il failover automatico, non è necessario eseguire test manuali di disponibilità. Tuttavia, continuiamo a monitorare l'infrastruttura tramite AWS CloudWatch per rilevare e intervenire prontamente su eventuali anomalie.

8.4.5 Manutenibilità

RNF06 - Le modifiche al codice devono poter essere effettuate rapidamente e senza introdurre regressioni.

Verifica: Per verificare la manutenibilità del sistema, utilizziamo un ambiente di sviluppo locale con strumenti di versionamento come GitLab per monitorare i cambiamenti nel codice e calcolarne i tempi di implementazione. Inoltre, eseguiamo test automatici tramite il framework JEST, assicurandoci che ogni modifica venga testata prima di essere distribuita.

9 Presentazione del prodotto

Panoramica delle principali pagine dell'applicazione, illustrandone le funzionalità attraverso descrizioni dettagliate e schermate rappresentative.

9.1 Accesso all'applicazione

Di seguito vengono presentate alcune schermate che rappresentano le pagine introduttive dell'applicazione web, incluse quelle dedicate alla presentazione e all'autenticazione. Queste sezioni, visualizzate prima di accedere alle funzionalità principali, guidano l'utente nella fase iniziale dell'interazione con il sistema, introducendolo al contesto dell'applicazione e permettendo un accesso sicuro tramite le credenziali di autenticazione.

9.1.1 Presentazione

Viene visualizzata una pagina introduttiva che offre una panoramica generale dell'applicazione, fornendo una descrizione sintetica delle sue principali caratteristiche e finalità. Al centro dell'attenzione è presente un pulsante che consente agli utenti di procedere alla fase di autenticazione, avviando così l'accesso alle funzionalità disponibili.

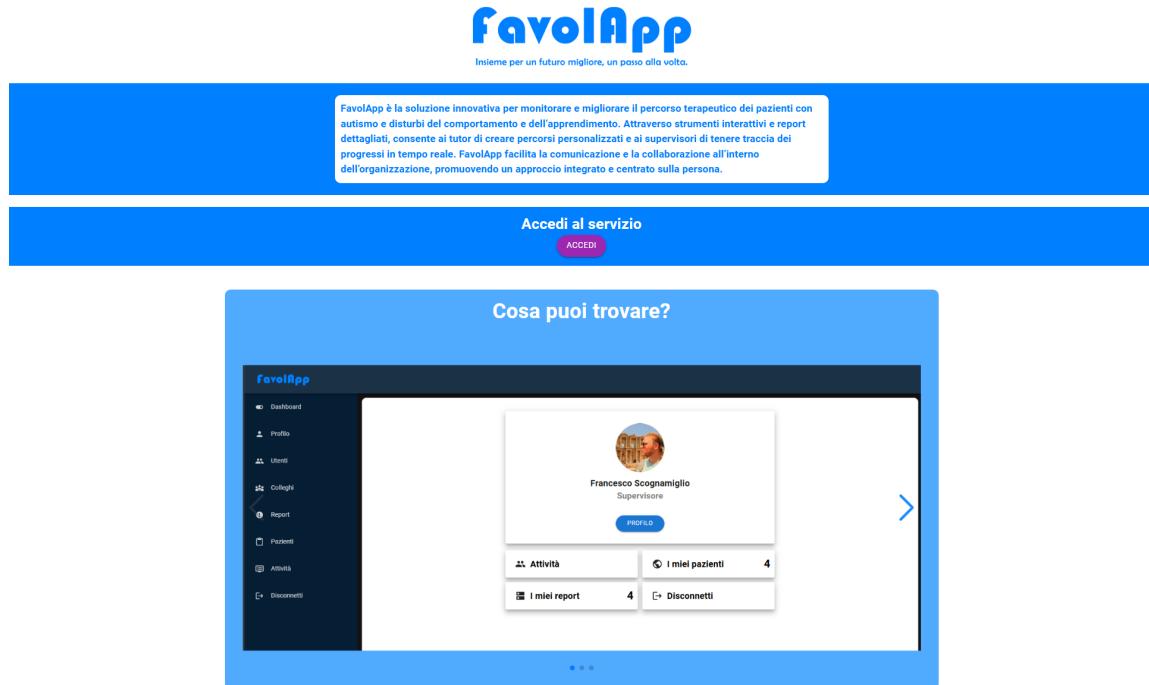


Figura 9.1: Pagina di presentazione.

9.1.2 Autenticazione

Cliccando sul pulsante "Accedi", l'utente viene reindirizzato alla pagina di autenticazione. In questa sezione è possibile inserire le credenziali di accesso, ovvero l'indirizzo email e la password, necessarie per effettuare il login e accedere alle funzionalità offerte dall'applicazione.

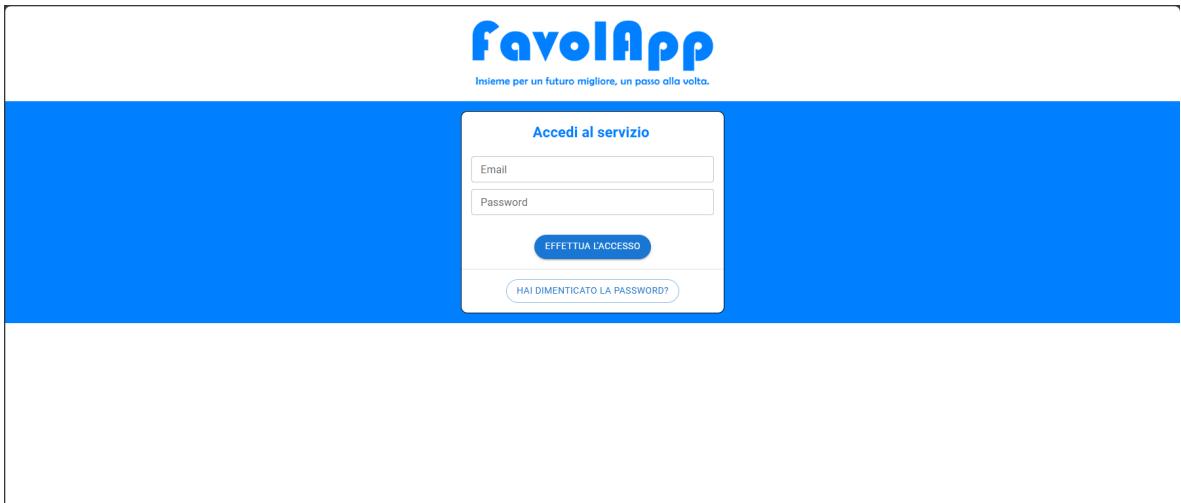


Figura 9.2: Pagina di autenticazione.

Successivamente, l'utente viene indirizzato a una pagina dedicata al secondo livello di autenticazione, dove viene mostrato un form per l'inserimento del codice OTP (One-Time Password). Questo codice, generato da un'app di autenticazione come DUO o Google Authenticator, garantisce un ulteriore livello di sicurezza per l'accesso all'applicazione.

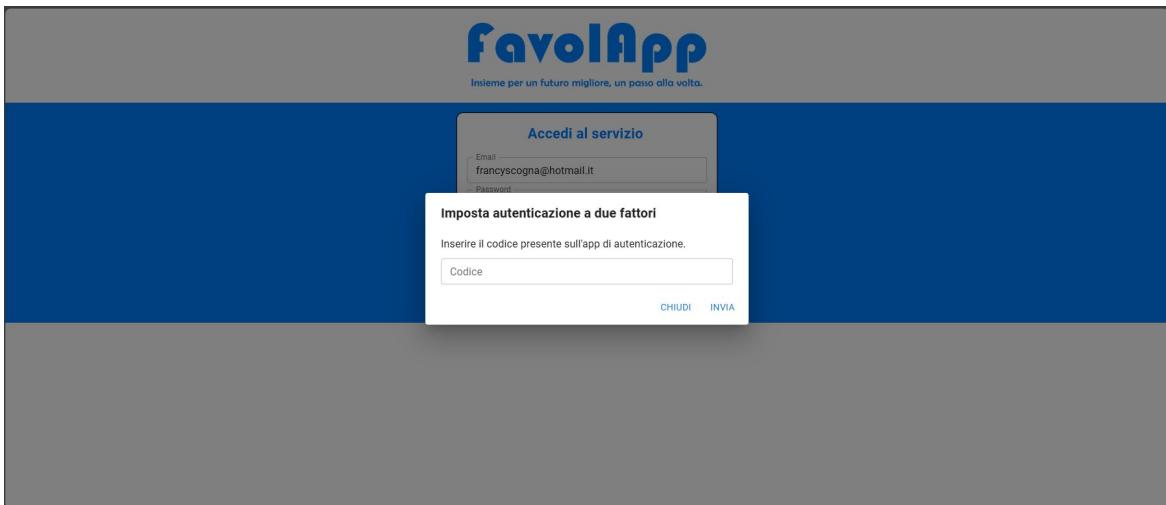


Figura 9.3: Form per l'inserimento dell'OTP.

9.2 Dashboard

Dopo aver completato con successo il processo di autenticazione, l'utente viene reindirizzato al lato protetto dell'applicazione, dove può accedere a tutte le funzionalità disponibili. La prima pagina visualizzata è la dashboard, una sezione centrale che offre una panoramica delle opzioni principali. Qui, l'utente trova collegamenti rapidi alle varie pagine dell'applicazione, consentendo un accesso immediato e intuitivo alle funzioni principali.

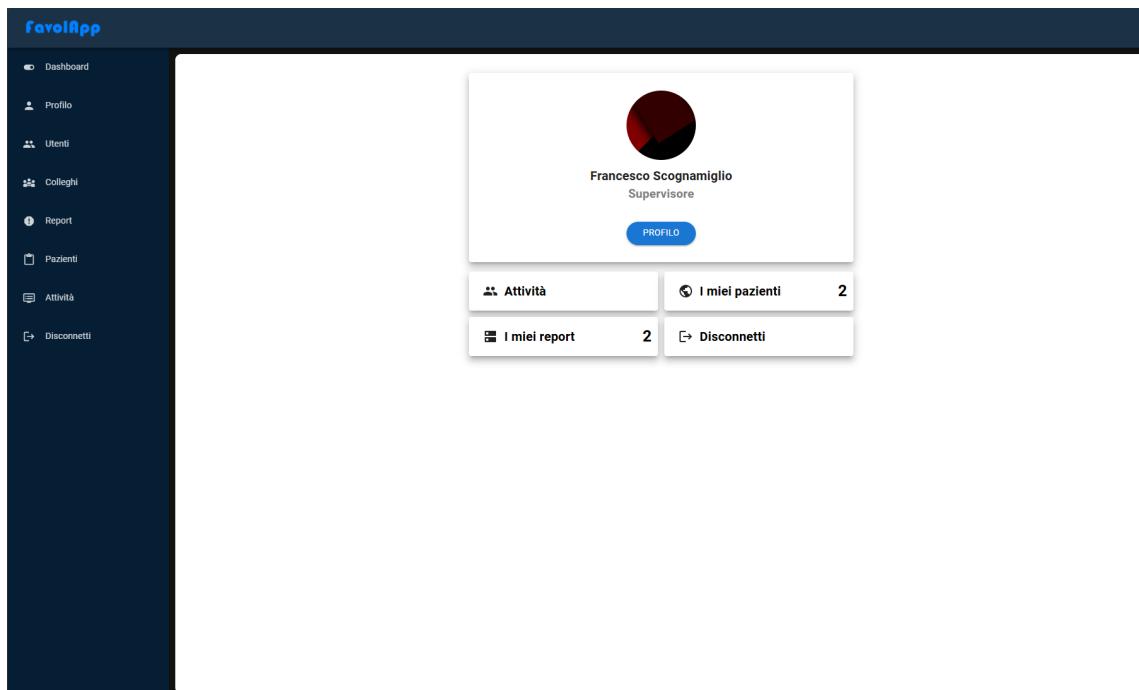


Figura 9.4: Dashboard.

9.3 Profilo personale

La sezione profilo, accessibile sia dalla dashboard che tramite la barra di navigazione, offre una panoramica dei dati personali dell'utente, inclusi i dettagli principali e l'avatar. Questi elementi possono essere modificati esclusivamente dagli utenti con ruolo di supervisore, garantendo un adeguato livello di controllo e sicurezza. Inoltre, cliccando sull'avatar, viene fornita la possibilità di aggiornare l'immagine del profilo, rendendo l'interfaccia personalizzabile e più interattiva.

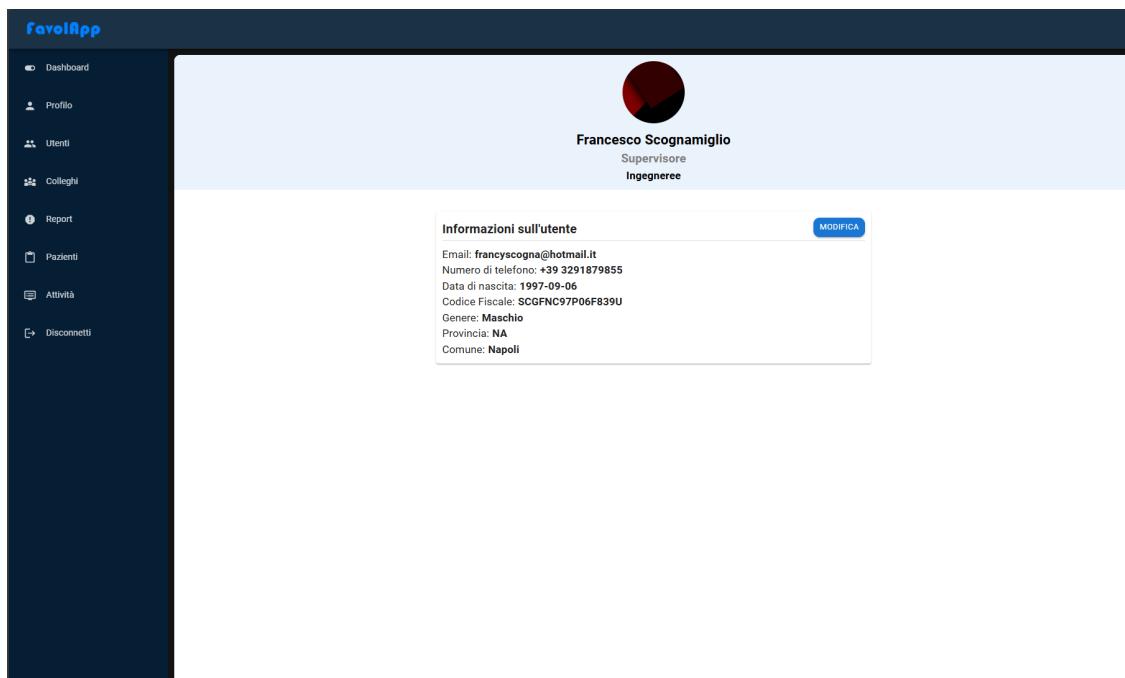


Figura 9.5: Profilo personale.

9.4 Utenti

La sezione utenti offre una panoramica completa di tutti gli utenti registrati sulla piattaforma, presentando una lista dettagliata e facilmente consultabile. È disponibile una funzione di ricerca avanzata che consente di trovare rapidamente un utente specifico utilizzando parametri come nome, cognome o codice fiscale. Per ogni utente elencato, sono disponibili opzioni per visualizzare, modificare o eliminare le relative informazioni, garantendo un controllo completo e immediato. Inoltre, la sezione permette l'aggiunta di nuovi utenti attraverso un form dedicato, che viene visualizzato cliccando sul pulsante "Aggiungi utente", offrendo un'interfaccia intuitiva per la gestione dei dati utente.

The screenshot shows the FavolAPP user management interface. On the left is a dark sidebar with navigation links: Dashboard, Profilo, Utenti, Colleghi, Report, Pazienti, Attività, and Disconnectti. The main area has a header 'Utenti' with a blue 'AGGIUNGI UTENTE' button. Below it is a search bar with placeholder 'Cerca un utente...'. A table lists users with columns: Ruolo, Nome, Cognome, Titolo, Codice Fiscale, and Azioni. The table contains 12 rows of data. At the bottom right of the table is a page number '1-12 of 12' with navigation arrows.

Figura 9.6: Pagina degli utenti.

The screenshot shows the FavolAPP user addition form. It has a header 'Aggiungi utente all'organizzazione' with a close button 'X'. The form is divided into two sections: 'Informazioni sull'utente' and 'Riguardo l'organizzazione'. The 'Informazioni sull'utente' section contains fields for Nome, Cognome, Email, Numero di telefono (with a dropdown for country code and a placeholder '+39 Numero di telefono'), Data di nascita (with a date picker), Genere (with a dropdown), Codice Fiscale, Provincia (with a dropdown), and Comune (with a dropdown). The 'Riguardo l'organizzazione' section contains fields for Titolo di professione and Ruolo (with a dropdown). At the bottom are 'CHIUDI' and 'AVANTI' buttons. To the right of the form is a sidebar showing a list of users with columns: Ruolo, Nome, Cognome, Titolo, Codice Fiscale, and Azioni. The sidebar contains 12 rows of data. At the bottom right of the sidebar is a page number '1-12 of 12' with navigation arrows.

Figura 9.7: Aggiunta utente.

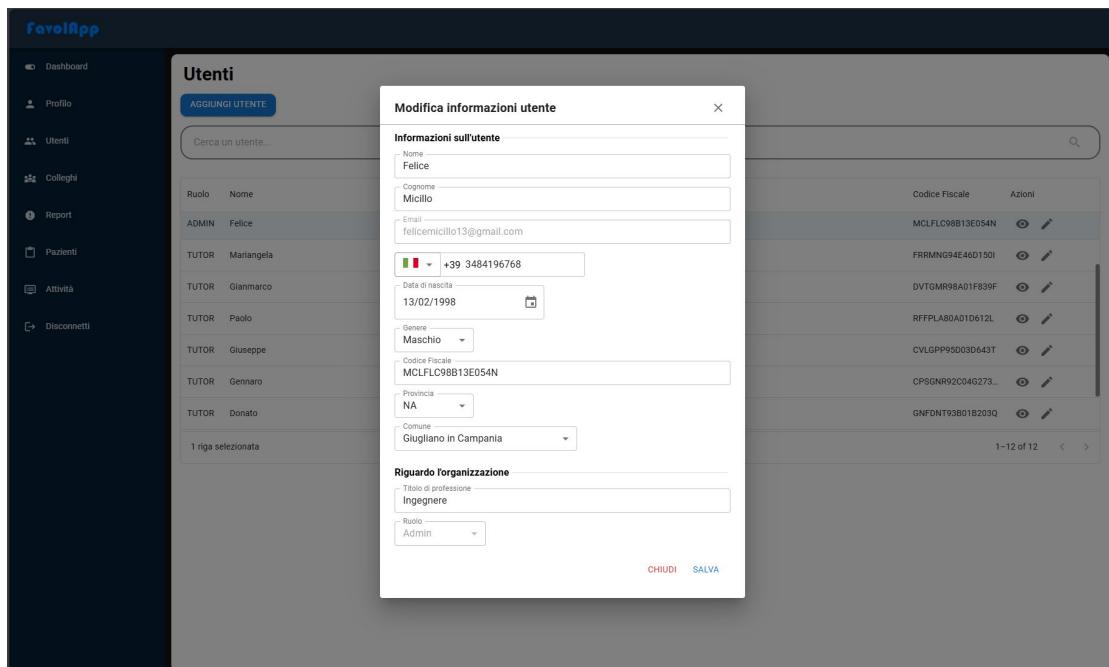


Figura 9.8: Modifica utente.

9.5 Pazienti

La sezione pazienti fornisce una lista completa di tutti i pazienti registrati nella piattaforma, offrendo un'interfaccia chiara e organizzata. Per ogni paziente è possibile effettuare l'operazione di visualizzazione, consentendo agli utenti di accedere rapidamente alle informazioni dettagliate. Inoltre, è disponibile una funzionalità di ricerca avanzata che permette di individuare facilmente un paziente specifico utilizzando filtri come nome, cognome o codice fiscale. La sezione include anche la possibilità di aggiungere nuovi pazienti tramite un apposito form, garantendo una gestione semplice ed efficiente dei dati relativi ai pazienti.

Figura 9.9: Pagina dei pazienti.

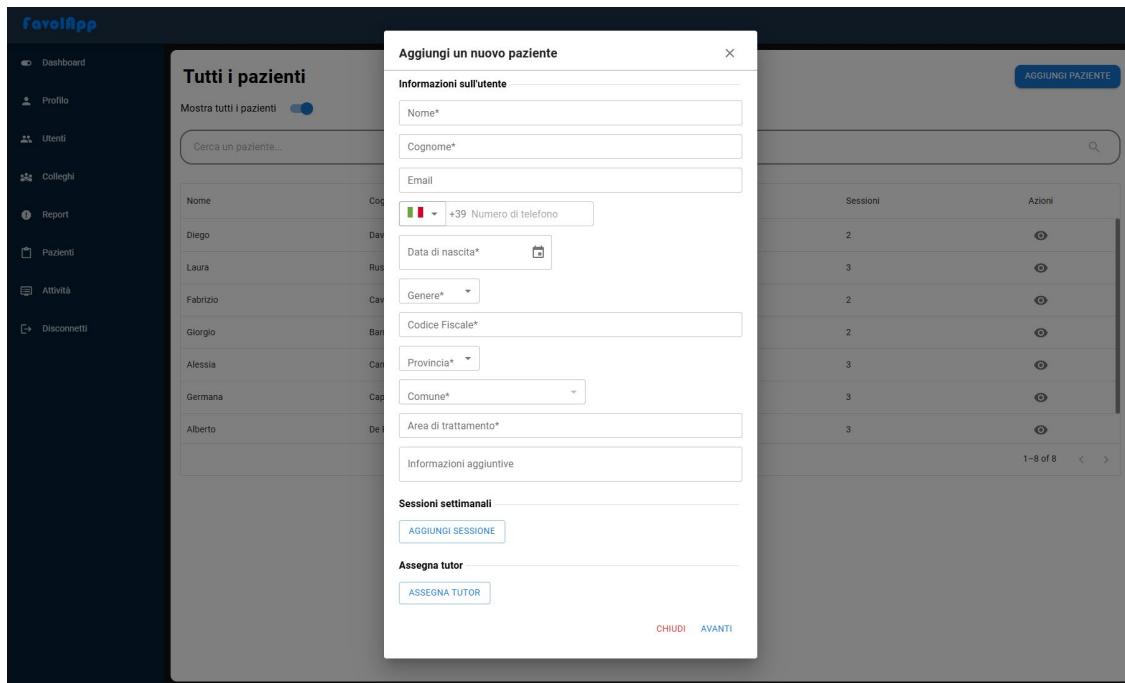


Figura 9.10: Aggiunta paziente.

9.5.1 Scheda del paziente

Cliccando sul pulsante di visualizzazione accanto a un paziente nella lista, viene mostrata una scheda dettagliata del paziente. Questa scheda consente di consultare tutte le informazioni personali del paziente, inclusi i dettagli delle sessioni settimanali di terapia e i tutor assegnati. Inoltre, è presente un'opzione per modificare i dati del paziente tramite un apposito form, che permette di aggiornare le informazioni in modo rapido e intuitivo.

Informazioni sull'utente
Nome: Alberto
Cognome: De Rosa
Data di Nascita: 2019-12-11
Numero di Telefono: Non indicato.
Genere: M
Email: Non indicata.
Codice Fiscale: DRLSLRT19T11F111A
Provincia: NA
Comune: Melito di Napoli
Informazioni: Non indicate.
Trattamento: Discalculia
Numero di Sessioni: 3
Numero di Tutor: 2

Sessioni settimanali
Giorno: Lunedì Ora di inizio: 16:00 Ora di fine: 17:30
Giorno: Mercoledì Ora di inizio: 16:00 Ora di fine: 17:30
Giorno: Venerdì Ora di inizio: 16:00 Ora di fine: 17:30

Tutor assegnati
Felice Micillo Ingegner MCLFLC98B13E054N
Maria Esposito Psicoterapeuta SPSMRA93B41F205G

Figura 9.11: Scheda del paziente.

Figura 9.12: Modifica del paziente.

9.6 Report

La sezione report offre una lista completa di tutti i report presenti sulla piattaforma, con la possibilità di visualizzare e modificare ciascun report. Tuttavia, la modifica è riservata esclusivamente all'autore del report, garantendo un controllo adeguato sulle modifiche apportate. La sezione include anche la funzionalità per creare un nuovo report, consentendo di associarlo a un paziente specifico tramite un'interfaccia intuitiva.

Report						
Mostra i report dei colleghi		TUTTI I REPORT		CREA REPORT		
Data	Nome Tutor	Ruolo	Paziente	Titolo Report	Stato	Azioni
10/12/2024	Francesco Scognamiglio	ADMIN	Donato Panariello	Valutazione Clinica e Piano di Intervento per ADHD	Visto	
1 riga selezionata						

Figura 9.13: Pagina dei report.

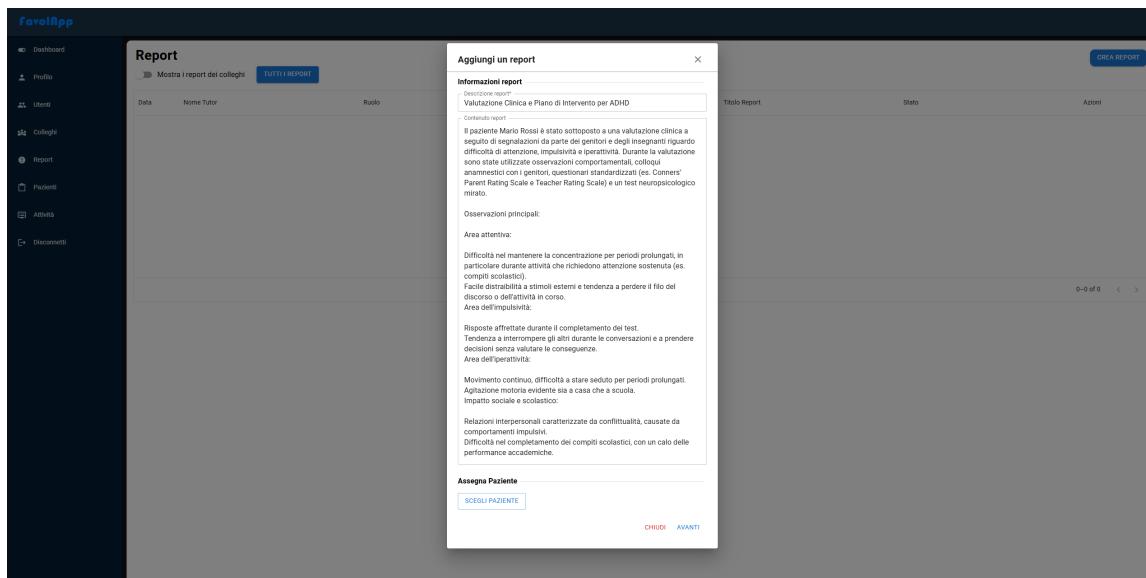


Figura 9.14: Crea un report.

Quando un report viene visualizzato, sono mostrate informazioni dettagliate, come il nome dell'autore, la data di creazione e il contenuto del report, fornendo un quadro chiaro e completo delle informazioni archiviate.

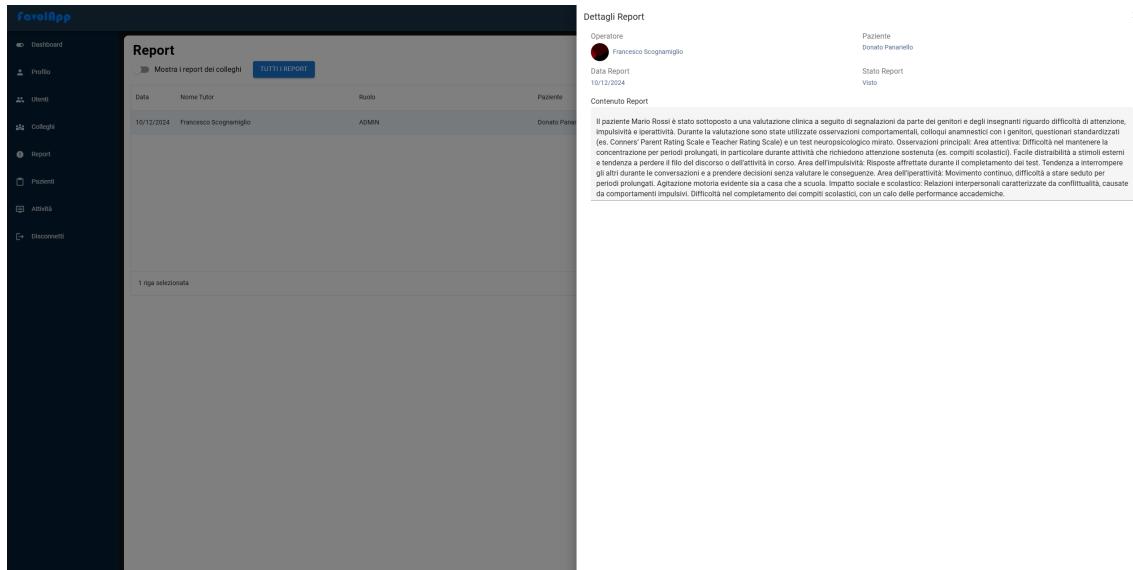


Figura 9.15: Visualizza report.

Figura 9.16: Modifica report.

9.7 Colleghi

La sezione colleghi presenta una lista di utenti che condividono la gestione di pazienti comuni, facilitando la collaborazione tra i diversi attori coinvolti. Questa sezione permette di identificare rapidamente i colleghi associati a uno stesso paziente e di accedere alle loro informazioni. Per ogni collega elencato, è possibile visualizzarne il profilo personale.

Ruolo	Nome	Cognome	Titolo	Codice Fiscale	Azioni
TUTOR	Francesco	Scognamiglio	ING	SCGFNC97P06F839U	
TUTOR	Gianmarco	De Vita	Logopedista	DVTGMR98A01F839F	
ADMIN	Francesco	Scognamiglio	Ingegnere	SCGFNC97P06F839U	

Figura 9.17: Lista colleghi.

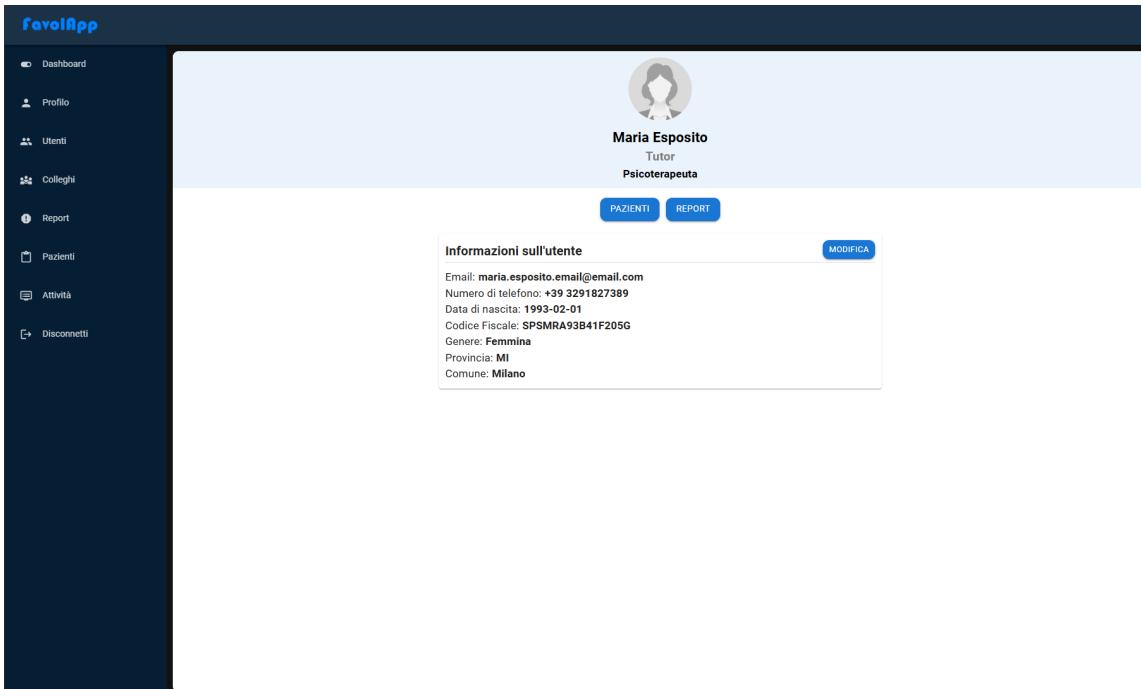


Figura 9.18: Profilo di un altro utente.

9.8 Attività

La sezione attività offre al supervisore una panoramica delle principali operazioni eseguite all'interno del sistema, consentendo un monitoraggio efficace delle azioni compiute. Per ogni attività registrata, è possibile visualizzare i dettagli specifici, fornendo una comprensione approfondita dell'evento. Inoltre, la sezione include una funzionalità di filtro avanzata che permette di ordinare le attività in base all'autore o alla data, facilitando la ricerca e l'analisi di eventi specifici.

Data	Event ID	Autore
2024-12-1...	38466328764943922349659178140296147149494213527078100964	Francesco Scognamiglio
2024-12-1...	38466328319251454914972079593599313099584583858707456	Francesco Scognamiglio
2024-12-1...	384663234107447149712613473531632254879828325077184	Francesco Scognamiglio
2024-12-1...	384663213277337782928012102548188239918940967985054468	Francesco Scognamiglio
2024-12-1...	38466167516441515234232956270091026756663868274189696	Francesco Scognamiglio
2024-12-1...	38466164013937768695103705391916759232916244083810544	Francesco Scognamiglio
2024-12-1...	38466163816025572177484619179940597450058278661108512	Francesco Scognamiglio

1 riga selezionata

Figura 9.19: Visualizzazione di un'attività.