

## Progetto di programmazione ad oggetti ed Ingegneria del software.

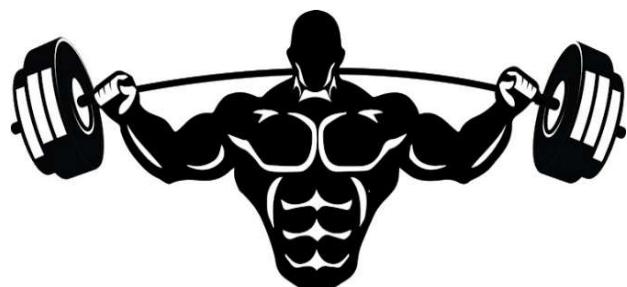
**Titolo: Gym for You.**

**Studenti: Infortuna Emanuele 479483.**

**Giuseppe Primerano 476933.**

**Docente: Salvatore Distefano.**

**SIMBOLO:**



## Sommario

1. Prefazione.....	11
2. Panoramica del software.....	12
2.1 Linguaggi per la descrizione UML.....	12
2.2 Tecnologie implementative.....	14
2.3 Team di sviluppo.....	15
2.4 Requisiti Team.....	15
2.5 Tempistiche di progetto.....	15
3. Requisiti informali.....	16
4. Modello di sviluppo.....	19
4.1 RUP (Rapid Unified Process).....	19

## PRIMO PROTOTIPO

5. Prima fase INCEPTION.....	22
5.1.1 Analisi dei requisiti.....	22
5.1.2 Descrizione stakeholder.....	22
5.1.3 Specifica dei requisiti.....	22
5.1.4 Requisiti funzionali.....	23
5.1.5 Requisiti non funzionali.....	23
5.2 Indagine sul sistema.....	24
5.2.1 Analisi del dominio.....	24
5.2.2 Studio fattibilità.....	25
5.3 Analisi dei rischi.....	30
5.3.1 Rischi di progetto.....	30
5.3.2 Rischi di prodotto.....	31
5.3.3 Tabella valutazione rischi.....	31
6. Seconda fase ELABORATION.....	32
6.1.1 Prima attività requisiti.....	32
6.1.2 Requisiti funzionali.....	33
6.1.3 Requisiti non funzionali.....	34
6.1.4 Documento SRS.....	35

6.1.5 Diagramma dei casi d'uso.....	37
6.1.6 Descrizione attori.....	38
6.1.7 Descrizione casi d'uso.....	39
6.1.8 Diagramma delle attività.....	42
7. Seconda attività: ANALISI.....	44
7.1 Diagramma dei package.....	45
7.2 Approccio BCE.....	46
7.3 Diagramma delle classi.....	47
8. Diagramma di sequenza.....	50
9. Terza attività: PROGETTAZIONE.....	55
9.1 Architettura di sistema.....	55
9.2 Server.....	56
9.3 Client.....	56
9.4 Protocollo di comunicazione.....	57
9.5 Configurazione.....	57
9.6 Database (db).....	58
9.7 Diagrammi degli stati.....	60
10. Descrizione delle GUI.....	65
10.1 Descrizione algoritmo.....	73
11. Quarta attività: IMPLEMENTAZIONE.....	76
11.1 Diagramma dei componenti.....	77
12. Quinta attività: TESTING.....	78
13. Verifica.....	83
14. Ultima attività: DEPLOYMENT.....	83
Conclusioni primo prototipo.....	83
15. SECONDO PROTOTIPO - Prima fase INCEPTION.....	84
15.1 Analisi dei requisiti.....	84
15.2 Descrizione stakeholder.....	85

15.3 Specifica dei requisiti.....	85
15.4 Requisiti funzionali.....	85
15.5 Indagine sul sistema.....	85
15.6 Analisi del dominio.....	85
15.7 Studio fattibilità.....	86
<b>16. Analisi dei rischi.....</b>	<b>86</b>
16.1 Rischi di progetto.....	86
16.2 Rischi di prodotto.....	86
16.3 Tabella valutazione rischi.....	87
<b>17. Seconda fase ELABORATION.....</b>	<b>87</b>
17.1 Prima attività requisiti.....	87
17.1.2 Requisiti funzionali.....	87
17.1.3 Requisiti non funzionali.....	88
17.1.4 Documento SRS.....	88
17.1.5 Diagramma dei casi d'uso.....	90
17.1.6 Descrizione attori.....	90
17.1.7 Descrizione casi d'uso.....	91
17.1.8 Diagramma delle attività.....	93
<b>18. Seconda attività: ANALISI.....</b>	<b>94</b>
18.1 Diagramma dei package.....	95
18.2 Individuazione delle classi di analisi.....	95
<b>19. Diagrammi di sequenza.....</b>	<b>97</b>
<b>20. Terza attività: PROGETTAZIONE.....</b>	<b>101</b>
20.1 Architettura di sistema.....	101
20.2 Database (db).....	101
20.3 Diagrammi degli stati.....	102
<b>21. Descrizione delle GUI.....</b>	<b>104</b>
<b>22. Quarta attività: IMPLEMENTAZIONE.....</b>	<b>111</b>

22.1 Diagramma dei componenti client-server (aggiornato)...	111
22.2 Diagramma dei componenti: Gym for you client AGGIORNATO....	112
23. Quinta attività: testing.....	112
24. Verifica.....	114
25. Ultima attività: DEPLOYMENT.....	115
26. Diagramma della fasi-worflow RUP.....	115
27. TERZO PROTOTIPO - Prima fase INCEPTION.....	116
27.1.1 Analisi dei requisiti.....	117
27.1.2 Descrizione stakeholder.....	117
27.1.3 Specifica dei requisiti.....	117
27.1.4 Requisiti funzionali.....	117
27.1.5 Requisiti non funzionali.....	118
27.2 Indagine sul sistema.....	118
27.2.1 Analisi del dominio.....	118
27.2.2 Studio fattibilità.....	118
27.3 Analisi dei rischi.....	118
27.3.1 Rischi di progetto.....	118
27.3.2 Rischi di prodotto.....	119
27.3.3 Tabella valutazione rischi.....	119
28. Seconda fase ELABORATION.....	120
28.1.1 Prima attività requisiti.....	120
28.1.2 Requisiti funzionali.....	120
28.1.3 Requisiti non funzionali.....	121
28.1.4 Documento SRS.....	121
28.1.5 Diagramma dei casi d'uso.....	123
28.1.6 Descrizione attori.....	123
28.1.7 Descrizione casi d'uso.....	124
28.1.8 Diagramma delle attività.....	126

29. Seconda attività: ANALISI.....	128
29.1 Diagramma dei package.....	128
29.2 Diagrammi delle classi.....	128
30. Diagrammi di sequenza.....	131
31. Terza attività: PROGETTAZIONE.....	133
31.1 Architettura di sistema.....	133
31.2 Database (db).....	133
31.3 Diagrammi degli stati.....	134
32. Descrizione delle GUI.....	137
32.1 Descrizione algoritmo.....	145
33. Quarta attività: IMPLEMENTAZIONE.....	151
33.1 Diagramma dei componenti: client server (aggiornato) ..	151
33.2 Diagramma dei componenti: Gym for you client.....	152
33.3 Caratteristiche OOP.....	153
34. Quinta attività: testing.....	164
35. Verifica.....	166
36. Ultima attività: DEPLOYMENT.....	166
37. Diagramma delle fasi-workflow RUP.....	167
Bibliografia	

## Glossario.

### Termini e acronimi usati nella relazione.

Di seguito riportiamo tutti i termini e gli acronimi usati divisi per capitolo.

#### Capitolo 1.

**Fitness:** Nella terminologia medico-sportiva si indica con il termine inglese fitness, "forma, buona salute", il complesso di condizioni fisiche e psicologiche che consentono a un individuo di manifestare al meglio le sue attitudini e possibilità.

**Wellness:** neologismo coniato per indicare uno stato soggettivo di benessere fisico e psichico.

## Capitolo 2.

**Architettura:** L'architettura software è l'organizzazione di base di un sistema, espressa dalle sue componenti, dalle relazioni tra di loro e con l'ambiente, e i principi che ne guidano il progetto e l'evoluzione.

**Classe:** la classe definisce come sono fatti gli oggetti da essa derivati, quindi l'insieme dei possibili metodi, attributi.

**Cliente:** Persona o gruppo di persone che ha richiesto la creazione del prodotto.

**IDE:** Ambienti di sviluppo integrati. Forniscono strumenti che supportano il processo di sviluppo del software, inclusi editor per scrivere e modificare programmi e debugger per localizzare errori.

**Ingegneria del software:** Disciplina che studia le modalità e le metodologie con le quali avviene il processo di produzione del software, dalla raccolta dei requisiti fino alla realizzazione del prodotto e al suo rilascio in produzione. Tale disciplina ha pertanto molteplici obiettivi: individuare quali sono nel processo di produzione del software le diverse fasi attraverso le quali tale processo si struttura, il cosiddetto ciclo di vita del software; definire e documentare opportune metodologie di sviluppo che possano coprire le fasi individuate e identificare le tecnologie più appropriate per realizzare i prodotti utilizzando le suddette metodologie. Tali obiettivi vengono perseguiti considerando il processo di sviluppo del software da differenti angolazioni e utilizzando approcci complementari: non solo un approccio scientifico e tecnologico, ma anche un approccio di tipo economico e organizzativo, che sia in grado di evidenziare costi e benefici nel processo produttivo e che giustifica appieno il termine ingegneria del software assegnato a tale disciplina.

**Java:** Linguaggio di programmazione orientato agli oggetti.

**Object Oriented:** La programmazione orientata agli oggetti (OOP) è un modello di programmazione per computer che organizza la progettazione del software attorno a dati o oggetti, piuttosto che a funzioni e logica. OOP si concentra sugli oggetti che gli sviluppatori desiderano manipolare piuttosto che sulla logica richiesta per manipolarli.

**Oggetti:** nella programmazione object-oriented si definisce oggetto un qualcosa che possiede uno stato e degli attributi. Nei computer ovviamente un oggetto non è qualcosa di reale, ma, al più, una rappresentazione. Viene anche definito istanza di una classe.

**Panoramica software:** con questa parola s'intende una visione non troppo tecnicistica di quello che sarà il prodotto software. Vengono introdotte le caratteristiche del software senza entrare troppo nel dettaglio.

**Prodotto:** è il software che vogliamo realizzare Normalmente quando si cerca di pensare al software (inteso come il prodotto di un certo processo), viene da paragonarlo alla produzione di un qualche oggetto. **Sinonimi:** sistema.

**Software:** software termine inglese che indica l'insieme dei programmi necessari a un elaboratore elettronico per il suo funzionamento e per la soluzione di problemi.

Il software spesso è dipendente dalle caratteristiche dell'hardware del calcolatore stesso, ossia dall'insieme dei suoi componenti fisici (apparecchiature, circuiti, dispositivi ecc.). **Sinonimi:** prodotto, sistema.

**Standard:** Modello, tipo, norma, riferimento cui si devono uniformare, o a cui sono conformi, i prodotti e i procedimenti.

**UML:** Unified Modeling Language. Si rimanda al paragrafo 2.1 per ulteriori chiarimenti.

**XAMPP:** XAMPP è una piattaforma software multipiattaforma e libera costituita da Apache HTTP Server, il database MariaDB e tutti gli strumenti necessari per utilizzare i linguaggi di programmazione PHP e Perl. Il nome è un acronimo dei software sopra citati (la X sta per x-platform, l'abbreviazione di cross-platform in lingua inglese ovvero multipiattaforma).

## Capitolo 3

**Requisiti informali:** Le specifiche informali fanno uso di linguaggio naturale per descrivere i requisiti. La sintassi e la semantica non sono formalmente definite.

**Stakeholder:** Tutti i soggetti, individui od organizzazioni, attivamente coinvolti in un'iniziativa economica (progetto, azienda), il cui interesse è negativamente o positivamente influenzato dal risultato dell'esecuzione, o dall'andamento, dell'iniziativa e la cui azione o reazione a sua volta influenza le fasi o il completamento di un progetto o il destino di un'organizzazione. Nell'ambito di un progetto, sono i soggetti relativi al cliente, al fornitore, i membri del team di progetto, i fruitori dei risultati in uscita dal progetto, i gruppi di interesse locali relativamente all'ambiente dove il progetto si sviluppa e l'azienda opera. Tra gli stakeholder vi sono i soggetti senza i quali l'impresa non sopravvive. L'identificazione degli s. si ottiene mediante un elenco casuale e libero dei soggetti coinvolti nel progetto (tecniche di brainstorming) oppure mediante liste di controllo descrittive dell'ambiente di progetto o di progetti precedenti (check list) o infine mediante simulazioni dell'ambiente di progetto per rintracciare gli stakeholder interni ed esterni (rappresentazione). **Sinonimi:** cliente, utente.

## Capitolo 5.

**Autenticazione:** L'autenticazione è il processo attraverso il quale viene verificata l'identità di un utente che vuole accedere ad un computer o ad una rete. È il sistema che verifica, effettivamente, che un individuo è chi sostiene di essere. L'autenticazione è diversa dall'identificazione (la determinazione che un individuo sia conosciuto o meno dal sistema) e dall'autorizzazione (il conferimento ad un utente del diritto ad accedere a specifiche risorse del sistema, sulla base della sua identità).

**Requisiti funzionali:** descrivono i servizi, o funzioni, offerti dal sistema normalmente attivati da user-inputs.

**Requisiti non funzionali:** descrivono vincoli sui servizi offerti dal sistema, e sullo stesso processo di sviluppo.

**Skill:** conoscenze e abilità che qualcuno possiede. Possono essere acquisite attraverso il processo di istruzione, l'addestramento, l'esperienza lavorativa o essere semplicemente capacità innate.

**Rischi di progetto:** I rischi di progetto sono legati al compimento del progetto ad esempio un rischio potrebbe far slittare la consegna del software.

**Rischi di prodotto:** Rischi che generano un problema funzionale al prodotto software.

**Tabella dei rischi:** Tale tabella si basa, oltre che sul catalogo dei rischi, anche su una strategia suggerita dalla U.S. Air Force Questa strategia vuole che il capo progetto individui i fattori di rischio che incidono sui componenti di rischio del software: prestazioni, costi, supporto e tempi.

## Capitolo 6.

**Casi d'uso:** è una tecnica usata nei processi di ingegneria del software per effettuare in maniera esaustiva e non ambigua, la raccolta dei requisiti al fine di produrre software di qualità.

**GUI: acronimo.** Graphical User Interface in informatica è un tipo di interfaccia utente che consente l'interazione uomo-macchina in modo visuale utilizzando rappresentazioni grafiche.

**Documento SRS:** Documento in forma tabellare utilizzato per una descrizione più accurata di quelli che sono i casi d'uso di un sistema.

**Indispensabile:** Riferito ai casi d'uso senza la quale il funzionamento stesso del sistema potrebbe risentire della loro mancanza e non funzionare correttamente.

**Sarebbe meglio avere:** Riferito ai casi d'uso, senza essi il sistema funziona ugualmente. Inseriti per aumentare le funzionalità del nostro sistema.

**Facoltativo:** L'utente può decidere se farlo o no.

**Diagramma dei casi d'uso:** Il diagramma dei casi d'uso assegna i casi d'uso agli attori ed evidenzia le relazioni tra i casi d'uso. Nel diagramma dei casi d'uso un attore può essere rappresentato da un omino stilizzato, con sotto il nome dell'attore (normalmente rappresenta un attore esterno). Un caso d'uso è rappresentato come un ellissi con all'interno il nome del caso d'uso. Gli attori sono associati ai casi d'uso per mezzo di frecce che puntano dall'attore al caso d'uso. Oltre a questo è possibile la generalizzazione degli attori e dei casi d'uso.

**Diagramma delle attività:** Si prendono le descrizioni dei flussi: sia quello principale che quelli alternativi sono fusi nello stesso diagramma per poter far vedere l'insieme delle azioni compiute dal caso d'uso preso in considerazione. Per preparare un diagramma delle attività si disegna un cerchio pieno che indica il punto di inizio dell'attività; quando un caso d'uso viene attivato, si ci sposta sulla prima attività (rappresentata da un rettangolo dai bordi arrotondati), quando l'attività ha compiuto il suo compito, si ci sposta alla seconda e così via fino allo stato finale (rappresentato da un cerchio pieno inscritto in un secondo cerchio vuoto) in cui il caso d'uso termina. Nel momento in cui bisogna fare una scelta (e quindi passare ad un flusso alternativo) il collegamento avviene attraverso un rombo che permette di spezzare la linea di

congiunzione tra le attività a seconda delle condizioni specificate su ogni ramo (sono racchiuse tra parentesi quadre); quando i flussi alternativi si riuniscono lo fanno attraverso un secondo rombo.

## Capitolo 7.

**Pug-in:** Il plu-gin in campo informatico è un programma non autonomo che interagisce con un altro programma per ampliarne o estenderne le funzionalità originarie: possono essere utilizzati non solo su software, ma anche su qualunque cosa che possa essere visitata da chiunque, quindi pubblica.

**Diagramma delle classi:** Questo diagramma viene fornito da UML, grazie ad esso è possibile visualizzare graficamente le classi e le loro associazioni. Una classe viene rappresentata tramite un rettangolo contenente un nome; questo rettangolo può essere diviso in tre compatti. Il primo riporta il nome, il secondo gli attributi ed il terzo le operazioni. La linea che collega due classi indica un'associazione, le associazioni possono avere molteplicità (segnata agli estremi), nome (segnato al centro), nomi di ruoli (segnati agli estremi). Se la linea è in realtà una freccia, significa che la navigabilità è solo in quella direzione.

**Package:** Essi mettono insieme classi che si occupano della stessa semantica, vengono solitamente trovati osservando i casi d'uso e le relazioni tra le classi. Le relazioni che potrebbero indicare i package sono ereditarietà, aggregazione, composizione o dipendenza.

## Capitolo 8.

**Diagramma di sequenza:** I diagrammi di sequenza, comunemente usati dagli sviluppatori, modellano le interazioni tra gli oggetti in un unico caso d'uso. Essi illustrano come le diverse parti di un sistema interagiscono tra loro per svolgere una funzione, e l'ordine in cui le interazioni avvengono quando un particolare caso d'uso viene eseguito. In parole più semplici, un diagramma di sequenza mostra diverse parti del lavoro di un sistema in una 'sequenza' per ottenere qualcosa.

**Query:** In informatica, interrogazione di un database per estrarre o aggiornare i dati che soddisfano un certo criterio di ricerca.

**Thread:** Un thread è un percorso di esecuzione indipendente all'interno di un programma.

## Capitolo 9.

**DBMS:** DataBase Management System è un sistema software progettato per consentire la creazione, la manipolazione e l'interrogazione efficiente di database, per questo detto anche "gestore o motore del database", è ospitato su architettura hardware dedicata oppure su semplice computer. La teoria dei database e dei DBMS rappresenta da sempre uno dei filoni più solidi e importanti dell'informatica.

**Diagrammi degli stati:** Un diagramma di stato viene utilizzato per rappresentare la condizione del sistema o parte del sistema in istanze di tempo finite. È un diagramma comportamentale e rappresenta il comportamento utilizzando transizioni di stato finite. I diagrammi della macchina a stati UML mostrano i diversi stati di un'entità. I

diagrammi a stati possono anche mostrare come un'entità risponde a vari eventi passando da uno stato all'altro.

## Capitolo 10.

**GUI:** la GUI ( Graphical User Interface ) è l'interfaccia grafica per l'utente di un software. È un termine informatico poco conosciuto dagli utenti. L'acronimo GUI identifica una interfaccia grafica in grado di consentire una migliore interazione tra l'utente e il software.

**Frame:** interfaccia o finestra che appare ai nostri attori.

## Capitolo 11.

**Diagramma dei componenti:** I diagrammi dei componenti UML vengono utilizzati nella modellazione degli aspetti fisici dei sistemi orientati agli oggetti che vengono utilizzati per visualizzare, specificare e documentare i sistemi basati sui componenti e anche per costruire sistemi eseguibili tramite ingegneria diretta e inversa. I diagrammi dei componenti sono essenzialmente diagrammi di classe che si concentrano sui componenti di un sistema che spesso vengono utilizzati per modellare la vista dell'implementazione statica di un sistema.

# 1. PREFAZIONE

Oggi il mercato del fitness e del wellness non soltanto sono di moda, ma finalmente molti stanno comprendendo l'importanza dell'attività fisica e dello sport.

Anche gli altri numeri indicano una tendenza in crescita del mercato del fitness.

Tra i più significativi:

Gli introiti complessivi del mercato del fitness ammontano nel 2017 a 26,6 miliardi di Euro, con una crescita del 3,8%, che fanno dell'Europa il più importante mercato mondiale del settore.

Su 794 milioni di persone che vivono nell'Unione Europea e in Norvegia, Russia, Svizzera, Turchia e Ucraina, il 7,5% sono iscritte alle palestre, con un'espansione del 9% sulle persone oltre i 15 anni.

Le più importanti aziende del mercato del fitness nel settore hanno aumentato i loro introiti del 2,9% (oltre 3 miliardi di Euro).

In questa situazione di emergenza per l'epidemia di COVID-19 in Italia, molti adolescenti si trovano a trascorrere la giornata in casa, in spazi chiusi e impegnati in attività per la maggior parte sedentarie.

Guardare la TV, giocare ai videogame, passare il tempo con lo smartphone, il PC o il tablet diventano quindi le attività predominanti, accentuando la messa in atto di stili di vita che possono aumentare il rischio di sovrappeso, problemi osteoarticolari, disturbi del sonno, comportamenti aggressivi, irritabilità e difficoltà di concentrazione, attenzione e comprensione. Tra le diverse attività, senza dubbio, l'esercizio fisico assume un ruolo prioritario per la salute dei giovani.

Con l'esercizio fisico aumentano, infatti, le energie e lo stato di benessere generale, migliora la qualità del sonno, l'autostima, la fiducia in sé stessi e vengono scaricate eventuali tensioni.

Fare movimento anche a casa è quindi un modo semplice ed efficace per gestire lo stress, rendere attiva la mente e reagire al senso di costrizione e alla frustrazione, che la situazione attuale può generare. Allenarsi in casa di certo ha i suoi vantaggi: lo si può fare in qualsiasi momento, in sostituzione o a supporto dell'attività fisica in palestra.

## 2. PANORAMICA DEL SOFTWARE.

In questo capitolo andremo a descrivere le caratteristiche che compongono il nostro software, vogliamo fare in modo che leggendo questa parte preliminare il nostro cliente oppure un qualsiasi lettore della nostra relazione riesca a capire le parti fondamentali che compongono il nostro prodotto.

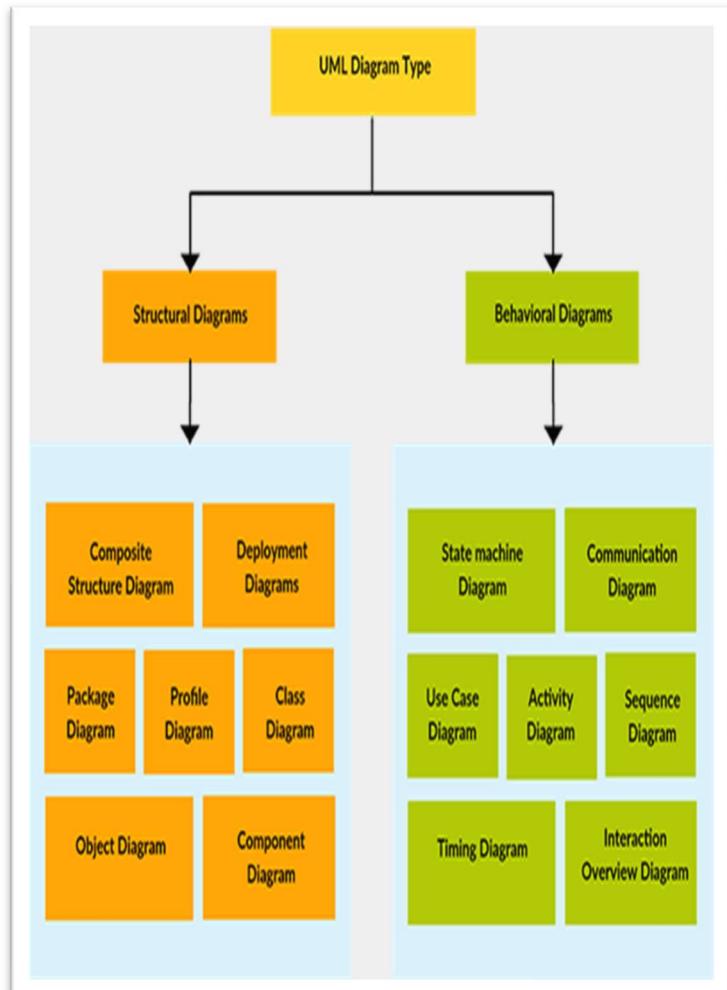
### 2.1 Linguaggi per la descrizione UML.

L'UML (Unified Modeling Language) è un linguaggio di modellazione in ambito di progettazione di software object oriented, è il linguaggio universale per modellare gli oggetti che può essere utilizzato da ogni industria produttrice di software. L'UML è, dunque, un metodo per descrivere l'architettura di un sistema in dettaglio. La forza dell'Unified Modeling Language consiste nel fatto che il processo di disegno del sistema può essere effettuata in modo tale che i clienti, gli analisti, i programmati e chiunque altro sia coinvolto nel sistema di sviluppo possa capire ed esaminare in modo efficiente il sistema e prendere parte alla sua costruzione in modo attivo. UML è un linguaggio ricco ed esteso che può essere utilizzato per modellare non solo l'ingegneria del software orientata agli oggetti, ma anche la struttura e il comportamento delle applicazioni e anche i processi aziendali. A causa della sua ampia portata, UML è il linguaggio visivo perfetto per comunicare informazioni dettagliate sull'architettura al maggior numero di utenti. È giusto precisare che per comprendere UML non è necessario avere una conoscenza tecnica

dettagliata dell'ingegneria del software. Essendo uno **standard**, è ampiamente utilizzato e accettato come linguaggio per delineare i programmi. UML viene utilizzato in una varietà di scopi e la sua leggibilità e riutilizzabilità lo rendono una scelta ideale per i programmatori. Tra i vantaggi dell'utilizzo di UML troviamo sicuramente:

- 1) **Rappresentazione visiva:** Un diagramma UML è una rappresentazione visiva delle relazioni tra classi ed entità infatti per comprendere un software basta conoscere cosa fa ogni oggetto di una classe e come si relaziona con altri oggetti di altre classi. Grazie ad UML possiamo mostrare tutte queste informazioni in un diagramma e facilitarne la comprensione.
- 2) **Leggibilità e riusabilità:** Un diagramma UML è vantaggioso in quanto è molto leggibile. Il diagramma è pensato per essere compreso da qualsiasi tipo di programmatore e aiuta a spiegare le relazioni che ci sono in un programma in modo semplice. Tradizionalmente, per comprendere un programma, un programmatore legge direttamente il codice potrebbe trattarsi di migliaia o milioni di righe di codice in programmi molto grandi avere un diagramma UML aiuta a illustrare rapidamente queste relazioni. Inoltre, utilizzando un diagramma per mostrare il codice in esecuzione in un programma, un programmatore è in grado di vedere il codice ridondante e riutilizzare parti di codice già esistenti piuttosto che riscrivere quelle funzioni.
- 3) **Pianificazione:** UML aiuta a pianificare un programma prima che abbia luogo la programmazione. Questo può aiutare a ridurre le spese generali durante la fase di implementazione di qualsiasi programma. Inoltre, un diagramma del modello UML è facile da modificare, mentre la riprogrammazione di una sezione di codice può essere noiosa e può richiedere molto tempo.
- 4) **Minore spreco di risorse:** utilizzando UML si vanno a ridurre i costi e i tempi di realizzazione di un prodotto.

Qui a fianco (*figura 2.1*) è riportato un piccolo specchietto di tutti quelli che sono i diagrammi UML che è possibile utilizzare.



*Figura 2.1*

## 2.2 Tecnologie implementative.

Data la natura del nostro progetto software, abbiamo deciso di utilizzare come linguaggio di programmazione il linguaggio Object Oriented per eccellenza ovvero **Java**. In quanto esso ci permette di fare una programmazione a moduli object based, è possibile riutilizzare il codice, è di grande aiuto per la creazione di progetti di grandi, piccole o medie dimensioni

Lo sviluppo del software, a casa della necessità di una struttura Client Server, ha reso necessario affiancare altre tecnologie. In particolare si sceglie di utilizzare un database MYSQL, in quanto si sposa perfettamente con la scelta del linguaggio java grazie ai driver JDBC (Java Database).

Per editare il codice ci siamo avvalsi dell'IDE Netbeans®.

Per scrivere la relazione è stato utilizzato il programma di video scrittura Microsoft Word®.

Per la creazione dei diagrammi presenti nella relazione ci siamo serviti della piattaforma open source Draw.io.

Per la realizzazione dell'architettura client-server abbiamo sfruttato le tecnologie messe a disposizione da XAMPP .

## 2.3 Team di sviluppo.

Il team di sviluppo del software si compone di due soli elementi che svolgono il compito sia di sviluppatori che di ingegneri del software e sono:

- Emanuele Infortuna.
- Giuseppe Primerano.

## 2.4 Requisiti team.

Per poter realizzare il prodotto software il team ha disposizione nel proprio background le seguenti risorse:

- Programmazione Object Oriented, conoscenza del linguaggio Java.
- Programmazione Procedurale, conoscenza linguaggio C.
- Programmazione Web, nello specifico la conoscenza dei linguaggi HTML, PHP, Javascript.
- Conoscenza e gestione dei database relazionali, utilizzo del linguaggio per la gestione degli stessi ovvero MySQL.
- Comprensione delle architetture web nel caso specifico client-server.
- Conoscenza delle reti e dei protocolli per lo scambio d'informazioni nello specifico HTTP.

## 2.5 Tempistiche di progetto.

Le tempistiche relative alla consegna del prodotto finito non sono state specificate, il team però prevede di consegnare il prodotto finito entro tre mesi (100 giorni circa) dall'inizio dello sviluppo. Non è previsto a priori quanti saranno i prototipi rilasciati però il piano di consegna prevede di rilasciare un prototipo ogni 30 giorni circa, per fare in modo che alla fine di ogni mese il cliente abbia a sua disposizione una release utilizzabile. Come descritto in precedenza il team di sviluppo si compone di soli due elementi perciò la suddivisione del lavoro non è fatta in modo rigido ma è basata su una continua collaborazione che garantisce una suddivisione equa delle cose da fare.

### 3. REQUISITI INFORMALI.

Per fare un'analisi dei requisiti informale bisogna sottoporre il cliente, ad una serie di domande per andare a identificare i requisiti da rispettare, le funzionalità da implementare e soprattutto evitare di deludere le aspettative degli stakeholder.

Le interviste che possono essere fatte si dividono in due diverse tipologie **aperta** e **chiusa**.

Nell'intervista aperta le domande che vengono poste al nostro cliente sono delle domande aperte cioè domande a cui è possibile dare una vasta gamma di risposte.

Fatte queste considerazioni, per le caratteristiche che compongono il nostro sistema abbiamo deciso di sottoporre il nostro cliente ad un'**intervista aperta**.

Dopo un'accurata intervista fatta al cliente sono stati identificati i requisiti informali.

#### **Partiamo descrivendo le funzionalità.**

La prima cosa che deve essere fatta dai nostri utenti è la registrazione, dove dovranno essere inserite le informazioni personali come nome, cognome, genere, email, ecc, ed anche le informazioni relative alla propria struttura corporea ovvero peso ed altezza.

Ora gli utenti hanno la possibilità di accedere ai diversi strumenti che mettiamo a loro disposizione.

Per prima cosa mettiamo a disposizione dei nostri utenti un servizio che permette di avere una panoramica corretta su quello che è il proprio corpo, infatti i nostri utenti potranno calcolare la loro massa grassa e massa magra il peso corporeo muscolare massimo, il peso corporeo ammassato massimo e il peso corporeo.

In seguito è possibile utilizzare un secondo servizio che permetterà di andare a calcolare quali sono le misure massime muscolari. Questo strumento permetterà di capire quanto possono stressare i propri muscoli, questo è uno strumento molto importante in quanto molte volte le persone che si accingono a praticare l'attività sportiva sia a livello amatoriale che a livello professionale non conoscono questi dati e vanno a stressare in maniera decisa i propri muscoli rischiando dei problemi muscolari che potranno anche diventare molto seri con il passare degli anni.

Un altro servizio messo a disposizione dei nostri utenti è il calcolo della formula di Brzycki. Questa formula permette di sapere quante ripetizioni è possibile fare con un determinato peso oppure il peso massimo che un utente può alzare.

Per ottenere le misure massime muscolari gli utenti dovranno inserire le misure della circonferenza del polso e della caviglia. L'algoritmo che implementeremo utilizzerà questi valori insieme alla percentuale della massa grassa e li utilizzerà per dire agli utenti quanto dovranno misurare al massimo i loro muscoli. I muscoli che il nostro algoritmo tiene in considerazione e che dovranno essere allenati dagli utenti sono: muscoli del petto, dei bicipiti, degli avambracci, delle cosce e infine dei polpacci. Inoltre l'algoritmo utilizzando sempre gli stessi dati che abbiamo citato in precedenza potrà informare gli utenti su quello che è il valore del loro peso corporeo, peso corporeo ammassato massimo e peso corporeo muscolare massimo. Questo è un servizio molto importante perché capita spesso di apprendere dai servizi di informazione come mass media e/o social media, notizie che raccontano di persone che per colpa di un allenamento sbagliato vanno a ledere il proprio fisico in modo irreparabile oppure adirittura rischiano la morte.

Prima di parlare di un altro servizio legato a quello descritto sopra è giusto fare una piccola digressione e introdurre un altro componente indispensabile per il nostro sistema ovvero i medici. Essi hanno un ruolo impostante in quanto, qualsiasi utente che voglia allenarsi utilizzando *Gym for You* dovrà prima sottoporsi ad un questionario stilato dal medico. Il questionario servirà da assicurazione e da “lascia passare” per l'utilizzo dei servizi di allenamento personalizzato, infatti se il medico che analizzerà il questionario darà un parere negativo l'utente non potrà utilizzare tale servizio, ciò serve per tutelare il nostro sistema e le persone che ci lavorano.

Se l'esito del questionario risulta positivo allora i nostri utenti potranno accedere al servizio che abbiamo intitolato *scheda for fou*. Questo servizio innanzitutto va a recuperare quali sono le misure massime muscolari che un utente può raggiungere, successivamente grazie all'algoritmo da noi creato capirà quale parte del corpo dovrà essere aumentata e quale no e andrà a fornire all'utente una scheda personalizzata, scheda che permetterà agli utenti di accrescere i propri muscoli senza andare a stressare troppo il fisico, proprio per evitare quelle problematiche che sono state descritte in precedenza. Inoltre per fare in modo che gli esercizi vengano eseguiti nel modo corretto mettiamo a disposizione dell'utente dei piccoli tutorial che consigliano la postura e i movimenti da eseguire.

Un altro ruolo importante all'interno del nostro sistema è sicuramente quello dell'amministratore o admin. Egli infatti rappresenta colui che ha commisionato questo prodotto che andremo a realizzare. Il suo ruolo all'interno del sistema sarà quello di vigilare sul corretto utilizzo del sistema da parte di tutti gli attori che sono coinvolti. L'amministratore è colui che sceglierà i trainer (che descriveremo in seguito) e i medici. L'admin dovrà inoltre fornire le credenziali di accesso per fare in modo che essi possano utilizzare il sistema.

Un altro ruolo interessante all'interno del sistema è quello svolto dai trainer.

Un altro servizio messo a disposizione agli utenti è quello che permette la creazione di una scheda totalmente personale. Tale scheda potrà essere creata dall'utente in base alle sue esigenze fisiche e temporali, infatti gli utenti potranno scegliere le parti del corpo da allenare, se vogliono dimagrire, oppure aumentare la loro massa muscolare, potranno addirittura andare a suddividere il loro allenamento nei diversi giorni della settimana ed anche nei diversi momenti che compongono un giorno, potranno scegliere l'intensità di allenamento per ogni giorno e infine scegliere se gli esercizi da fare dovranno essere esercizi generici oppure presi da uno dei molteplici corsi che fanno parte del nostro sistema.

In seguito alla creazione di un cosiddetto diario di allenamento entrerà in gioco l'automazione che in maniera completamente automatica andrà a creare una scheda totalmente in linea con le esigenze comunicate dall'utente. I trainer giocano un ruolo importante in quanto saranno loro a creare la scheda e nel caso in cui notano delle discrepanze in alcuni esercizi potranno modificarli in maniera semplice e veloce.

Come detto in precedenza è possibile scegliere degli allenamenti provenienti dai corsi. I corsi saranno gestiti, creati, eliminati e infine modificati solamente dall'admin.

Per noi di *Gym for You* è importante il parere dei nostri clienti, appunto per questo tutti gli utenti potranno esprimere una recensione relativa a un nostro collaboratore (trainer o medico). In questo modo medici e/o trainer potranno non solo comunicare con i nostri utenti ma potranno leggere le valutazioni che li riguardano ottenendo così feedback che li aiuteranno a migliorare il loro operato.

Il nostro progetto si pone come obiettivo quello di inserirsi nel business del fitness per renderlo accessibile a quelle persone che per diversi motivi (tra cui i costi delle palestre, la chiusura delle stesse dovuta alla pandemia, la non possibilità di raggiungerle per problemi di trasporti, ecc.) non hanno

la possibilità di allenarsi in una palestra e quindi grazie al nostro sistema offriamo la possibilità di allenarsi comodamente a casa.

## 4. MODELLO DI SVILUPPO.

Dopo aver analizzato i requisiti fondamentali che dovremmo andare a rispettare nel nostro prodotto, abbiamo fatto un'attenta ricerca di quello che poteva essere il modello di sviluppo più soddisfacente per le nostre esigenze. Si è deciso di utilizzare il modello di sviluppo RUP.

### 4.1 RUP (Rapid Unified Process).

Il progetto per lo sviluppo di questo metodo nasce nel 1988 capitanato da Jacobson. Nel 1996 viene rilasciata la prima versione denominata però ROP (Rational Objectory Process). Nel Giugno 1998 viene rilasciato Rational Unified Process (RUP). Questo è il ROP ribattezzato, e ampliato con materiale di processo ottenuto da altre società di utensili acquistate da Rational Corporation nonché materiale sviluppato dal gruppo RUP guidato da Phillippe Kruchten. Questa versione descrive come applicare i diagrammi UML nello sviluppo di sistemi basati su oggetti.

RUP è un metodo che per le sue caratteristiche viene inserito nelle metodologie agili anche se non rispetta a pieno tutte le regole delle suddette metodologie, data questa circostanza è anche possibile utilizzare una variante di RUP che prende il nome di AUP (Agile Unified Process) che va ad abbracciare a pieno le metodologie agili.

Andiamo a vedere quelle che sono le caratteristiche principali di RUP. Il Rational Unified Process è un processo di sviluppo software iterativo ed incrementale basato sui componenti, centrato sull'architettura e guidato dai casi d'uso.

- 1) Con iterativo si intende un processo non sequenziale che ripassa più e più volte su alcune attività e/o fasi di sviluppo al fine di migliorare sempre di più il prodotto.
- 2) Con incrementale si intende che ad ogni interazione viene aggiunto qualcosa al prodotto.
- 3) Con basato sui componenti si intende che il prodotto è strutturato in modo che ogni unità sia autonoma e collabori con le altre unità; favorisce lo sviluppo attraverso la programmazione ad oggetti
- 4) Con centrato sull'architettura si intende che si presuppone che si scelga una volta per tutte il tipo di "sistema" su cui gira la macchina (client-server, distribuita, locale, centralizzata, ecc.), i comportamenti

delle interfacce (come devono dialogare le varie parti), ed in generale l'organizzazione del sistema.

5) Con guidato dai casi d'uso si intende che il prodotto viene sviluppato guardando principalmente a cosa deve fare e secondariamente al come.

Lo Unified Process si basa sull'UML (*vedi* 2.1) come supporto per la documentazione tecnica che viene prodotta. Sviluppare un prodotto con l'ausilio di RUP significa procedere a piccoli passi, migliorando di volta in volta i risultati ottenuti e modellare lo sviluppo seguendo una metodologia ad oggetti, cioè si separano le competenze dei singoli moduli del programma. Tale modello è tutt'oggi molto utilizzato per lo sviluppo di prodotti di grandi o piccole dimensioni.

Il Rational Unified Process (come si vede in figura 5.1), divide lo sviluppo del software in 4 fasi; ogni fase è a sua volta costituita da 5 attività (o flussi di lavoro). Ogni fase si considera conclusa una volta prodotti i documenti previsti, mentre le singole attività avranno peso diverso a seconda della fase corrente. A differenza di quanto ci si possa aspettare il confine delle fasi è piuttosto blando e si passa da una fase all'altra con estrema naturalezza.

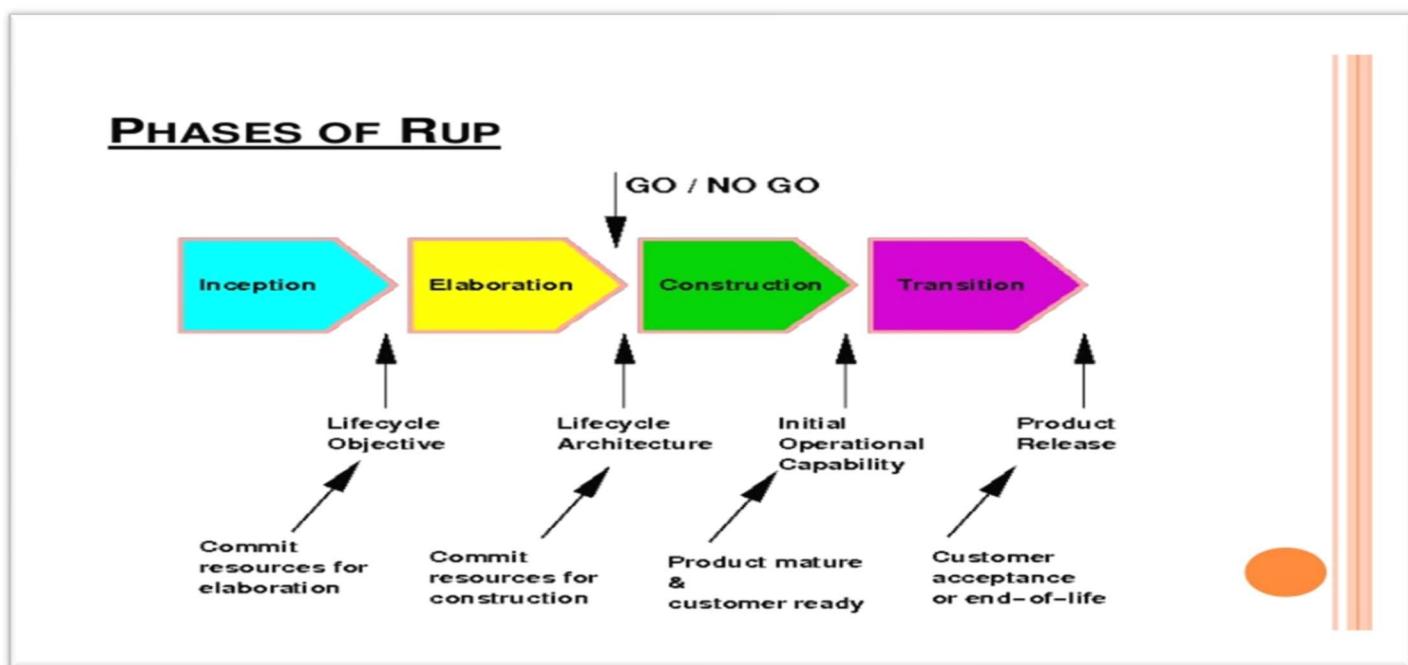


Figura 5.1

Le fasi sono le seguenti:

- 1) **Inception:** Lo scopo di questa fase è trovare gli obiettivi del ciclo di vita.
- 2) **Elaboration:** Lo scopo di questa fase è trovare l'architettura di base per il ciclo di vita.
- 3) **Construction:** Lo scopo di questa fase è la costruzione del software.

**4) Transition:** Il suo scopo è quello di rilascio del prodotto anche detto Product release.

Le attività sono le seguenti:

- 1) Requisiti:** Lo scopo di tale attività è trovare cosa deve fare il sistema.
- 2) Analisi:** Lo scopo di questa attività è raffinare e strutturare i requisiti.
- 3) Progettazione:** Lo scopo di questa attività è realizzare i requisiti in un sistema architetturale.
- 4) Implementazione:** Lo scopo di questa attività è costruire il software.
- 5) Test:** Lo scopo di questa attività è verificare che l'implementazione lavori come desiderato.

Ognuna di queste attività o flusso di lavoro compare in maniera più o meno accentuata a seconda della fase in cui si ci trova.

La nostra scelta è ricaduta su RUP per diversi motivi. Innanzitutto esso permette uno sviluppo incrementale che ci permette di suddividere lo sviluppo del nostro sistema in incrementi e di sviluppare un incremento alla volta in base alle funzionalità e ai requisiti. Permette la prototipizzazione che presenta numerosi vantaggi essa infatti fa in modo che progettista ed implementatore del software possano ottenere un prezioso feedback dagli utenti nelle prime fasi del progetto, inoltre ogni prototipo che viene realizzato non viene “eliminato”, come accade nel modello throw-away, ma è riutilizzato interamente per farlo evolvere a poco a poco nel prodotto finale. Inoltre permette di mutare le esigenze in modo indipendente sia che provengano dal cliente che dal processo stesso. È in grado di risolvere proattivamente i rischi del progetto associati alle esigenze in evoluzione del cliente che richiedono un'attenta gestione delle richieste di modifica. Rispetto ad altri modelli RUP richiede meno tempo per l'integrazione poiché il processo di integrazione continua durante il ciclo di vita dello sviluppo del software. Permette il riutilizzo dei componenti che facilità il tempo di sviluppo.

Presenta anche dei piccoli svantaggi come ad esempio il fatto che bisogna essere abbastanza esperti nell'uso di RUP. Un altro “difetto”, se così si può dire, è che si appoggia ad UML, questo fa sì che i sistemi grandi sono estremamente difficili da realizzare e mantenere ed i diagrammi UML non aiutano di certo alla comprensione.

### 5. Prima fase INCEPTION.

#### 5.1.1 Analisi dei requisiti.

I requisiti sono caratteristiche, proprietà e comportamenti che l'applicazione deve avere al termine dello sviluppo. Per effettuare l'analisi dei requisiti si individuano i casi d'uso, gli attori e le relazioni che intercorrono tra di essi. I casi d'uso descrivono le interazioni tra il sistema e l'utente, mentre gli attori rappresentano i ruoli degli utenti che interagiscono con il sistema. Essi possono anche essere dei sottosistemi. Un attore svolge uno o più casi d'uso e allo stesso modo un caso d'uso coinvolge uno o più attori.

#### 5.1.2 Descrizione stakeholder.

In questo paragrafo andremo a descrivere tutti gli stakeholder che avranno accesso al nostro software.

- **Utente:** colui che accede al servizio in modo gratuito, deve autenticarsi e inserire le proprie caratteristiche fisiche e la circonferenza dei muscoli, ha la possibilità di ottenere una scheda personalizzata prima però deve aver compilato il questionario e aver ottenuto un esito positivo dal medico. Ha accesso ai tutorial.
- **Medico:** si autentica ed ha la possibilità di comunicare con gli utenti. Il suo ruolo è quello di sottoporre un questionario agli utenti, dopo averlo analizzato ha il compito di decidere se l'utente può partecipare ai corsi oppure usufruire del servizio *scheda for you*.
- **Amministratore:** è a capo del nostro sistema ed ha il compito di gestire al meglio il suo personale dove per personale intendiamo medici e trainers. Egli deve inserire nel sistema i dati relativi ai suoi dipendenti per fare in modo che nell'autenticazione essi vengano riconosciuti come tali. Ha un ruolo manageriale. Ha un'interazione con i suoi dipendenti. Vigila sui servizi offerti dal sistema.

#### 5.1.3 Specifica dei requisiti.

Per specifica dei requisiti si intende una descrizione completa del comportamento di un sistema software da sviluppare. Per requisito si intende qualcosa che esso deve fare o per meglio dire una caratteristica che esso deve possedere. Nella SE si diversificano i tipi di requisiti infatti abbiamo requisiti funzionali e requisiti non funzionali.

## 5.1.4 Requisiti funzionali.

### **Utente può:**

- Registrarsi.
- Autenticarsi.
- Compilare il questionario creato dal medico.
- Inserire la circonferenza espressa in cm dei propri muscoli.
- Chiedere un parere al medico.
- Personalizzare il proprio allenamento con l'aiuto dell'algoritmo.
- Inserire la circonferenza espressa in cm dei propri muscoli.
- Personalizzare il proprio allenamento con l'aiuto dell'algoritmo.
- Visualizzare i tutorial.
- Caricare i propri progressi.
- Modificare le proprie credenziali.
- Modificare peso ed altezza.

### **Medico può:**

- Autenticarsi.
- Stilare il questionario.
- Analizzare e valutare il questionario.
- Rispondere all'utente per informarlo dell'esito del questionario.

### **Amministratore può:**

- Autenticarsi.
- Assumere e licenziare un dipendente.
- Visualizzare i medici.
- Modificare le credenziali di un dipendente.
- Modificare le proprie credenziali.

## 5.1.5 Requisiti non funzionali.

Di seguito è riportato l'elenco dei requisiti non funzionali del sistema.

**Usabilità:** l'interfaccia utente del sistema è stata implementata cercando di garantire la massima operabilità, un veloce apprendimento e una facile localizzazione dei comandi da utilizzare. Viene garantiti inoltre un'interfaccia coerente in tutte le sezioni dell'applicazione.

**Sicurezza:** l'applicazione gestisce informazioni sensibili, pertanto deve garantire un determinato livello di sicurezza per preservarle è stata perciò

implementata una procedura di autenticazione che permette di separare i diversi profili utente garantendo in questo modo diversi livelli di privilegi e di funzioni utilizzabili.

**Requisiti legislativi:** Come recita l'articolo 4 del Regolamento europeo che definisce il trattamento dei dati personali, come qualsiasi operazione o insieme di operazioni, compiute con o senza l'ausilio di processi automatizzati e applicate a dati personali o insiemi di dati personali. Il trattamento di dati personali può costituire un'ingerenza con il diritto al rispetto della vita privata. Qualsiasi trattamento deve, quindi, essere svolto in maniera lecita e secondo correttezza, i dati devono essere raccolti e trattati per scopi determinati, esplicativi e legittimi, e utilizzati in termini compatibili con tali scopi. Infine, devono essere conservati per un periodo non superiore al tempo necessario per raggiungere gli scopi del trattamento, trascorso il quale i dati vanno cancellati oppure anonimizzati. Quindi nel nostro sistema verranno rispettate tali regole dettate dalla legge.

## 5.2 Indagine sul sistema.

Qui andremo a fare una piccola descrizione delle linee guida del progetto prima di addentrarci nello sviluppo vero e proprio del nostro sistema. Andremo quindi a fare lo studio di fattibilità e l'analisi del contesto.

### 5.2.1 Analisi del dominio.

Il dominio applicativo del nostro software fa parte del mondo del fitness e del wellness si riferisce ad un pubblico di persone che vogliono migliorare la loro condizione fisica e mentale.

Data la delicatezza del dominio il team non è capace di gestire in maniera autonoma tutto ciò che lo riguarda per questo durante lo sviluppo del sistema si è servito dell'aiuto di persone esperte come laureandi e laureati in scienze motorie ed anche di personale medico.

Per comprendere al meglio il dominio applicativo UML mette a disposizione degli ingegneri del software un diagramma che prende il nome di diagramma di contesto (o context diagram figura 5). Il diagramma di contesto viene utilizzato per stabilire il contesto e i confini del sistema da modellare: quali cose sono all'interno e all'esterno del sistema da modellare e qual è la relazione del sistema con queste entità esterne. Abbiamo realizzato il diagramma di contesto, per definire e chiarire i confini del sistema software esso serve anche per identificare i flussi di informazioni tra il sistema e le entità esterne. L'intero sistema software viene visualizzato come un unico processo.

## Diagramma di contesto.

<b>SIMBOLO</b>	<b>SIGNIFICATO</b>
Uomo stilizzato	Attore/ambiente esterno
Freccia che parte dall'uomo	Input
Freccia che parte dal centro	Output
Doppia freccia	Input e Output
Cilindro	Database

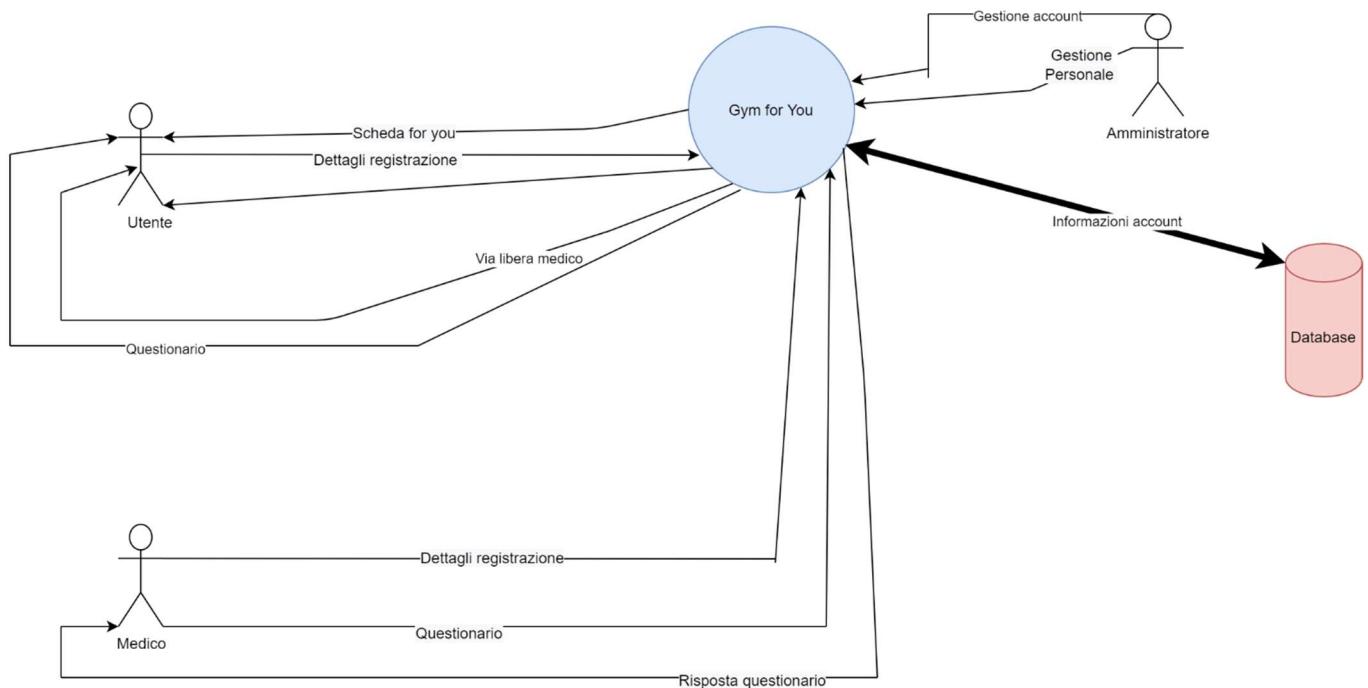


Figura 5 context-diagram

### 5.2.2 Studio fattibilità.

Uno studio di fattibilità viene utilizzato per determinare la fattibilità di un'idea, ad esempio per garantire che un progetto sia giuridicamente e tecnicamente fattibile oltre che economicamente giustificabile.

Lo studio di fattibilità dice se un progetto vale l'investimento. In generale il mancato utilizzo di quelle risorse per il tempo necessario allo svolgimento del progetto, può anche costare più di quanto l'organizzazione potrebbe guadagnare da quel determinato progetto. Uno studio di fattibilità ben progettato dovrebbe quindi offrire una serie di parametri che potremmo definire imprescindibili per la valutazione completa di un progetto. Nulla dovrebbe essere lasciato al caso.

Di seguito sono riportati alcuni vantaggi chiave della realizzazione di uno studio di fattibilità:

- Migliorare l'attenzione e la motivazione del team di progetto.
- Identificare nuove opportunità.
- Restringere le alternative commerciali.
- Identificare un motivo valido per intraprendere il progetto.
- Migliorare la percentuale di successo valutando più parametri.
- Aiutare il processo decisionale sul progetto.
- Identificare le ragioni per non procedere.

Per analizzarla meglio la fattibilità è stata suddivisa in 5 aree:

- 1) Fattibilità tecnica.
- 2) Fattibilità giuridica.
- 3) Fattibilità economica
- 4) Fattibilità operativa.
- 5) Pianificazione della fattibilità.

Di seguito analizzeremo ciascuna di esse in relazione al nostro prodotto.

### **Fattibilità tecnica.**

Questa valutazione si concentra sulle risorse tecniche disponibili per l'organizzazione. Aiuta le organizzazioni a determinare se le risorse tecniche soddisfano le capacità e se il team tecnico è in grado di convertire le idee in sistemi operativi. La fattibilità tecnica comporta anche la valutazione dell'hardware, del software e di altri requisiti tecnologici.

Il nostro team come descritto nel paragrafo 2.4 possiede tutte le conoscenze necessarie per sviluppare un sistema di questo tipo, anche essendo composto da due soli elementi. Uno degli obiettivi che ci poniamo e di creare un sistema accessibile a tutti, proprio per questo svilupperemo un sistema che richiede un hardware esiguo, facendo ciò eviteremo di stressare le nostre macchine e quelle dei clienti finali che andranno ad utilizzare *Gym for You*. Inoltre l'utilizzo del linguaggio Java che ha un'indipendenza molto elevata dalla macchina utilizzata ci permette di creare un sistema molto performante anche in ambienti diversi. Le risorse tecniche vengono soddisfatte, dal momento che il team possiede tutti gli strumenti tecnici ed è a conoscenza delle soluzioni algoritmiche e architetturali per la corretta realizzazione del software.

### **Fattibilità giuridica.**

Questa valutazione esamina se eventuali aspetti del progetto proposto sono in conflitto con requisiti legali.

Il nostro sistema rispetta appieno l'articolo 4 (vedi paragrafo 5.1.5), quindi non va in conflitto con i requisiti legali e tantomeno viola le norme relative al trattamento dei dati e al rispetto della privacy dei nostri utenti.

## Fattibilità economica.

Questa valutazione di solito comporta un'analisi costi/benefici del progetto, aiutando le organizzazioni a determinare la fattibilità, i costi e i benefici associati a un progetto prima che le risorse finanziarie siano assegnate.

Le risorse economiche richieste dal nostro sistema sono basse, questo è un vantaggio per il team di sviluppo in quanto ha la possibilità di ottenere grossi benefici mantenendo i costi relativamente bassi.

Nel diagramma in figura 5.1 vediamo rappresentata una linea di colore rosso essa rappresenta l'andamento dei costi e delle risorse in termini di tempo necessarie per lo sviluppo del nostro prodotto. All'interno del diagramma abbiamo inserito le 4 fasi del modello RUP. Analizzando l'andamento della linea rossa possiamo vedere come le prime due fasi richiedano minor tempo e costo infatti esse dipendono molto dalle fasi precedenti. Infatti come si può vedere la linea sale vertiginosamente in quanto bisogna identificare i requisiti e le funzionalità del nostro sistema e soprattutto descriverli nel migliore dei modi, di conseguenza se si procede in modo corretto nelle due prime fasi il costo e il tempo nelle fasi successive verranno decisamente abbassati. Da ciò si evince che fare bene le prime due fasi è di vitale importanza per rendere lo sviluppo del prodotto più veloce e meno costoso.

Nel secondo grafico (figura 5.2) abbiamo suddiviso i costi necessari per le diverse fasi e anche qui si può facilmente vedere che le prime due fasi richiedono costi maggiori. Abbiamo deciso di inserire questo secondo diagramma per la precisione un diagramma a torta, poiché nell'ambito economico per identificare i costi si preferisce utilizzare questa tipologia di diagramma.

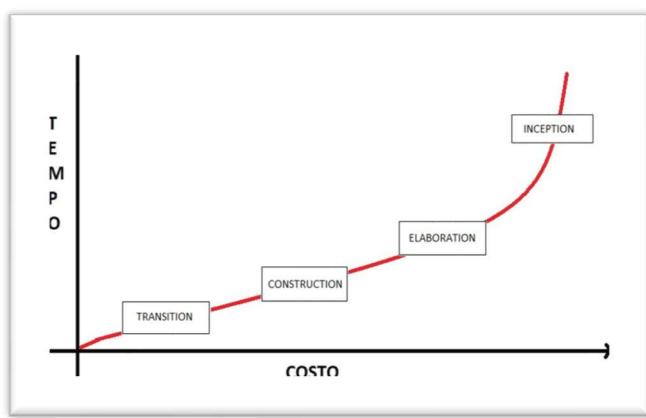


Figura 5.1

RUP COSTI

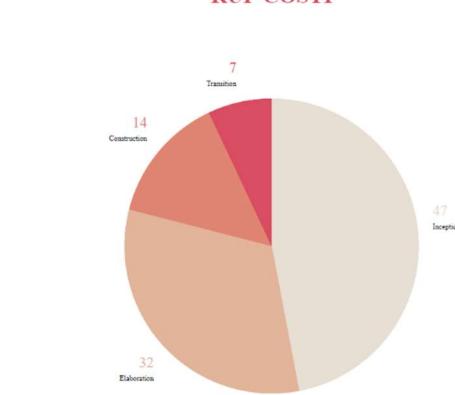


Figura 5.2 Diagramma a torta

Avendo svolto in maniere accurate l'analisi del progetto si evince che la sua realizzazione è fattibile, quindi può essere proposta al customer una quantificazione del sistema.

Come detto nel paragrafo 2.5 il limite massimo di giorni in cui il prodotto deve essere consegnato al customer è di 100 giorni, per cui per la realizzazione del software si ritiene necessario che vengano giornalmente impiegate 5 o 6 ore uomo al giorno per 5 giorni a settimana (dal lunedì al venerdì).

Analizzando i grafici riportati sopra possiamo ora andare a quantificare in modo più accurato e pratico i costi e i tempi necessari per la realizzazione di ciascuna fase.

- 1) **Inception:** 188 ore uomo necessario / 8 settimane.
- 2) **Elaboration:** 128 ore uomo necessario / 5 settimane circa.
- 3) **Construction:** 56 ore uomo necessario / 2 settimane.
- 4) **Transition:** 28 ore uomo necessarie / 1 settimana circa.

Per ottenere il calcolo delle settimane si è tenuto conto che gli sviluppatori dedicano allo sviluppo del prodotto 25 ore a settimana.

Il team quindi ritiene che in 400 ore riuscirà a consegnare il prodotto completo al cliente, inoltre riserviamo 100 ore uomo per apportare eventuali modifiche e revisioni in seguito ai feedback del customer.

Riprendendo il paragrafo 2.3, il team è composto solamente da due componenti che hanno il doppio ruolo ovvero sviluppatori e ingegneri del software, questo fa sì che possa essere applicata una divisione equa delle ore lavorative. Possiamo dunque dire che le ore complessive per la realizzazione del prodotto ammontano a 500 ore uomo che suddivise equamente diventano circa 250 ore uomo a componente.

Parlando invece di costi, il team propone un prezzo in linea con il mercato dello sviluppo software che si attesta a 17,00€ per ora uomo, quindi porterà il costo totale dello sviluppo del software a 8.500,00€ con un salario giornaliero che ammonta a 85,00€ cadauno.

Nel momento della firma del contratto abbiamo precisato al cliente che prima di iniziare lo sviluppo del software dobbiamo ricevere un acconto, appunto per questo il customer ha già versato nelle nostre casse la cifra di 500,00€.

Si vuole altresì precisare che durante la stipula del contratto il customer è stato informato che per lo sviluppo del software si è scelto un approccio incrementale e prototipale.

Il nostro sistema cercherà di rispettare una release mensile, quindi si è deciso in comune accordo con il cliente, di suddividere la cifra totale in base alle release. Sperando di riuscire a fare una release del software ogni mese il team prevede di ricevere circa 2.800,00€ a release.

### Fattibilità operativa.

Questa valutazione implica lo svolgimento di uno studio per analizzare e determinare se e in che misura i bisogni dell'organizzazione possano essere soddisfatti completando il progetto. Gli studi di fattibilità operativa analizzano anche come un piano di progetto soddisfa i requisiti identificati nella fase di analisi dei requisiti.

Il team ha individuato come metodologia operativa per portare al termine la realizzazione software uno sviluppo modulare che permetterà di suddividere il sistema in moduli e realizzare un modulo alla volta, facendo ciò si rende lo sviluppo più celere e si va a stressare di meno il team. Inoltre avendo scelto un approccio incrementale possiamo andare ad apportare continue modifiche al sistema in base ai feedback del customer e attraverso i prototipi possiamo sapere se i requisiti vengono rispettati e conoscere anche la soddisfazione del cliente.

### Tempo di fattibilità.

Questa valutazione è la più importante per il successo del progetto. Un progetto fallirà infatti, se non sarà completato in tempo. Nella pianificazione della fattibilità, un'organizzazione stima quanto tempo ci vorrà per completare il progetto con successo.

I due componenti del team di lavoro hanno avuto già esperienze nello sviluppo software in modo collaborativo per questo ritengono che dalle loro esperienze passate hanno ottenuto una padronanza delle tempistiche e riescono a gestire nel miglior modo possibile i tempi relativi alla realizzazione del progetto. Ovviamente, le stime sono tanto più rischiose e soggette ad errore quanto più il progetto è innovativo.

Per gestire al meglio i tempi abbiamo utilizzato il diagramma di Gantt (figura 5.3). Il diagramma di Gantt è uno strumento che consente la rappresentazione grafica di interi progetti e prende il nome da Henry Gantt. Esso rappresenta le attività in termini di durata e di disposizione cronologica, cioè mostra quando un'attività deve iniziare e finire. La sua

rappresentazione è basata su assi cartesiani, con le attività descritte nella WBS sull'asse verticale e il tempo sull'asse orizzontale. Proprio in riferimento alla base temporale, le attività vengono rappresentate mediante barre orizzontali.

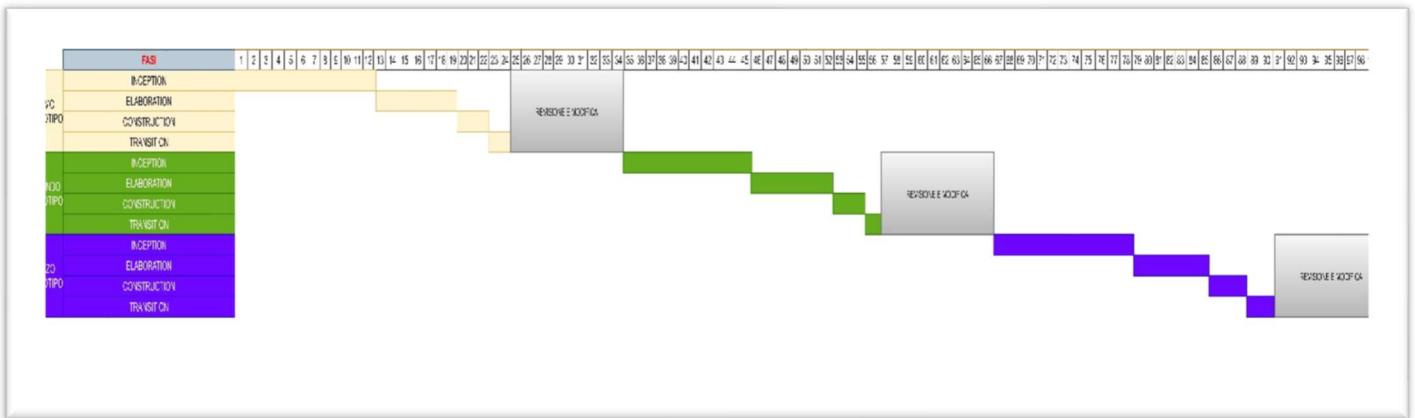


Figura 5.3 Diagramma di Gantt.

### 5.3 Analisi dei rischi.

L'analisi dei rischi prevede un' analisi sistematica e completa di tutti i possibili rischi che possono fare fallire o intralciare la realizzazione del sistema in una qualsiasi fase del processo di sviluppo. In relazione al software dobbiamo distinguere due diversi casi possibili di rischi:

- 1) Rischi di progetto.
- 2) Rischi di prodotto.

#### 5.3.1 Rischi di progetto.

Il rischio principale da evitare durante la progettazione del software è di non riuscire a rendere il prodotto adattabile ad ogni tipologia di utente. La struttura fisica di ogni persona è molto differente tra individuo ed individuo che sia di un sesso oppure di un altro. Infatti l'allenamento può avere effetti differenti dovuti proprio a questo aspetto. Per elaborare le schede di allenamento personalizzate infatti ci siamo avvalsi di uno studio (citato nella bibliografia) che ha analizzato per sei anni strutture fisiche differenti e solo dopo è riuscito ad elaborare dei calcoli che possono essere applicati, sempre con riserva, a diversi individui. Un altro rischio che va tenuto in considerazione è dato dai problemi fisici oppure malattie che possono colpire, oppure hanno colpito un nostro utente. Per questo motivo se un utente presenta dei problemi fisici nati durante l'utilizzo di *Gym for you* potrà subito comunicare con un medico che lo aiuterà a risolvere il problema, invece se presenta dei problemi antecedenti all'utilizzo del nostro

sistema l'utente potrà solamente utilizzare un numero ristretto di servizi offerti, questo grazie al questionario.

Un rischio da non sottovalutare nello sviluppo di un software è quello di non riuscire a consegnare il prodotto entro e non oltre le tempistiche pattuite ma il team tramite le proprie competenze e la scelta di modularizzare lo sviluppo del prodotto, potrà aumentare la produttività in quanto lavorando separatamente sui singoli componenti, il consolidamento finale sarà semplice e ben strutturato e nel caso di futuri update del software, le modifiche saranno anch'esse modularizzate (ottimizzandone lo sviluppo).

### 5.3.2 Rischi di prodotto.

I rischi di prodotto che si possono trovare andando a sviluppare il software sono diversi, ad esempio la difficoltà di adattare il nostro prodotto alle piattaforme su cui verrà installato.

Il Team, grazie alla propria esperienza, è a conoscenza dei requisiti per rendere un software appetibile all'utente finale, e dunque lo sviluppo delle interfacce è mirata in modo tale che l'utente riesca ad utilizzare in modo molto intuitivo quelli che sono i servizi offerti, senza creare problemi nella loro comprensione e nel loro utilizzo.

### 5.3.3 Tabella valutazione rischi.

RISCHI	VALUTAZIONE RISCHIO	PROBABILITA' CHE ACCADA	CONSEGUENZE	METODO DI RISOLUZIONE
Adattabilità software ad ogni utente.	Alto.	Poco probabile.	Utente che non riesce ad allenarsi.	Risulta molto difficile che un utente non riesca ad allenarsi in quanto il nostro sistema permette a chiunque di fare attività fisica.
Poca presenza di medici o trainers.	Alto.	Probabile.	Utente che attende troppo tempo l'esito del questionario oppure che non riesce a contattare il medico.	Sarà compito dell'admin andare ad aumentare la presenza di dipendenti nel nostro sistema per fare in modo che possano gestire una mole di utenti maggiore.
Poca voglia di usare il sistema da parte dell'utente	Medio.	Poco Probabile.	L'utente si stufa di fare sempre gli stessi esercizi e non ua più il sistema.	Il nostro sistema è composto da una miriade di esercizi, che verranno scelti casualmente.
Massime misurazioni muscolari non raggiungibili.	Alto.	Probabile.	L'utente nonostante usi il sistema non riesce ad accrescere i suoi muscoli.	Questo non dipende solo dal sistema ma anche dalle caratteristiche fisiche di ogni utente. Il nostro software è progettato per qualsiasi tipo di utente quindi garantiamo il quasi totale raggiungimento.
Difficoltà a non reperire un medico.	Medio.	Probabile.	Un utente oppure un admin non riescono a contattare un trainer o un medico.	È compito dell'admin aumentare le ore di lavoro dei suoi collaboratori per fare in modo che siano quasi sempre reperibili.
Richieste del cliente poco chiare.	Medio.	Probabile.	Il team non riesce a rispettare i requisiti.	Avendo fatto un'intervista e poi scelto una metodologia prototipale il team è certamente sicuro che i requisiti verranno rispettati.

Prodotto non funzionante su una piattaforma.	Basso.	Poco Probabile.	Il sistema non riesce a funzionare su uno o più device con piattaforme diverse.	Avendo usato Java, che ha come caratteristica principale quella di essere indipendente dalla piattaforma il rischio non si pone.
Allenamento che va a ledere la salute del nostro utente.	Basso.	Poco Probabile.	Un nostro utente ha problemi di salute derivati dall'allenamento.	Avendo implementato un meccanismo di controllo sulla salute degli utenti(questionario) e avendo utilizzato studi condotti per sei anni da personale specializzato per ottenere le schede di allenamento da svolgere, avendo un team di medici a disposizione degli utenti, il rischio è molto basso quasi inesistente.
Impossibilità di misurare il proprio corpo.	Basso.	Poco Probabile.	Un utente non riesce a misurare i propri muscoli e quindi non può utilizzare i servizi offerti.	All'interno del nostro sistema sono presenti diversi consigli su come misurare i muscoli. Dal punto di vista logistico è molto semplice, basta usare un metro da sarta 😊.
Tempistiche di consegna non adeguate.	Basso.	Poco Probabile.	Il prodotto non viene consegnato al cliente nei tempi stabiliti.	Lo studio di fattibilità condotto in precedenza e le metodologie di sviluppo software usate dal team fanno in modo che sia impossibile la non-consegna del prodotto finito.
Impossibilità di fare un esercizio per mancanza di attrezzature.	Basso.	Improbabile.	Un utente non può allenarsi perché non è munito di attrezzi per svolgere gli esercizi.	Il nostro sistema è appositamente studiato e realizzato per fare in modo che chiunque possa allenarsi, proprio per questo tutti gli allenamenti sono a "corpo libero", quindi non necessitano dell'utilizzo di attrezzi esterni.
Evoluzione delle richieste cliente.	Medio.	Probabile.	Il cliente trova il sistema non congruo a quelle che erano le richieste iniziali.	Utilizzando un approccio prototipale e un metodo come RUP è possibile anche in corso d'opera, apportare modifiche al sistema andando ad avallare le richieste del cliente.
Conflitti all'interno del team.	Basso.	Improbabile.	Il team non va d'accordo e non riesce a portare avanti lo sviluppo del software.	Il team ha dei compiti prestabiliti che non interferiscono tra di loro, ha già esperienze passate di lavoro in comune quindi il rischio è inesistente.
Impossibilità nel realizzare il sistema.	Basso.	Improbabile.	Il team non riesce a realizzare il sistema.	Il team ha fatto un attento studio di fattibilità e oltre a questo possiede ottime skill che permetteranno di realizzare il sistema senza problemi e interruzioni.

## PRIMO PROTOTIPO.

### 6 Seconda fase ELABORATION.

#### 6.1.1 Prima attività requisiti.

Prima di addentrarci nel dettaglio nell'analisi dei requisiti è possibile suddividere il nostro sistema in 5 parti che identificano in modo generale quelli che poi saranno i casi d'uso.

Le 5 parti che compongono il sistema sono:

1) Gestione accesso.

- 2) Servizi senza consenso del medico.
- 3) Servizi con consenso del medico.
- 4) Gestione salute utenti.
- 5) Gestione sistema.

### **6.1.2 Requisiti funzionali.**

Nel paragrafo 3 abbiamo descritto a grandi linee quelli che sono i requisiti funzionali. In questa fase andremo ad analizzarli in modo più accurato prendendo in considerazione le 5 parti in cui abbiamo deciso di suddividere il sistema.

#### **Requisito A: Gestione accesso UTENTE.**

**ID A.A.A:** Registra utente.

**ID A.A.B:** Login.

**ID A.A.C:** Logout.

**ID A.A.D:** Gestione profilo personale.

#### **Requisito A.B: Gestione accesso ADMIN.**

**ID A.B.A:** Login.

**ID A.B.B:** Logout.

**ID A.B.C:** Gestione profilo personale.

#### **Requisito A.C: Gestione accesso MEDICO.**

**ID A.C.A:** Login.

**ID A.C.B:** Logout.

#### **Requisito B: Servizi senza consenso del medico (UTENTE).**

**ID B.A.A:** Inserire circonference.

**ID B.A.B:** Inserire circonferenza vita.

**ID B.A.C:** Inserire circonferenza collo.

**ID B.A.D:** Inserire circonferenza fianchi.

**ID B.A.E:** Ottenere i valori della sua massa.

**ID B.A.F:** Ottenere il valore della sua massa grassa.

**ID B.A.G:** Ottenere il valore della sua massa magra.

**ID B.A.H:** Compilare il questionario.

#### **Requisito C: Servizi con il consenso del medico (UTENTE).**

**ID C.A.A:** Misure.

**ID C.A.B:** Inserisce le misure iniziali dei propri muscoli.

**ID C.A.C:** Inserisce le misure intermedie dei propri muscoli.

**ID C.A.D:** Modificare le misure intermedie dei propri muscoli.

**ID C.A.E:** Inserimento circonferenze 2.

**ID C.A.F:** Inserire circonferenza polso.

**ID C.A.G:** Inserire circonferenza caviglia.

**ID C.A.H:** Ottenere le misurazioni muscolari massime.

**ID C.A.I:** Riceve scheda per l'allenamento dall'algoritmo.

**ID C.A.L:** Ottenerne il valore del peso corporeo e del peso corporeo ammasato massimo.

#### **Requisito D: Gestione salute utenti MEDICI.**

**ID D.A.A:** Visualizza riposte questionari.

**ID D.A.B:** Valutare il questionario.

#### **Requisito E : Gestione sistema ADMIN.**

**ID E.A.A:** Visualizza dipendenti.

**ID E.A.B:** Inserisce un nuovo dipendente.

**ID E.A.C:** Elimina un dipendente.

**ID E.A.D:** Modifica le credenziali di un dipendente.

**ID E.A.E:** Gestisce il profilo personale.

### **6.1.3 Requisiti non funzionali.**

- 1) Compatibilità con diverse piattaforme.
- 2) Interfaccia GUI user friendly.
- 3) Possibilità di accedere al proprio allenamento da più e diversi dispositivi.
- 4) Utilizzo di una database per la memorizzazione dei dati.
- 5) Basso carico d'elaborazione da parte del dispositivo.
- 6) Trattamento dei dati nel maggior rispetto della privacy e rispetto delle leggi vigenti a riguardo.

## 6.1.4 Documento SRS.

<b>NOME REQUISITO</b>	<b>ID</b>	<b>PRIORITA'</b>	<b>DESCRIZIONE</b>	<b>REQUISITO PADRE</b>	<b>REQUISITO FIGLIO</b>
Gestione accesso.	A	Indispensabile.	Permette la registrazione e il login di un utente e la gestione del profilo personale.		A.A.A;A.A.B A.A.C;A.A.D; A.B.A; A.B.B; A.B.C; A.C.A; A.C.B
Registra utente	A.A.A	Indispensabile	Permette la registrazione di un utente.	A	
Login Utente	A.A.B	Indispensabile.	Permette all'utente di accedere al sistema.	A	
Logout Utente.	A.A.C	Sarebbe meglio averlo.	Permette all'utente di uscire dal sistema.	A	
Gestione profilo personale.	A.A.D	Sarebbe meglio averlo.	Permette all'utente di modificare la password, l'altezza ed il peso.	A	
Login Admin.	A.B.A	Indispensabile.	Permette all'admin di accedere al sistema.	A	
Logout Admin.	A.B.B	Sarebbe meglio averlo.	Permette all'admin di uscire dal sistema.	A	
Gestione profilo personale Admin.	A.B.C	Sarebbe meglio averlo.	Permette all'admin di modificare la password.	A	
Login Medico.	A.C.A	Indispensabile.	Permette al medico di accedere al sistema.	A	
Logout Medico.	A.C.B	Sarebbe meglio averlo.	Permette al medico di uscire dal sistema.	A	
Servizi senza consenso del medico	B	Sarebbe meglio avere..	Tutti quei servizi che l'utente può usare senza aver ottenuto un esito positivo dal medico.		B.A.A; B.A.E; B.A.H;
Inserire circonferenze	B.A.A	Indispensabile.	Permette l'inserimento delle circonferenze.	B	B.A.B B.A.C B.A.D
Inserire circonferenza vita.	B.A.B	Indispensabile.	Permette all'utente di inserire la circonferenza indicata.	B.A.A	
Inserire circonferenza collo.	B.A.C	Indispensabile.	Permette all'utente di inserire la circonferenza indicata.	B.A.A	
Inserire circonferenza fianchi.	B.A.D	Indispensabile.	Permette all'utente di inserire la circonferenza indicata.	B.A.A	
Ottenere i valori della sua massa.	B.A.E	Indispensabile.	Permette all'utente di ottenere e visualizzare i valori relativi alla massa magra e grassa del suo corpo.	B	B.A.F B.A.G
Ottenere il valore della sua massa grassa.	B.A.F	Indispensabile.	Permette all'utente di conoscere il vaolre della sua massa grassa e al nostro algoritmo di calcolare le misurazioni massime muscolari e i due diversi pesi corporei.	B.A.E	
Ottenere il valore della sua massa magra.	B.A.G	Sarebbe meglio averlo.	Permette all'utente di conoscere il vaolre della sua massa magra.	B.A.E	

Compilare il questionario.	B.A.H	Facoltativo.	Se l'utente non lo compila non può usufruire di tutti i nostri servizi.	B	
Servizi con il consenso del medico	C.	Indispensabile.	L'utente ha ottenuto l'esito positivo dal medico e adesso può usare tutti i servizi offerti.		C.A.A; C.A.E; C.A.F; C.A.H; C.A.I; C.A.L;
Misure	C.A.A	Indispensabile	Permette l'inserimento delle misure.	C.	C.A.B; C.A.C; C.A.D
Inserire le misurazioni iniziali dei propri muscoli.	C.A.B	Indispensabile.	Misure usate per ottenere la scheda di allenamento personalizzata.	C.A.A	
Inserire le misurazioni intermedie dei propri muscoli.	C.A.C	Indispensabile.	Misure usate per ottenere la scheda di allenamento personalizzata.	C.A.A	
Modificare le misurazioni intermedie dei propri muscoli.	C.A.D	Indispensabile.	Permette al nostro sistema e all'utente di ottenere informazioni sui suoi progressi e sul suo andamento nell'allenamento.	C.A.A	
Inserimento circonferenze 2.	C.A.E	Indispensabile.	Permette l'inserimento di diverse circonferenze.	C.	C.A.F; C.A.G;
Inserire circonferenza polso.	C.A.F	Indispensabile.	Permette all'utente di inserire la circonferenza indicata. Il sistema utilizzerà lo questo valore insieme alla massa grassa, per ottenere le misure massime muscolari e la scheda di allenamento personalizzata.	C.A.E	
Inserire circonferenza caviglia.	C.A.G	Indispensabile.	Permette all'utente di inserire la circonferenza indicata. Il sistema utilizzerà lo questo valore insieme alla massa grassa, per ottenere le misure massime muscolari e la scheda di allenamento personalizzata.	C.A.E	
Ottenere le misurazioni massime muscolari.	C.A.H	Indispensabile.	L'utente viene informato su quanto può spingere i suoi muscoli. Inoltre tali misure vengono usate per ottenere la scheda di allenamento personalizzata.	C.	
Riceve scheda di allenamento.	C.A.I	Indispensabile	L'utente ottiene la scheda personalizzata e può iniziare ad allenarsi.	C.	
Ottenere il valore del peso corporeo e del peso	C.A.L	Sarebbe meglio avere.	Informano l'utente sulle condizioni in cui versa il suo fisico.	C.	

corpoero ammassato massimo.					
Gestione salute utenti	D	Indispensabile.	I medici valutano il questionario e capiscono le condizioni di salute dell'utente.		D.A.A D.A.B
Visualizza risposte questionari	D.A.A	Indispensabile.	Il medico vede le risposte che ha dato l'utente.	D	
Valutare il questionario	D.A.B	Indispensabile.	I medici valutano il questionario e danno un esito.	D	
Gestione sistema.	E	Indispensabile.	Permette all'admin di gestire il sistema che stiamo progettando.		E.A.A; E.A.B; E.A.C; E.A.D; E.A.E
Visualizza dipendente.	E.A.A	Sarebbe meglio avere.	L'admin visualizza tutti i suoi dipendenti.	E	
Inserisce un nuovo dipendente.	E.A.B	Indispensabile.	L'admin inserisce un nuovo dipendente.	E	
Elimina un dipendente.	E.A.C	Sarebbe meglio avere.	L'admin elimina un dipendente.	E	
Modifica le credenziali di un dipendente.	E.A.D	Sarebbe meglio avere.	L'admin modifica la password di un dipendente.	E	
Gestisce il profilo personale.	E.A.E	Sarebbe meglio avere.	L'admin modifica la propria password.	E	

## 6.1.5 Diagramma dei casi d'uso.

### Legenda:

Simbolo	Significato
Uomo stilizzato	Attore
Ellisse	Caso d'uso
Freccia grande	Generalizzazione
Freccie rosse	Inclusione o estensione.

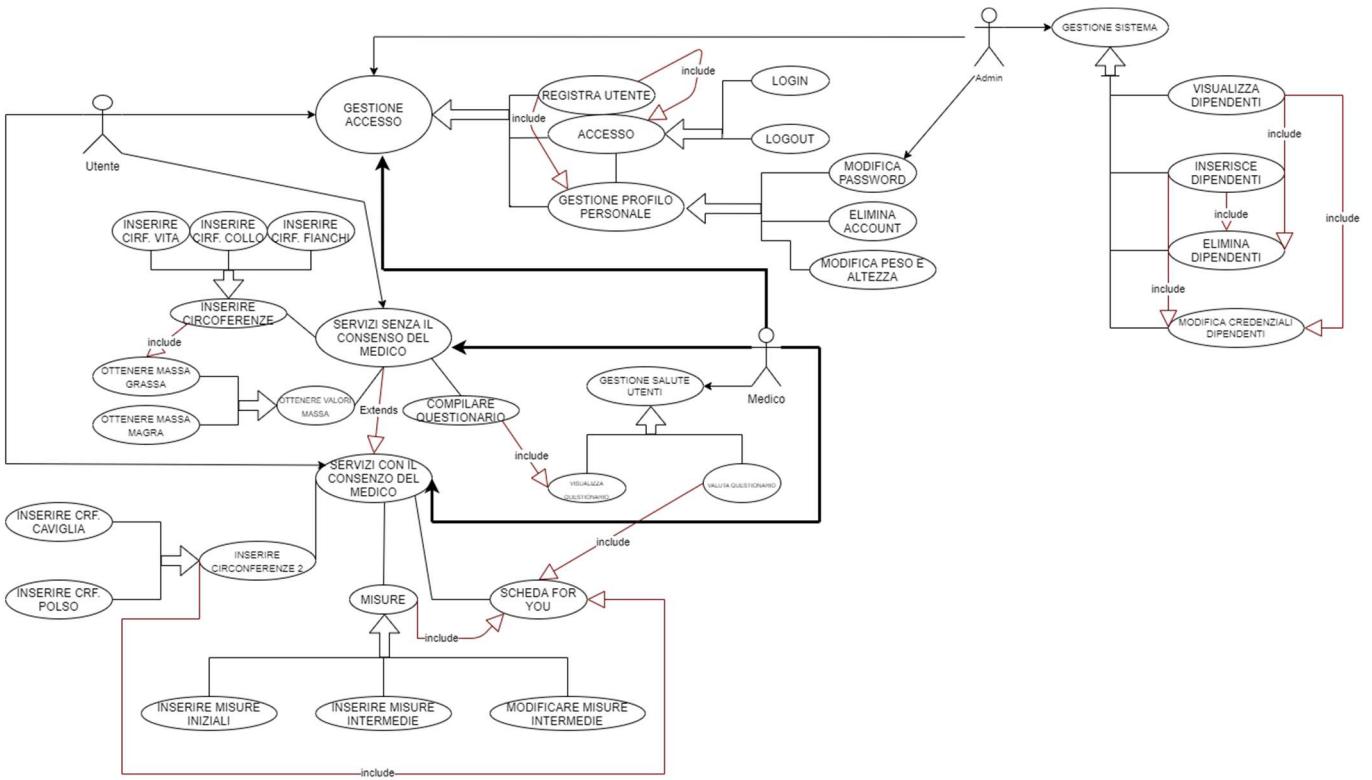


Figura 6.0 Diagramma dei casi d'uso

### 6.1.6 Descrizione attori.

<b>ATTORE</b>	Utente.
<b>ID</b>	1
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Egli è colui il quale userà il nostro sistema per allenarsi oppure per tenere d'occhio la sua massa grassa e magra.

<b>ATTORE</b>	Medico.
<b>ID</b>	2
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Egli ha il compito di verificare se l'utente che ha compilato il questionario può fare attività fisica, quindi solo dopo il suo consenso l'utente potrà usare i servizi di allenamento e non solo offerti dal nostro sistema.

<b>ATTORE</b>	Admin.
<b>ID</b>	3
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Ha un ruolo manageriale all'interno del sistema. Gestisce le assunzioni oppure i licenziamenti dei dottori.

### 6.1.7 Descrizione casi d'uso.

Caso d'uso	Gestione accesso.
ID	A.B
Descrizione	L'admin utilizza questa operazione per accedere alla piattaforma. Abbiamo delle sotto-operazioni che permettono all'admin di gestire il proprio profilo personale. Altre sotto-operazioni permettono di entrare nella piattaforma (log-in) e di uscire (log-out).
Attori	Admin.
Pre-condizioni	Conoscere le proprie credenziali.
Flusso principale	Accede al suo pannello
Scenario secondario.	L'admin non inserisce bene le sue credenziali quindi non entra nel suo pannello.
Flusso alternativo.	Non accede al pannello e viene visualizzato un pop up con l'errore.
Post-condizioni	Accesso al pannello admin.

Caso d'uso	Gestione accesso.
ID	A.C
Descrizione	Il medico utilizza questa operazione per accedere alla piattaforma. Abbiamo delle altre sotto-operazioni che permettono di entrare nella piattaforma log-in e di uscirne log-out.
Attori	Medico.
Flusso principale	Accede al suo pannello.
Scenario secondario.	Il medico non inserisce bene le sue credenziali e non entra nel suo pannello.
Flusso alternativo.	Non accede al pannello e viene visualizzato un pop up con l'errore.
Pre-condizioni	Conoscere le proprie credenziali fornite dall'admin.
Post-condizioni	Accesso al pannello medico.

Caso d'uso	Gestione accesso.
ID	A
Descrizione	L'utente prima di poter accedere alla piattaforma deve fare la sotto-operazione chiamata <b>registrazione</b> . Abbiamo delle altre sotto-operazioni che permettono di entrare nella piattaforma log-in e di

	uscirne log-out. Abbiamo un'altra sotto-operazione che permette all'utente di gestire il proprio profilo personale.
Attori	Utente.
Pre-condizioni	Conoscere le proprie credenziali.
Flusso principale	Accede al suo pannello.
Scenario secondario.	Sbaglia ad inserire email e password durante il login, non inserisce in modo corretto la password durante la registrazione oppure nel nostro database è già presente un'email uguale a quella inserita dall'utente.
Flusso alternativo.	Non accede al pannello e viene visualizzato un pop up con l'errore.
Post-condizioni	Accesso al pannello utente.

Caso d'uso	Servizi senza il consenso del medico.
ID	B
Descrizione	L'utente una volta registratosi e fatto il log-in ha a sua disposizione delle sotto-operazioni che possono essere usate senza che il medico abbia dato esito positivo al questionario. Le sotto-operazioni sono tre: <b>inserire circonferenze</b> che permette l'inserimento di alcune circonferenze; <b>ottenere valori massa</b> una volta inserite le cinconferenze calcola la massa magra e grassa dell'utente; <b>compilare questionario</b> permette all'utente di compilare ed inviare il questionario al medico, così che possa essere fatta la valutazione.
Attori	Utente.
Flusso principale	Utilizza i servizi in modo corretto.
Scenario secondario	Inserisce in modo sbagliato le circonferenze.
Flusso alternativo	I risultati che dipendono dalle circonferenze inserite saranno errati.
Pre-condizioni	Avere effettuato l'accesso.
Post-condizioni	Ottenimento vaolri massa e invio questionario.

Caso d'uso	Servizi con il consenso del medico.
------------	-------------------------------------

ID	C
Descrizione	L'utente dopo aver ricevuto l'esito positivo dal medico può fare diverse sotto-operazioni legate a questo caso d'uso: <b>Inserire</b> altre circonference, esse serviranno per accedere ad altre operazioni tra cui <b>ottenere peso</b> . Altra sotto-operazione è quella che permette di inserire le proprie <b>misure muscolari</b> esse serviranno per ottenere la scheda di allenamento personalizzata.
Attori	Utente.
Flusso principale	Utilizza i servizi in modo corretto.
Scenario secondario.	1) Inserisce in modo sbagliato sia le circonference che le misure iniziali, oppure sbaglia a modificare quelle intermedie. 2) L'utente raggiunge il massimo consentito dai suoi muscoli.
Flusso alternativo.	1) Tutto ciò che dipende dagli input citati sopra darà un output errato. 2) Appare una finestra pop-up con i complimenti.
Pre-condizioni	Avere effettuato l'accesso ed avere l'esito del questionario positivo.
Post-condizioni	Panoramica sulla struttura muscolare dell'utente e scheda personalizzata.

Caso d'uso	Gestione salute utenti.
ID	D
Descrizione	Il medico accede al proprio pannello e può visualizzare le domande e le risposte date al questionario da ciascun utente, in base ad esse può dare un esito negativo o positivo.
Attori	Medico.
Flusso principale	Utilizza i servizi messi a sua disposizione.
Scenario secondario.	Dimentica o sbaglia le credenziali durante il log-in.
Flusso alternativo.	Il medico non riuscirà ad entrare nel suo pannello. Appare un pop-up che espone al medico per quale motivo non riesce ad accedere.
Pre-condizioni	Avere effettuato l'accesso.
Post-condizioni	Verranno sbloccate determinate funzionalità all'utente.

Caso d'uso	Gestione sistema.
------------	-------------------

ID	E
Descrizione	L'admin ha a sua disposizione 3 sotto-operazioni tra cui: <b>visualizzare</b> i suoi dipendenti, <b>aggiungere</b> un dipendente, <b>modificare</b> la password di un dipendente nel caso in cui essa venga dimenticata.
Attori	Admin.
Flusso principale	Utilizza i servizi messi a sua disposizione.
Scenario secondario.	Sbaglia le credenziali durante il login.
Flusso alternativo.	L'admin non può usare i servizi messi a sua disposizione.
Pre-condizioni	Avere effettuato l'accesso.
Post-condizioni	Assunzione o licenziamento di dipendenti.

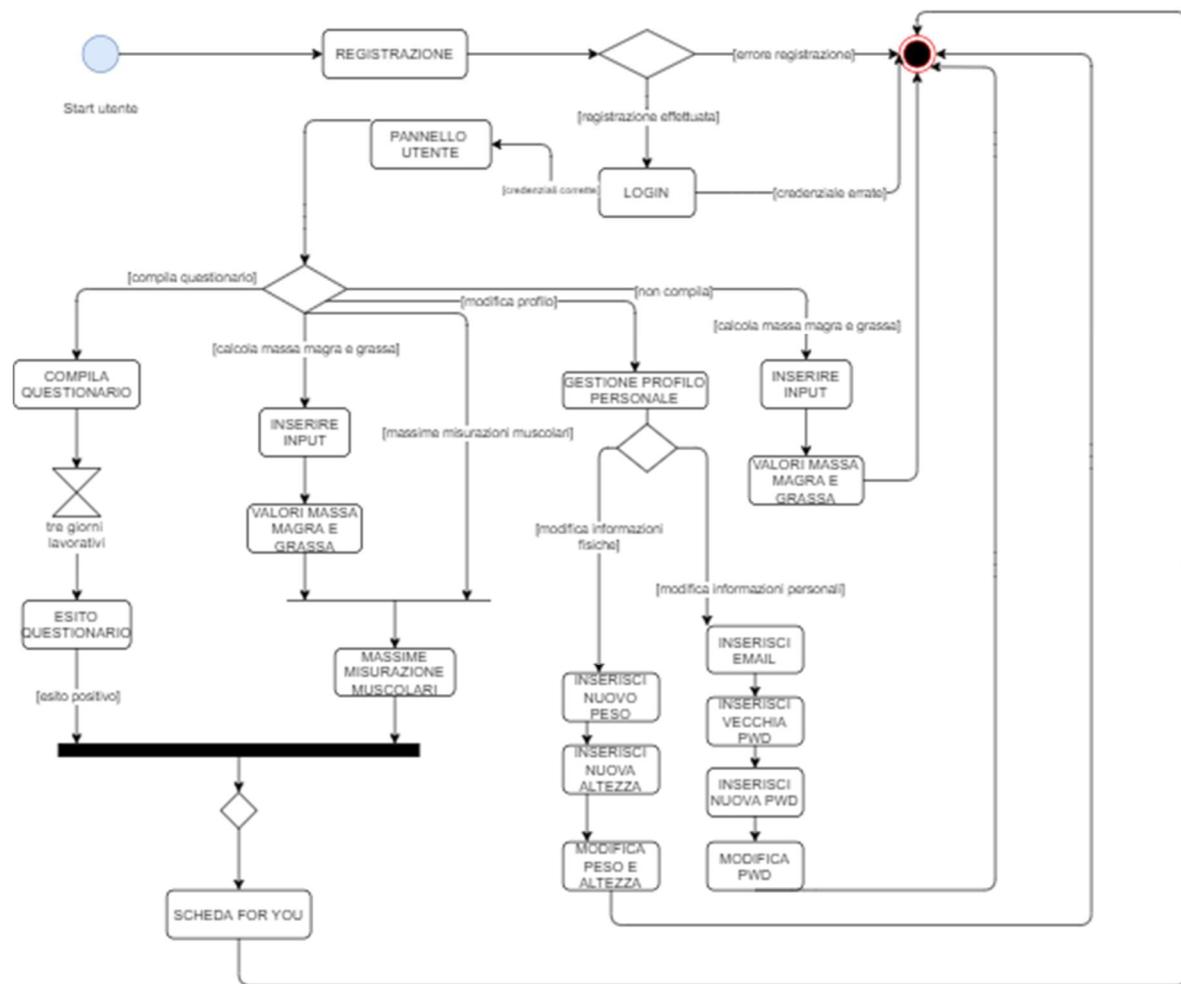
### 6.1.8 Diagramma delle attività.

Questo diagramma si aggiunge alla documentazione che può essere presentata. La sua funzione è quella di prendere le descrizioni dei flussi sia quello principale che quelli alternativi, sono fusi nello stesso diagramma per poter far vedere l'insieme delle azioni compiute dal caso d'uso preso in considerazione.

#### Legenda:

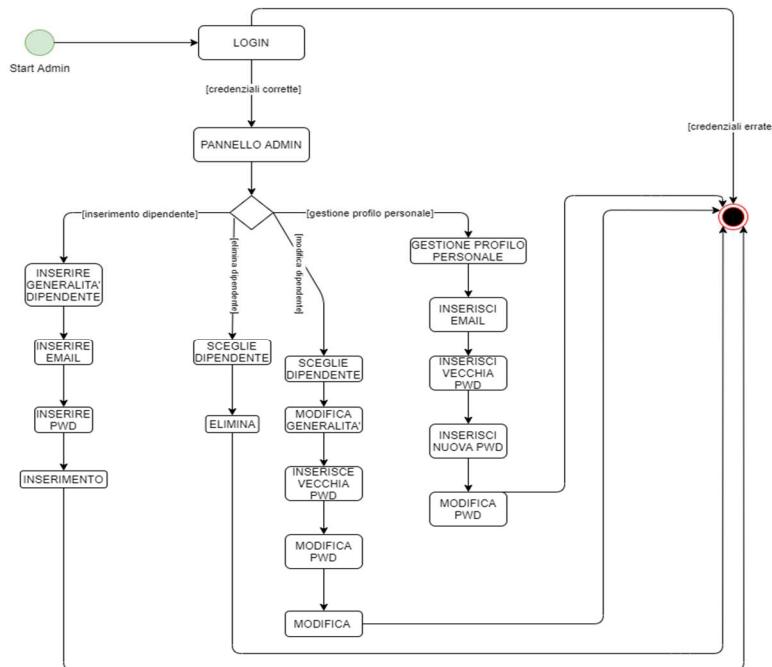
Simbolo	Significato
Cerchio pieno	Punto d'inizio.
Rettangolo con bordi smussati	Attività
Rombo	Punto di scelta

## Diagramma delle attività Utente.



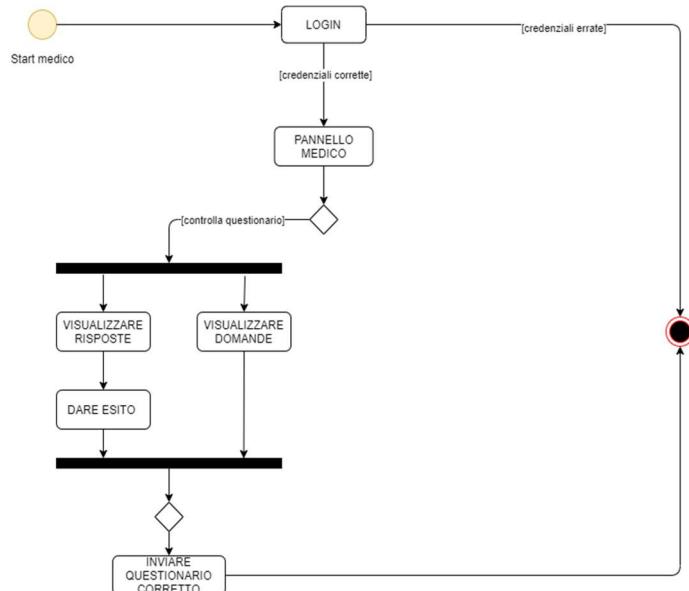
## *Activity Diagram Utente*

## Diagramma delle attività Admin.



## *Figura Activity Diagram Admin*

## Diagramma delle attività Medico.



## *Figura Activity Diagram Medico*

# PRIMO PROTOTIPO.

## 7. Seconda attività: ANALISI.

Scopo dell'attività di analisi è quello di creare modelli che catturano il comportamento desiderato del sistema. L'attività di Analisi inizia non appena sono pronti i primi requisiti e costruisce le classi di analisi. Il

modello di analisi descrive la storia del sistema, cattura l'insieme e contiene i modelli appartenenti al dominio del problema (quindi con esclusione di classi di comunicazione o di database). È proprio durante questa attività che si stabilisce l'architettura di base. L'architettura di base del software è costituita dalle strutture del sistema, che comprendono i componenti software, le loro proprietà visibili e le relazioni fra di essi.

Tutta l'attività, dunque, verte su:

- 1) Analisi architetturale.
- 2) Individuazione delle classi di analisi.
- 3) Analisi dei casi d'uso.

## Analisi architetturale.

### 7.1 Diagramma dei package.

Il nostro sistema è composto da 6 diversi package, al loro interno troviamo le classi che lo compongono.

I diagrammi dei package vengono utilizzati per strutturare elementi di alto livello del sistema. I package vengono utilizzati per organizzare un sistema di grandi dimensioni che contiene diagrammi, documenti e altri risultati chiave. Il diagramma dei package può essere utilizzato per semplificare diagrammi di classi complessi infatti può raggruppare classi in pacchetti.

Nel diagramma (figura 7.0) abbiamo rappresentato tutti i package realizzati finora. Come si può vedere tali package sono collegati fra di loro da linee tratteggiate che indicano le dipendenze. Per una maggiore comprensione abbiamo deciso di dividere le dipendenza in dipendenze **access** e dipendenze **import**.

- Le dipendenze <> access>> indicano che un package accede agli elementi pubblici dell'altro package, facendovi riferimento in modo esplicito. Quindi i collegamenti che abbiamo fatto indicano che un package utilizza i metodi contenuti in un altro package.
- Le dipendenze <> import>> indicano che un package accede agli elementi pubblici dell'altro package, senza farvi riferimento in maniera esplicita, ma importando al proprio interno il namespace, cioè ogni volta che si fa riferimento ad un oggetto, e questo non appartiene al package corrente viene ricercato tra gli elementi pubblici del package su cui vi è la dipendenza import.

## Legenda:

Simbolo	Significato
Rettangolo	Package.
Freccia tratteggiata	Dipendenze.

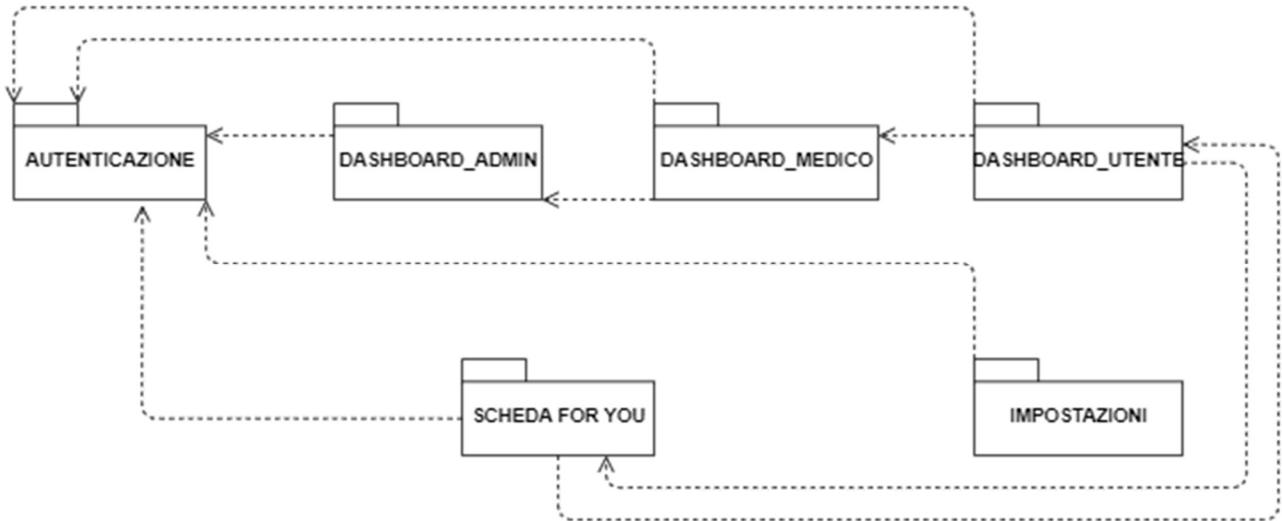


Figura 7.0 Diagramma dei pacchetti.

## 7.2 Approccio BCE

Per la realizzazione delle classi il team ha deciso di utilizzare l'approccio BCE (Boundary-Control-Entity). Il team ha deciso di usare quest'approccio perché una tecnica buona nella programmazione ad oggetti è quella di dividere la visualizzazione, dall'implementazione, dai dati associati altro vantaggio di questo approccio sta nel raggruppamento di classi in strati gerarchizzati, ciò migliora la comprensione del modello e ne riduce la complessità. BCE divide le classi in tre categorie:

- 1) Classe Boundary: descrive gli oggetti che rappresentano l'interfaccia tra un attore ed il sistema. Rappresenta una parte dello stato del sistema che è presentata all'utente in forma visiva;
- 2) Classe Control: descrive gli oggetti che percepiscono gli eventi generati dall'utente. Rappresenta un supporto per le operazioni e le attività di un caso d'uso.
- 3) Classe Entity: descrive gli oggetti che rappresentano la semantica delle entità nel dominio applicativo. Corrisponde ad una struttura dati nel database del sistema.

Le classi boundary corrispondono dunque alla GUI, le classi control corrispondono alle funzionalità del sistema, mentre quelle entity agli oggetti che rappresentano i dati nella memoria del sistema. Le classi boundary dialogano con l'utente e con le classi control, le classi control dialogano con le classi boundary ed entity. L'approccio BCE è un “modo di pensare” che riflette una buona pratica di ingegneria del software e come tale dovrebbe essere parte di qualsiasi metodo di sviluppo delle applicazioni, indipendentemente dalla sottostante piattaforma d'implementazione.

### Individuazione delle classi di analisi.

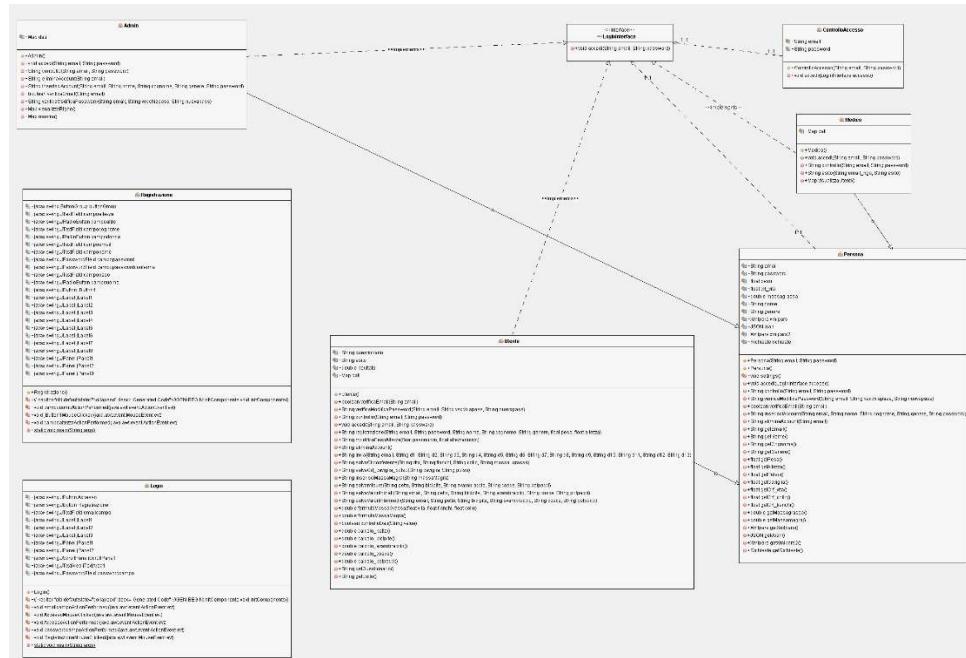
## 7.3 Diagramma delle classi.

Di seguito andremo a mostrare tutti i diagrammi delle classi che riguardano il nostro sistema. I diagrammi verranno divisi in base ai package che compongono il nostro sistema. Per ottenere i diagrammi ci siamo serviti del plug-in (easyUML) dell'IDE Netbeans. I seguenti diagrammi consentono di descrivere le entità e le loro caratteristiche ma anche le eventuali relazioni che intercorrono tra loro.

### Legenda:

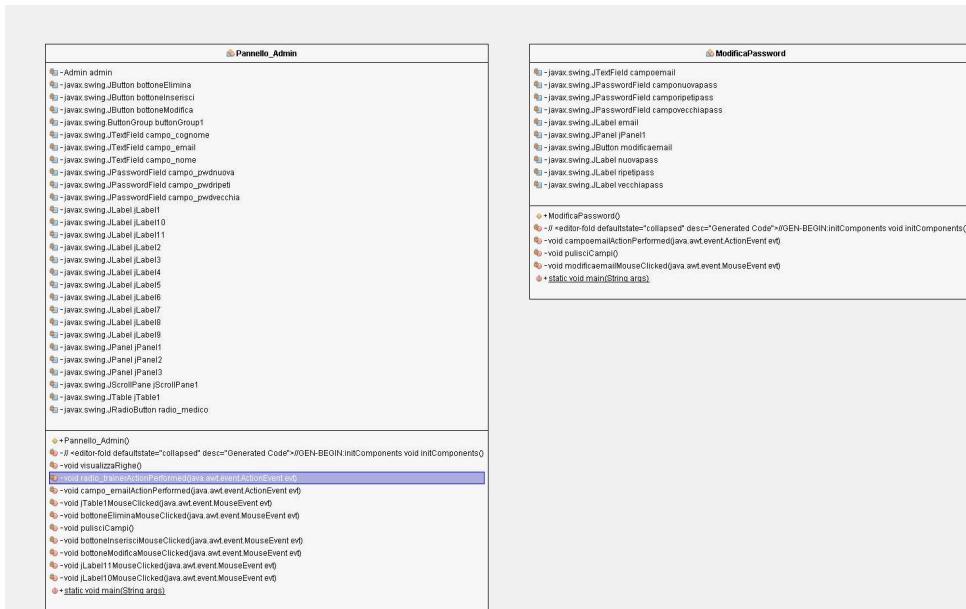
Simbolo	Significato
Rettangolo	Package.
Secondo rettangolo	Attributi.
Terzo rettangolo	Operazioni.
Freccia	Associazioni.

## Analisi delle classi: Autenticazione.



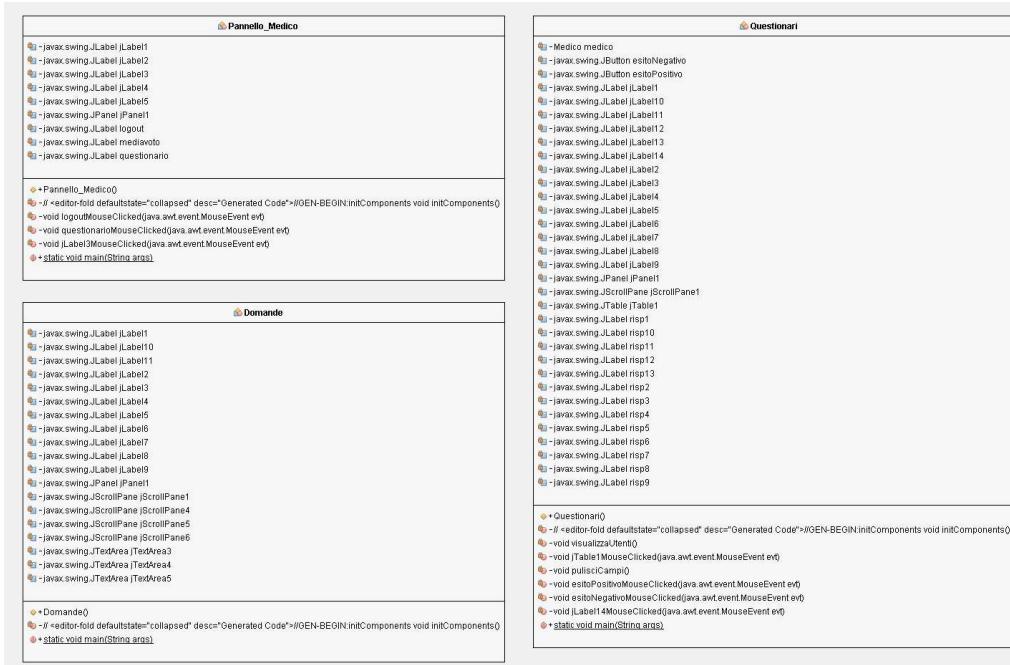
*Figura 7.1 Diagramma delle classi package autenticazione.*

## Analisi delle classi: Dashboard admin.



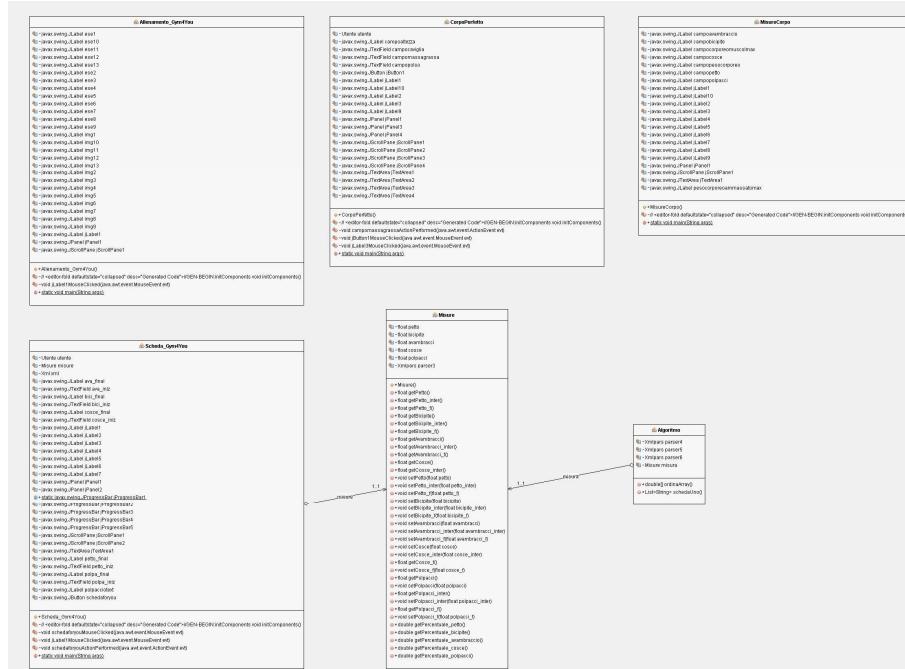
*Figura 7.2 Diagramma delle classi package dashboard\_admin*

## Analisi delle classi: Dashboard medico.



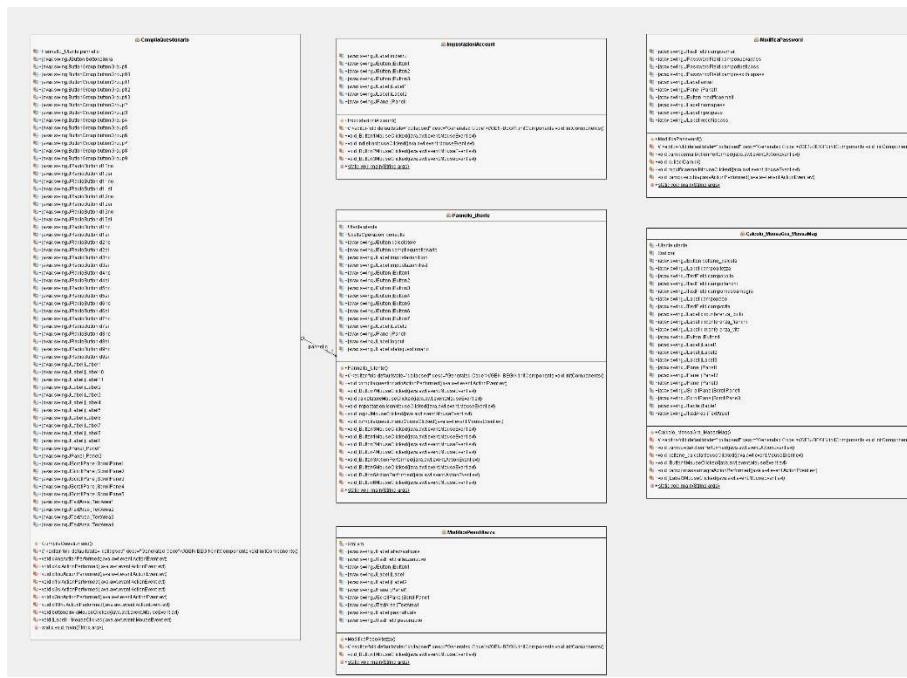
*Figura 7.3 Diagramma delle classi package dashboard\_medico.*

# Analisi delle classi: Scheda for you.



*Figura 7.4 Diagramma delle classi package scheda for you*

## Analisi delle classi: Dashboard utente.



*Figura 7.5 Diagramma delle classi package dashboard\_utente*

## Analisi dei casi d'uso.

## 8. Diagramma di sequenza.

Come si potrà vedere dalle figure che verranno inserite in seguito tutte le operazioni fatte dagli stakeholder, quindi dal client verranno prima processate dal server e solo dopo inviate al database per poter essere eseguite, dopodiché il server invierà al client un messaggio di errore oppure di conferma delle operazioni, sarà quindi il client ad occuparsi di mostrarlo all'utente. Questo viene fatto perché si è deciso di utilizzare l'architettura client-server che verrà descritta con maggiore attenzione nei capitoli successivi.

## Diagramma di sequenza Admin.

Nel seguente diagramma di sequenza (figura 8) andiamo ad analizzare la sequenza delle operazioni che vengono fatte dall'admin. Innanzitutto viene fatto il login inserendo le proprie credenziali, se le condizioni (indicate nel frame) vengono rispettate allora si può entrare nella piattaforma dedicata. Poi abbiamo le operazioni di inserimento dove l'admin inserisce nei campi appropriati le generalità e le credenziali del nuovo dipendente, se tutto va a buon fine appare un messaggio che informerà dell'avvenuto inserimento, altrimenti apparirà un messaggio d'errore. Nella modifica della password del dipendente, l'admin invierà al server l'email del dipendente e la sua vecchia password, si procederà con un controllo e se i dati inviati sono

corretti allora avverrà la modifica, altrimenti verrà visualizzato un messaggio d'errore. L'ultima cosa che l'admin può fare è modificare la propria password, per fare ciò invierà al server la sua email e la vecchia password, se sono corretti e anche la nuova password è corretta allora verrà fatta la modifica. Altrimenti se l'email è errata oppure le password non coincidono apparirà all'admin un messaggio di errore.

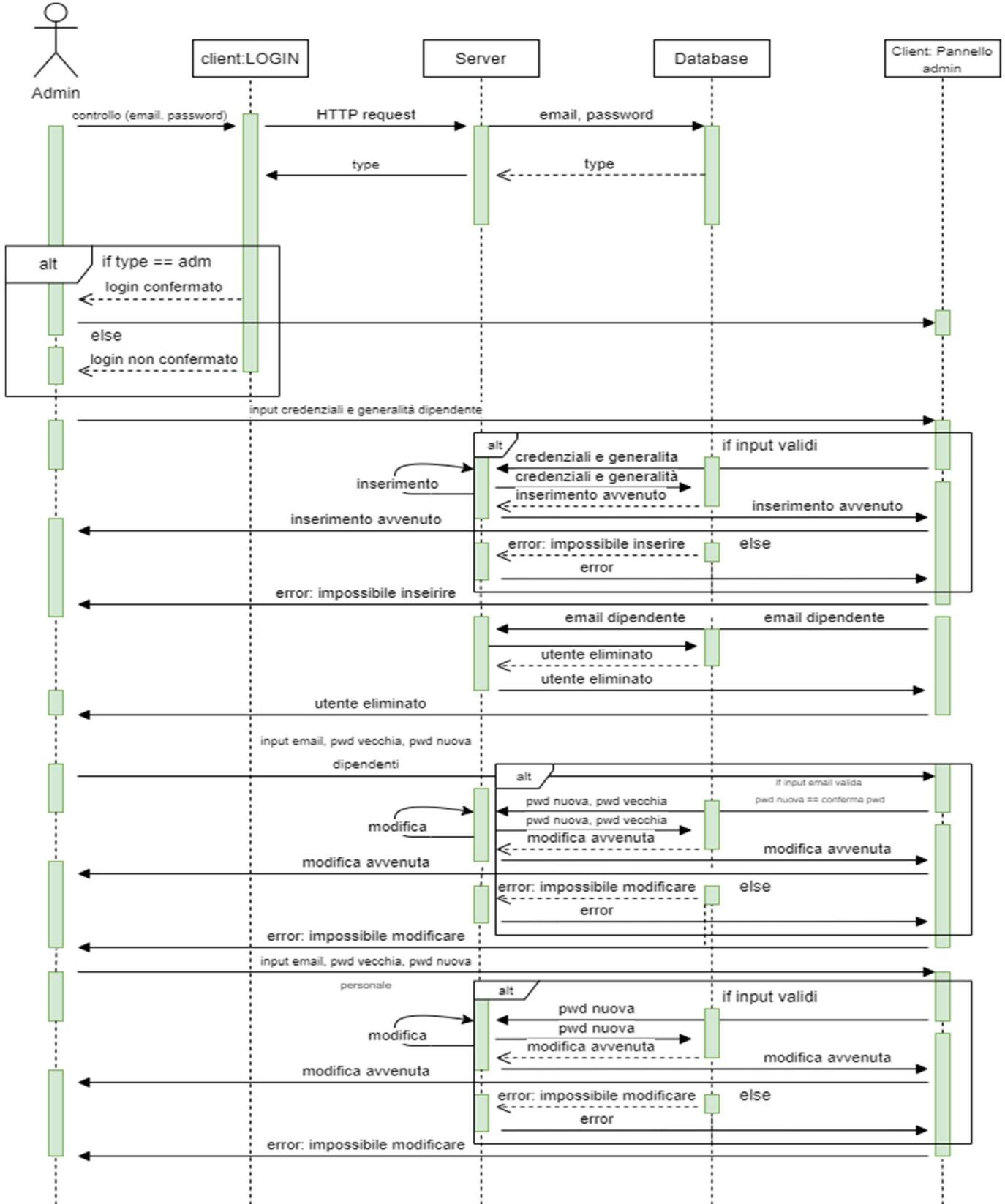


Figura 8.0 Diagramma di sequenza admin.

## Diagramma di sequenza Medico.

Nel seguente diagramma (figura 8.1), mostriamo come si relaziona il medico con il nostro sistema. Innanzitutto il medico, grazie alle credenziali fornitegli dall'admin può accedere al sistema, dopodiché all'interno del proprio pannello ha a sua disposizione tutti i questionari compilati che vengono prelevati dal database passati al server e poi mostrati nel pannello dedicato, cliccando su un questionario può visualizzare le risposte e in base ad esse può dare una valutazione positiva o negativa, questa valutazione verrà inviata al client che la passerà al server per poi essere salvata nel database.

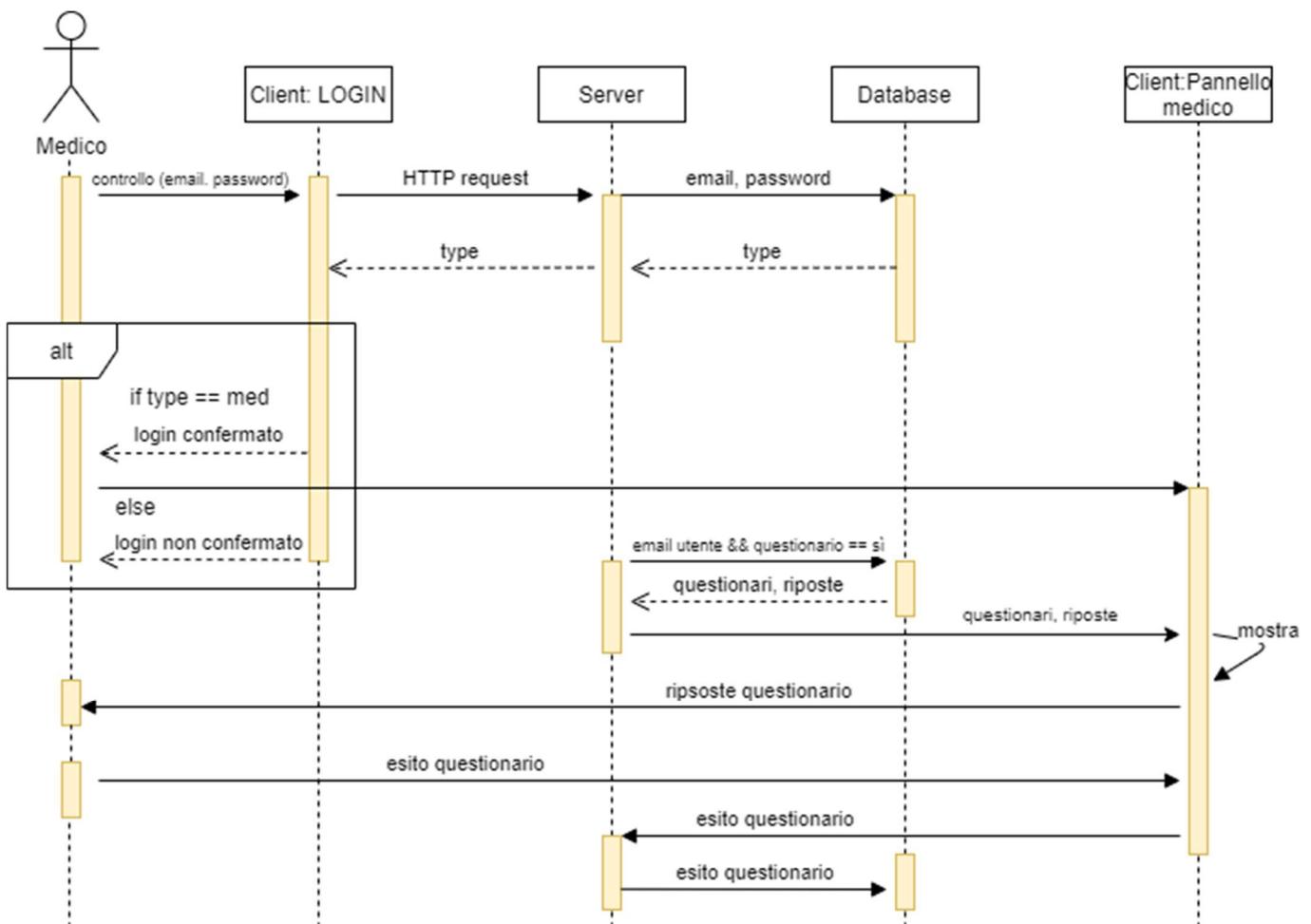


Figura 8.1 Diagramma di sequenza medico.

## Diagramma di sequenza Utente.

Come si può vedere dal diagramma (figura 8.2), per prima cosa il nostro utente deve registrarsi, se gli input inseriti durante la registrazione sono corretti il nostro nuovo utente verrà aggiunto al sistema. In seguito l'utente deve effettuare il log-in, se le credenziali saranno corrette l'utente ha accesso alla propria interfaccia. Al suo interno trovare dei servizi che potranno essere usati senza aver compilato il questionario. Per ottenere massa grassa e magra egli deve inserire le proprie misure che verrano passate dal client al server, dopodichè avverrà il calcolo della massa grassa e magra che verranno salvate nel nostro database e poi mostrate all'utente. Se l'utente compila il questionario può innanzitutto inserire la circonferenza del polso e dalla caviglia che verranno inviate dal client al server, esse serviranno per calcolare le misurazioni massime muscolari, il peso corporeo massimo e il peso corporeo ammassato massimo. Il sistema avendo a disposizione le informazioni elencate prima perché salvate nel database, può creare una scheda di allenamento personalizzata per l'utente. Se gli input risultano errati l'utente riceverà un messaggio di errore. Inoltre l'utente può modificare diverse cose tra cui l'altezza, il peso, le misure attuali dei propri muscoli e la propria password, ciò avviene solo se vengono inseriti correttamente, il controllo viene fatto al momento dell'invio dei dati dal client al server, altrimenti apparirà un messaggio di errore. L'ultima cosa che può fare l'utente è eliminare il proprio account, per fare ciò viene passata l'email dal client al server, successivamente il server la invia al database che si occupa di eseguire la query, infatti verranno eliminate tutte le righe delle tabelle in cui è presente quell'email.

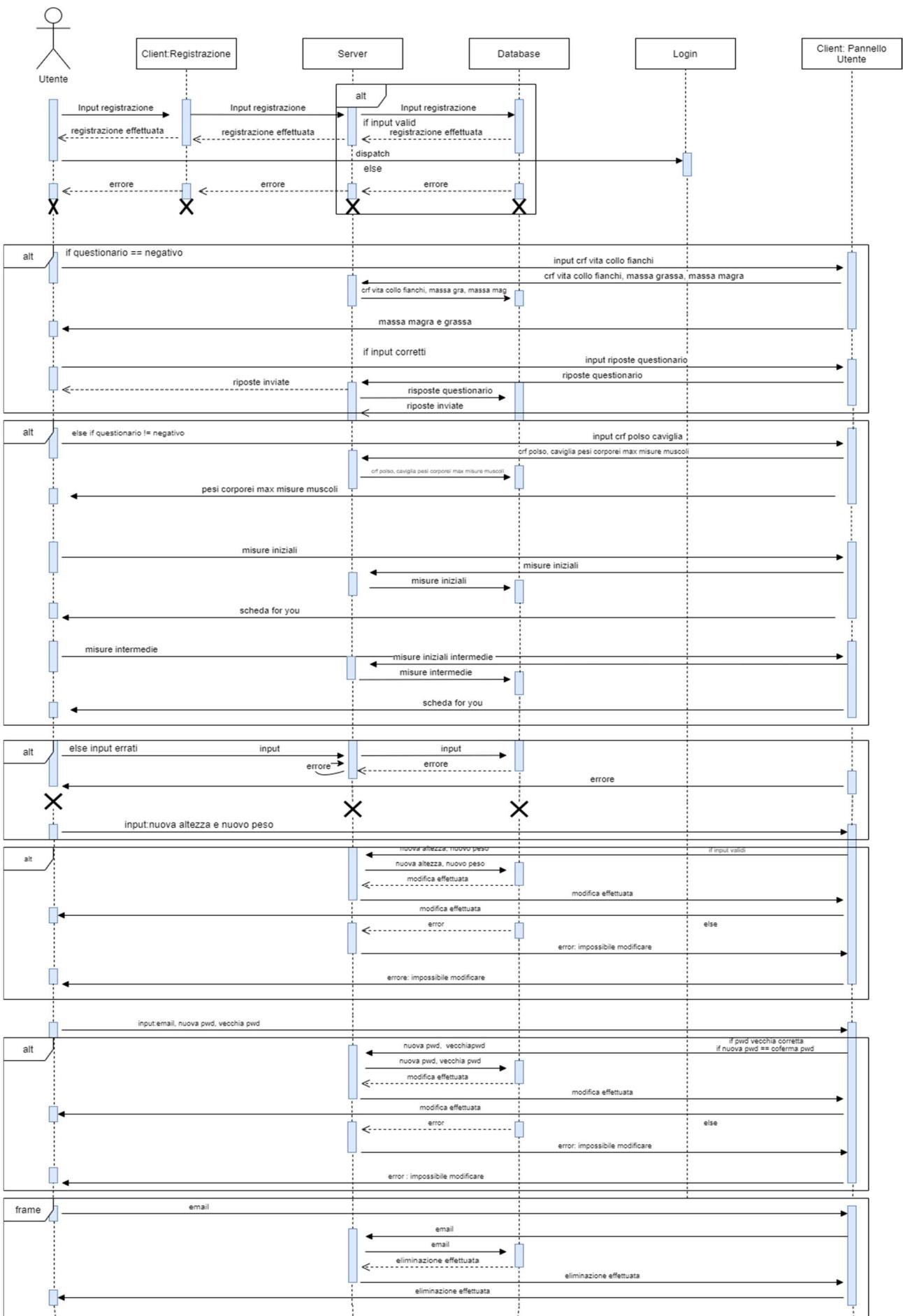


Figura 8.2 Diagramma di sequenza utente.

## 9. Terza attività: PROGETTAZIONE.

L'attività di progettazione verte sul come deve funzionare il sistema, piuttosto che sul cosa deve fare. L'attività di Design riprende, dunque, i documenti prodotti dalla fase di analisi e vi aggiunge elementi appartenenti al dominio della soluzione. Il modello di progettazione va a guardare l'implementazione tecnica, osservando i requisiti non funzionali e l'architettura sottostante, mette maggiore risalto sulle interfacce. È durante questa attività che l'architettura di base si evolve in maniera definitiva andando a decidere l'esatto tipo di sistema (client-server, distribuito, ecc.).

### 9.1 Architettura di sistema.

Per quanto riguarda l'architettura, come già anticipato sarà un'architettura di tipo client-server con la possibilità di più connessioni client che avvengono contemporaneamente. Per quanto concerne l'infrastruttura di rete abbiamo deciso di utilizzare la rete internet. Il ruolo centrale del server sarà quello di creare diversi thread per ogni connessione, in modo da poter gestire più richieste in modo contemporaneo, di regolare l'accesso e la registrazione di ogni diverso utente. Il ruolo del client invece sarà quello di effettuare i calcoli necessari per ottenere la massa grassa, la massa magra, le misurazioni massime muscolari e infine la scheda personalizzata , tutti i calcoli verranno fatti in modo trasparente l'utente visualizzerà solamente i risultati, la trasparenza si rifletterà anche sull'algoritmo da noi progettato e utilizzato nel sistema. Anche dal punto di vista dell'efficienza, verrà così astratta dal tipo di computer utilizzato dal cliente.

Utilizzare questo tipo di architettura presenta dei vantaggi e degli svantaggi ne andremo ad analizzare alcuni:

#### Vantaggi:

- Ogni cliente ha la possibilità di accedere tramite l'interfaccia desktop, eliminando la necessità di accedere in modalità terminale o processore.
- I server hanno un migliore controllo dell'accesso e delle risorse per garantire che solo i client autorizzati possano accedere o manipolare i dati e gli aggiornamenti del server siano amministrati in modo efficace.
- Possibilità per l'utente di accedere a *Gym for you* in ogni momento e da diversi dispositivi (pc, tablet, smartphone).

#### Svantaggi:

- Quando il server sarà implementato, funzionerà ininterrottamente. Il che significa che deve ricevere la giusta manutenzione. In caso di

problemi, è necessario risolverli immediatamente senza alcun ritardo. Quindi, dovrebbe esserci un gestore di rete specializzato nominato per mantenere il server.

- Nel caso in cui il server principale subisca un guasto o un'interferenza, l'intera rete verrà interrotta. Pertanto, le reti client server mancano di robustezza.
- Lo svantaggio principale della rete client-server è la congestione del traffico a cui è soggetta. Se troppi client effettuano richieste dallo stesso server, si verificheranno arresti anomali o rallentamenti della connessione. Un server sovraccarico crea molti problemi nell'accesso alle informazioni.

Dopo aver valutato le caratteristiche positive e negative, si è giunti alla conclusione che l'utilizzo di un tipo di architettura client-server possa fornire un valore aggiunto al software, e si è dunque confermato l'avvio dello sviluppo con questa metodologia. Quest'architettura da sviluppare sarà divisa in due parti, la parte del server e la parte del client che dovranno interagire tra di loro come vedremo in seguito.

## 9.2 Server.

Il server sarà strutturato come un normalissimo server web: attenderà una richiesta di connessione da parte di un client sulla propria porta adibita al collegamento (utilizzerà come porta di default la 8085) e, all'arrivo di una nuova richiesta di connessione, creerà una nuova porta di collegamento con il singolo client che verrà utilizzata per tutti i futuri messaggi. La porta indicata rimarrà in ascolto fin quando l'utente sarà collegato, nel momento del log-out essa cesserà di funzionare. L'attesa del thread client non sarà attiva, essendo le chiamate ai metodi per la lettura di oggetti bloccanti, tuttavia nel caso di un gran numero di utenti collegati in contemporanea potrebbero sorgere problemi relativi alla terminazione dello spazio disponibile per i vari thread. Altro compito importante che sarà affidato al server riguarda l'invio dei dati che dovranno essere poi salvati nel nostro database. Per fare ciò il client invierà, tramite il protocollo, i dati al server quest'ultimo li passerà al database per essere poi salvati nella tabella corrispondente. Altro compito del server sarà quello di popolare i file XML con gli input inseriti dall'utente.

## 9.3 Client.

Anche il client avrà un comportamento molto simile ad un qualsiasi client-web. Il suo compito sarà quello di raccogliere tutte le richieste provenienti dall'utente, inviare tali richieste e sarà poi il server ad esudire le richieste

del client e quindi ritornerà un messaggio contenente il risultato della richiesta stessa. Il client sarà colui che elaborerà i dati provenienti dal server per restituire un risultato all'utente, tra i risultati possiamo avere la scheda personalizzata creata dal nostro algoritmo oppure il valore della massa magra e grassa ecc. Utilizzando per la memorizzazione di dati non solo il database ma anche i file XML, altro compito del client sarà quello di inserire dei valori di default all'interno di questi file nel caso in cui essi non siano stati già creati.

## 9.4 Protocollo di comunicazione.

La scelta sul protocollo di comunicazione è ricaduta sul protocollo HTTP. Tale protocollo infatti sta alla base sull'architettura che abbiamo deciso di utilizzare. HTTP prevede che il client faccia la richiesta e il server ha il compito di dare una risposta, infatti se la richiesta e i dati che sono stati inviati sono errati il server risponderà al client con un messaggio di errore. Per via della natura sincrona di HTTP il client deve sempre attendere la risposta del server e allo stesso tempo anche l'utente si troverà sempre ad attendere la riposta del server. Per il trasporto dei dati siamo sicuri che essi arrivino a destinazione in quanto all'interno di HTTP è presente il protocollo di trasporto TCP/IP che garantisce che i dati arrivino a destinazione senza nessuna manomissione o modifica.

## 9.5 Configurazione.

Per fare la connessione client-> server -> database, si sono dovuti inserire diversi parametri tra cui:

### Connessione server:

```
<?xml version="1.0" encoding="UTF-8"?>
- <IMPOSTAZIONI>
  <PORT_DATABASE>3306</PORT_DATABASE>
  <KEYSTORE_PASSWORD>pass</KEYSTORE_PASSWORD>
  <PATH_KEYSTORE>C:\Users\HP250G3
    \Documents\NetBeansProjects\GymForYou_Server\sparkserver.jks</PATH_KEYSTORE>
  <PORT_SERVER>8085</PORT_SERVER>
  <USER_DATABASE>root</USER_DATABASE>
  <SERVER_DATABASE>localhost</SERVER_DATABASE>
  <PASS_DATABASE/>
  <NAME_DATABASE>gymforyou</NAME_DATABASE>
</IMPOSTAZIONI>
```

Figura 9.0 Configurazione server.

## Connessione client:

```
<?xml version="1.0" encoding="UTF-8"?>
- <IMPOSTAZIONI>
  <PROTOCOL>http</PROTOCOL>
  <SERVER_PORTA>8085</SERVER_PORTA>
  <SERVER_ADDRES>localhost</SERVER_ADDRES>
</IMPOSTAZIONI>
```

Figura 9.1 Configurazione client.

Nel nostro sistema si è inoltre deciso di salvare i nostri dati esternamente all'interno di file XML, questo ci permette di prelevare e modificare dati a nostro piacimento senza andare a effettuare troppe chiamate al database. Questi file si troveranno all'interno della stessa directory in cui è presente il codice sorgente. Nel caso in cui i file siano vuoti il client inserirà dei valori di default. Invece il server richiederà all'utente di inserire tali valori e creerà un nuovo file XML.

## 9.6 Database (db).

Il team di sviluppo dopo avere analizzato i requisiti e dopo aver deciso di sfruttare per il proprio sistema un tipo di architettura client-server ha analizzato quali potevano essere le migliori soluzioni per la gestione dei dati e la scelta è ricaduta su un DBMS. Tale scelta è stata fatta in quanto viene migliorata l'organizzazione dal punto di vista strutturale, migliora la velocità e la fluidità gestionale del software. In particolare va definito che l'uso di un database deve essere necessariamente di supporto. Il nostro sistema si prospetta che sarà utilizzato da molti utenti, appunto per questo per gestire questa situazione ci siamo serviti dell'uso di un database relazionale. Il compito di tale database sarà di gestire tutti gli utenti che utilizzeranno *Gym for You*. Nel nostro database saranno salvati tutti i valori relativi alla composizione corporea e muscolare del nostro utente, al suo interno troveremo anche tutte le informazioni utili alla registrazione e all'accesso alla nostra piattaforma, non solo degli utenti ma anche degli altri attori che compongono il nostro sistema. Grazie alle misurazioni muscolari e alla composizione fisica salvate nel db, il nostro utente potrà tenere traccia quando e dove vuole, quindi anche su diverse piattaforme, di dati come altezza, peso, massa grassa, ecc, ed anche dei suoi progressi ottenuti dall'allenamento fatto tramite il nostro sistema. Il database verrà

anche utilizzato per immagazzinare le risposte date dall'utente al questionario, in questo modo il medico potrà accedere alla piattaforma quando vuole, visualizzare le risposte e valutare il questionario. Il database permetterà così al medico di dare l'esito al questionario quando egli vuole, riducendo i tempi di attesa dell'utente.

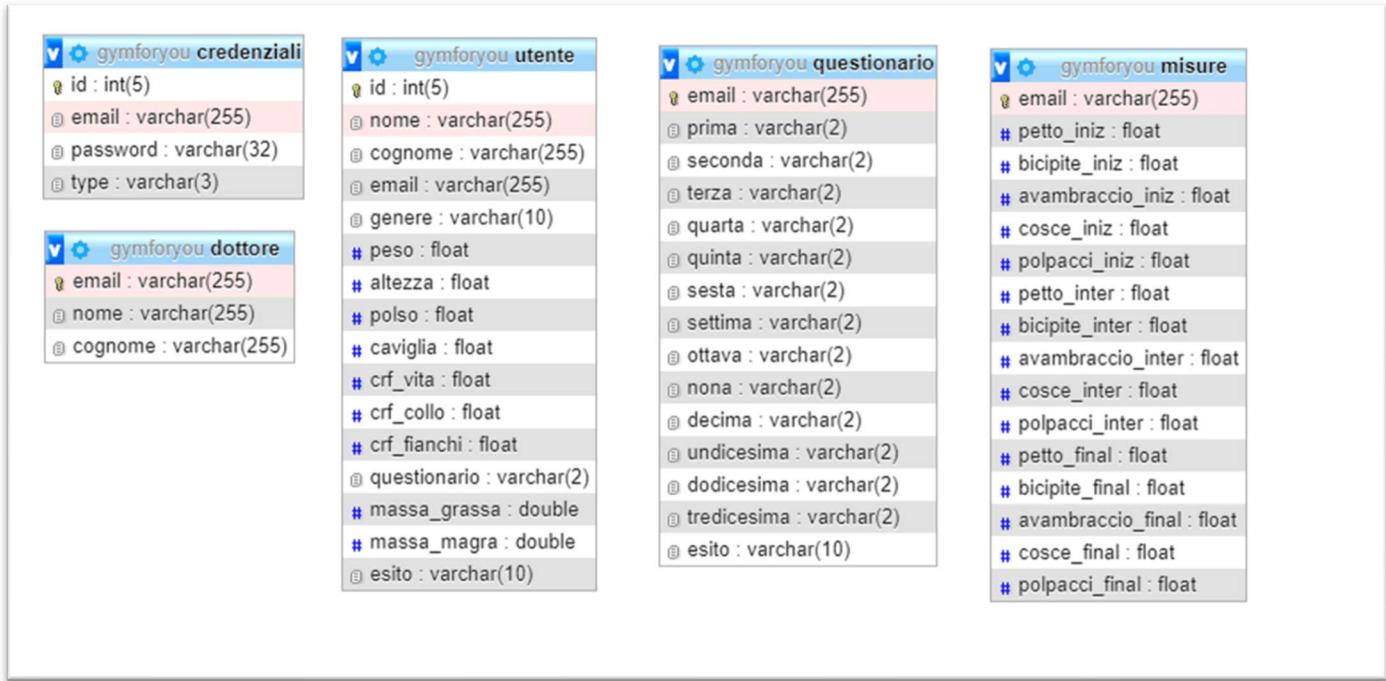


Figura 9.2 Tabelle database.

Le operazioni che possono essere effettuate sul nostro database sono le seguenti:

- 1) Login;
- 2) Registrazione utenti;
- 3) Prelevamento, memorizzazione, modifica ed eliminazione dei dati relativi all'utente;
- 4) Prelevamento delle risposte del questionario e memorizzazione esito da parte del medico;
- 5) Prelevamento, memorizzazione e modifica dei dati relativi ai dipendenti (medico). Modifica dati personali relativi all'admin.

Tutte le operazioni sopra indicate sono presenti all'interno della classe database, contenuta nel progetto **GymForYou\_server**.

## 9.7 Diagrammi degli stati.

Allo stesso modo dell'attività di analisi, durante la fase di progettazione si riprendono i diagrammi di interazione per entrare nelle specifiche dei casi d'uso il focus in questo caso, non è il cosa, ma il come. Entrano in scena tutte le classi di utilità e di contorno e specificano, per ogni caso d'uso, tutti i passi che deve effettuare il sistema per risolverlo. L'UML viene incontro alla rappresentazione degli stati di una classe attraverso il diagramma degli stati.

### Diagramma degli stati: gestione accesso.

Se un utente vuole usufruire dei servizi di *Gym for You*, per prima cosa dovrà effettuare la registrazione. Una volta effettuata la registrazione l'utente verrà rimandato nella pagina di log-in, se immetterà le giuste credenziali potrà accedere al proprio pannello. Per quanto riguarda il login, bisognerà inserire le proprie credenziali, dopodiché verrà fatto un controllo sull'email, in base al tipo di utente che sta effettuando l'accesso verrà reindirizzato al proprio pannello. All'interno di ogni pannello sarà presente un tasto log-out che permetterà di uscire dalla piattaforma e tornare alla pagina di log-in.

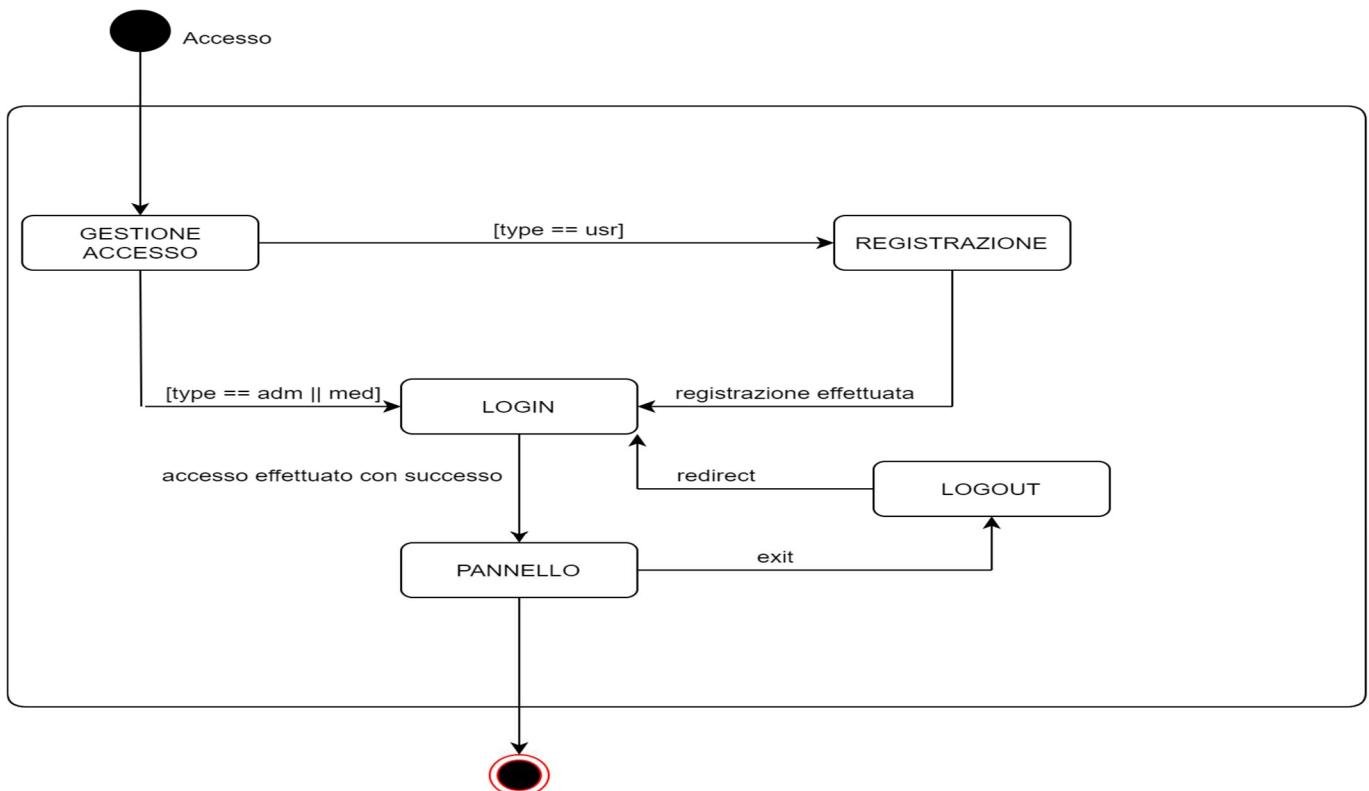


Figura 9.0 Diagramma degli stati gestione accesso.

## Diagramma degli stati: servizi senza il consenso del medico.

Nel diagramma (figura 9.2) mostriamo cosa può fare l'utente nel caso in cui abbia compilato il questionario ed è in attesa dell'esito, oppure ha ricevuto un esito negativo. Può usufruire del servizio che gli permette di conoscere il quantitativo di massa magra e grassa presenti nel suo corpo. Per ottenere i valori delle masse deve inserire le circonferenze indicate, dopodichè verranno prelevati l'altezza e il peso inseriti nel momento della registrazione e fatti i calcoli. Inoltre l'utente può modificare quando vuole anche più di una volta le circonferenze, l'altezza e/o il peso e infine la propria password all'interno dell'apposita gestione account.

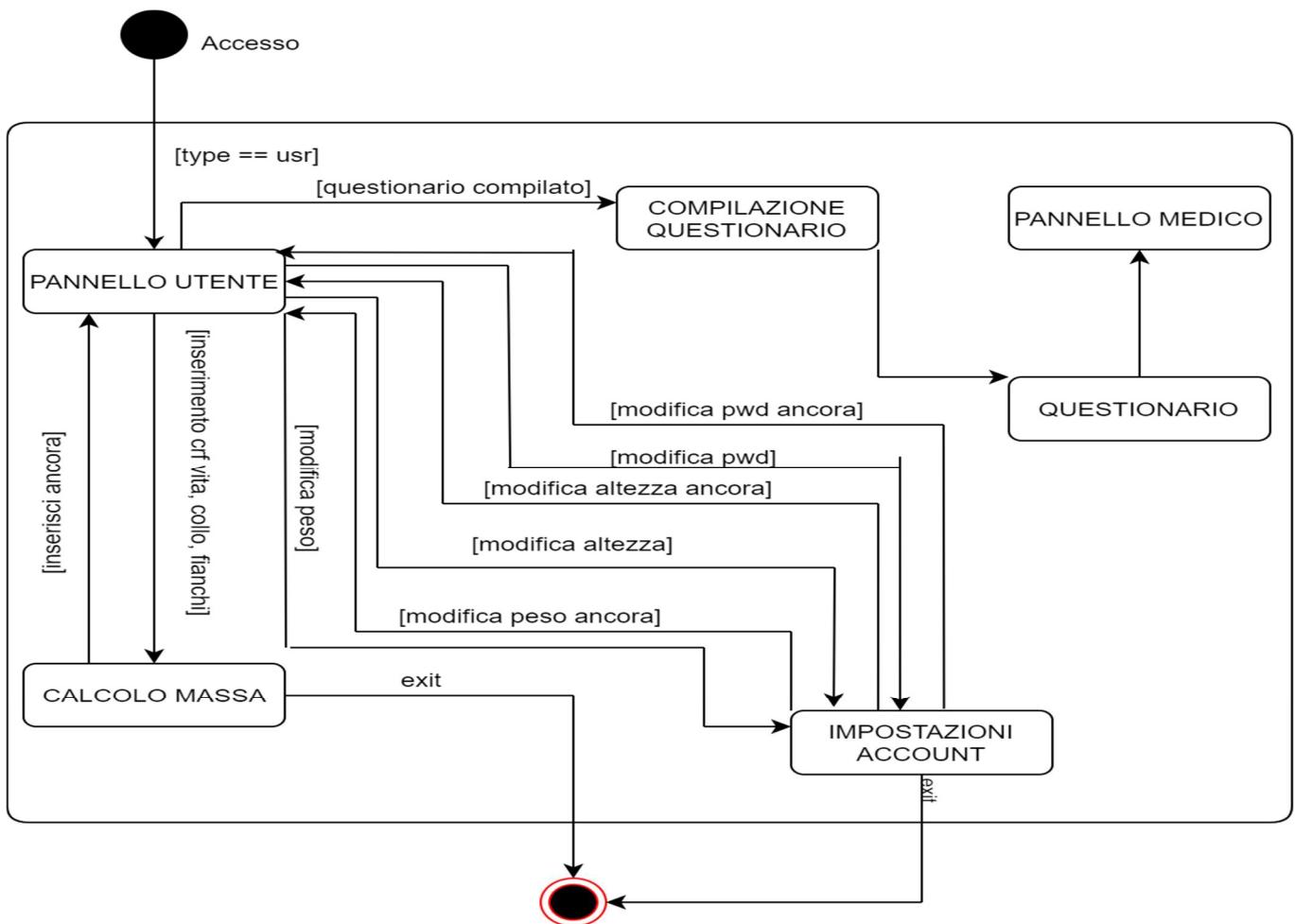


Figura 9.1 Diagramma degli stati Servizi senza il consenso del medico.

## Diagramma degli stati: gestione salute utenti.

Di seguito (figura 9.2) riportiamo come si interfaccia il medico con il nostro sistema. Per prima cosa deve fare il login, in base al type restituito verrà reindirizzato nel suo pannello. All'interno del proprio pannello il medico potrà visualizzare tutti gli utenti che hanno compilato il questionario. In questo modo il medico potrà valutare lo stato di salute dell'utente e dare un esito negativo o positivo in base alle risposte date dall'utente.

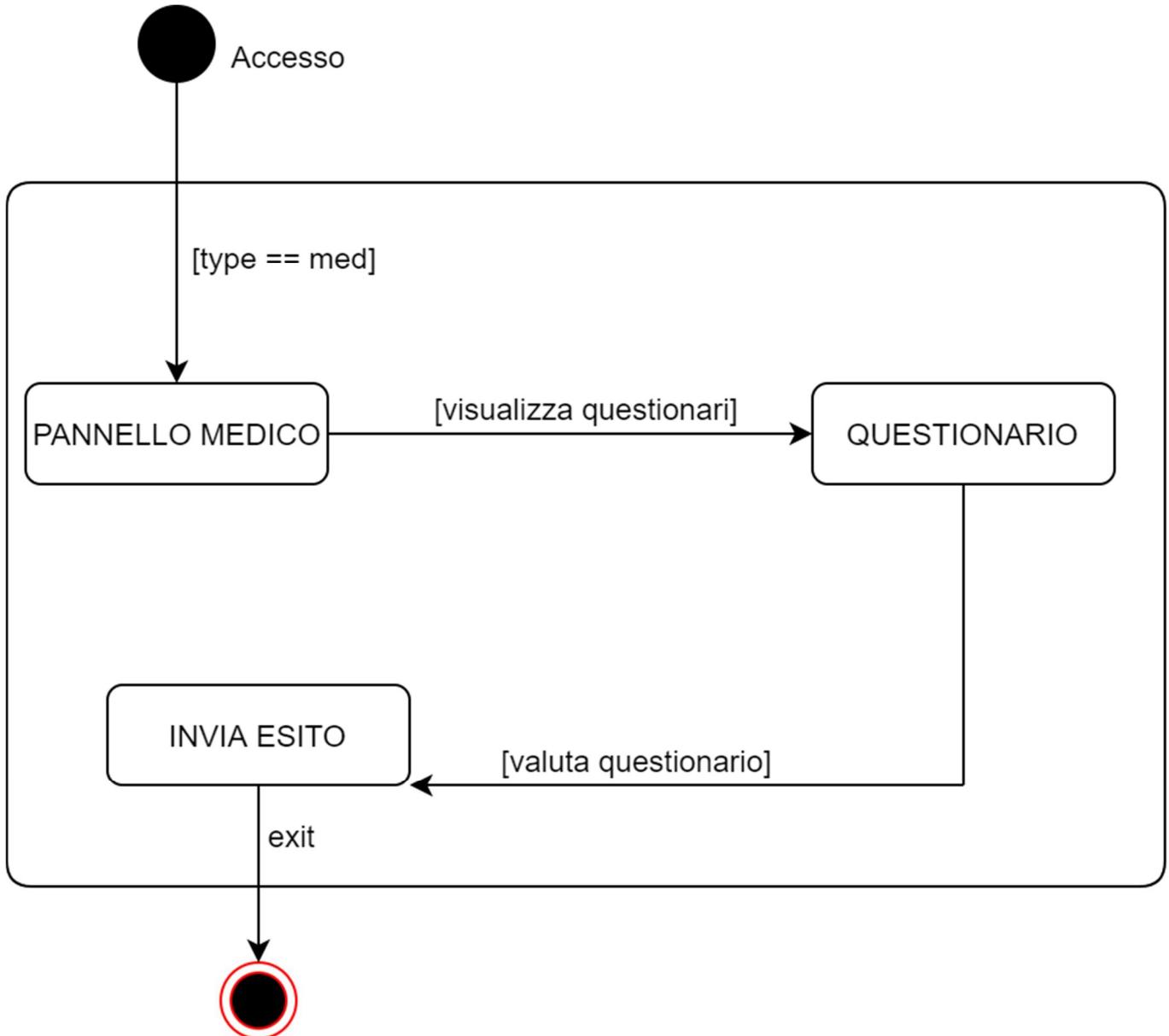


Figura 9.2 Diagramma degli stati gestione salute utenti.

## Diagramma degli stati: servizi con il consenso del medico.

Il nostro utente naturalmente dopo aver effettuato l'accesso, si troverà all'interno del proprio pannello, ma questa volta però avrà il "via libera" del medico. I servizi che potrà utilizzare come descritto nel diagramma (figura xx) riguardano inanzitutto la composizione muscolare dell'utente, infatti egli è chiamato ad inserire le circonferenze indicate, invece la massa grassa verrà prelevata in modo automatico. Il sistema sfruttando l'algoritmo da noi creato andrà a calcolare il peso corporeo, il peso corporeo ammassato massimo e le massime misurazioni muscolari, quest'ultime indicano quanto possono crescere al massimo i muscoli del nostro utente. Dopodichè i valori relativi ai muscoli verranno inviati a *scheda for you* questo strumento permetterà all'utente di inserire le misure attuali dei propri muscoli. Il nostro algoritmo farà un confronto tra le misure attuali e quelli finali e capirà a quale parte del corpo bisognerà dare la priorità nell'allenamento, di conseguenza verranno forniti all'utente allenamenti mirati a far crescere quei muscoli che sono indietro rispetto agli altri. Oltre ad essere inserite, le misure attuali potranno naturalmente essere modificate in qualsiasi momento, in questo modo le modifiche ai muscoli si rifletteranno anche sugli allenamenti che verranno assegnati in maniera diversa.

**N.B** Ci teniamo a precisare che l'utente può anche utilizzare tutti quei servizi che sono stati descritti nel diagramma in figura 9.1, abbiamo decisi di non metterli nel seguente diagramma per non risultare ripetitivi.

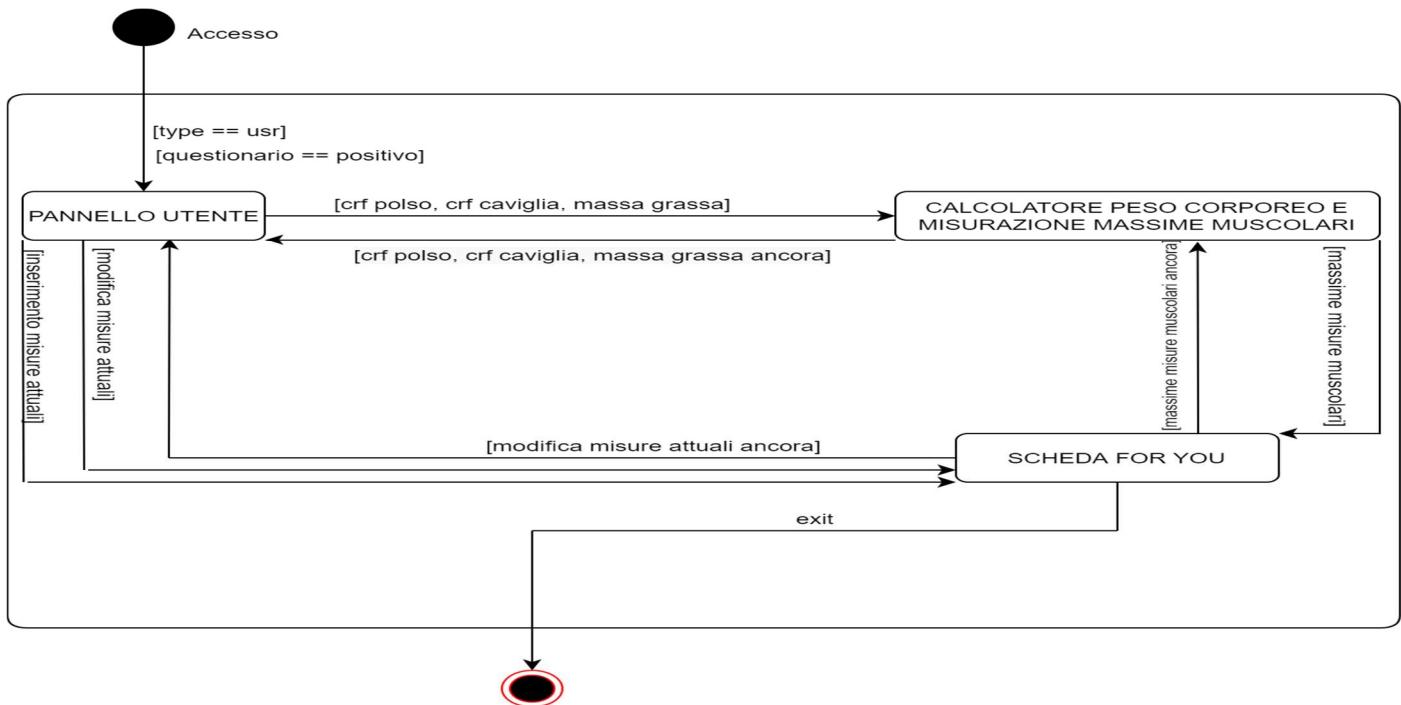


Figura 9.3 Diagramma degli stati servizi con il consenso del medico.

## Diagramma degli stati: gestione sistema.

L'admin come tutti gli altri attori del nostro sistema verrà riconosciuto in base alla sua email, e poi reindirizzato all'interno del proprio pannello. All'interno della propria pagina l'admi può inserire un nuovo dipendente compilando i campi appositi. Davanti a se l'admin si troverà una lista di tutti i suoi dipendenti, cliccando su uno di essi può scegliere se eliminarlo dal sistema oppure cambiare la password di qualcuno di essi nel caso in cui venga smarrita. Inoltre l'admi può accedere al proprio profilo personale e modificare la propria password.

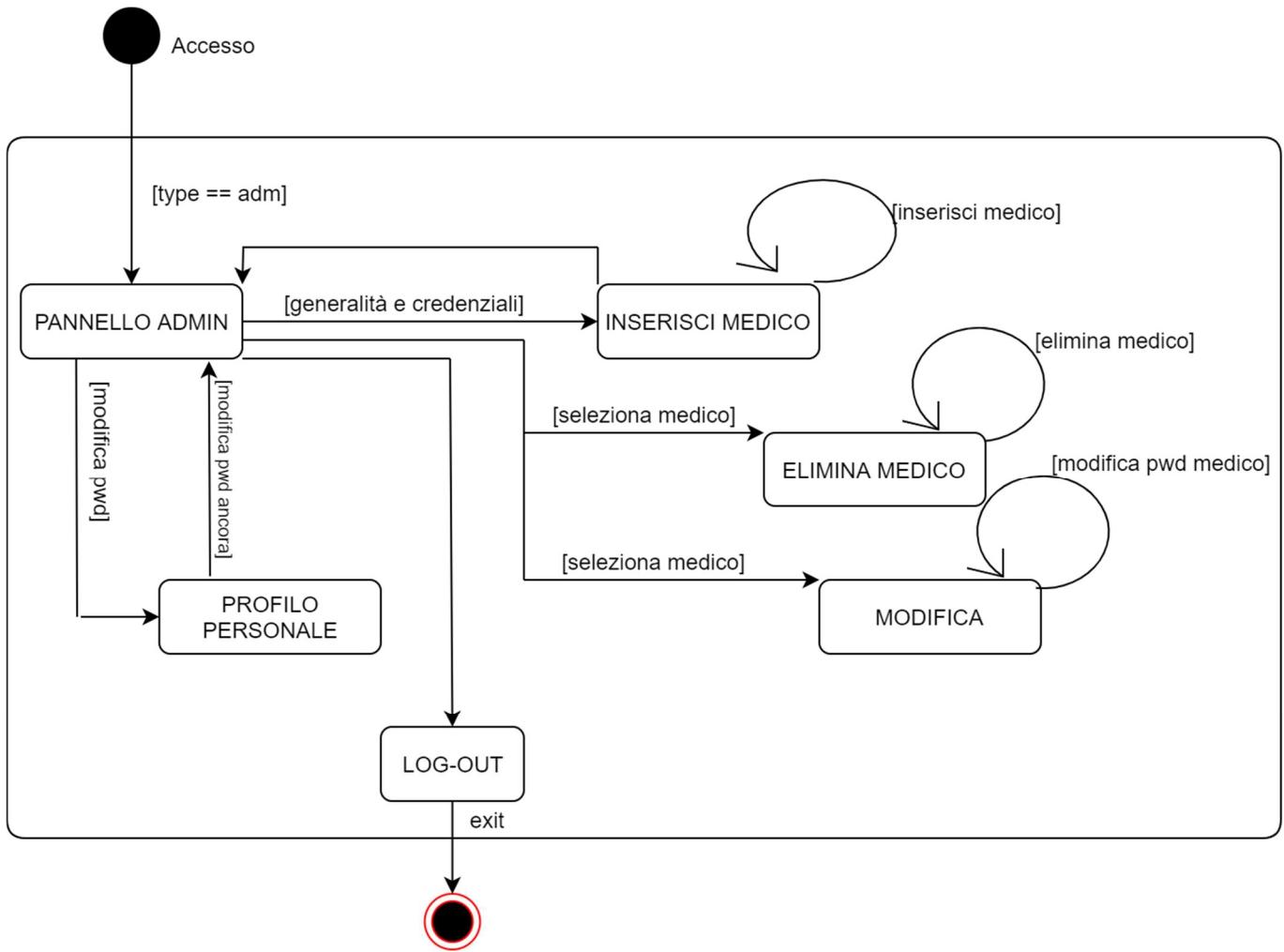


Figura 9.4 Diagramma degli stati gestione sistema.

## 10. Descrizione delle GUI.

In questo paragrafo andremo a descrivere le interfacce che sono state realizzate per far interagire gli stakeholders con il nostro sistema. Per la realizzazione delle interfacce si è andati alla ricerca della semplicità e dell’usabilità. La struttura delle GUI si basa su un insieme di frame, al cui interno, tramite dei “button” ci sposteremo tra di essi. La struttura è abbastanza semplice in quanto ogni frame viene richiamato solo quando un evento voluto dall’utente lo farà comparire, gli eventi scaturiscono nel momento in cui viene cliccato un button. Per la realizzazione delle GUI ci siamo affidati al framework messo a disposizione dall’IDE Netbeans denominato Swing. Ora per una maggiore comprensione delle interfacce le andremo a descrivere ognuna di esse singolarmente.

### Frame login.

Questo frame (figura 10) verrà utilizzato da utente, medico e admin per accedere ai propri pannelli personali. L’interfaccia di login si compone di due *textField* che dovranno essere compilati con le credenziali di chi sta accedendo. Nel momento in cui si clicca il *button* “accedi” verranno eseguite le funzioni che avranno il compito di controllare che le credenziali siano corrette, nel caso in cui lo fossero in base al *type* allegato all’email inserita il nostro stakeholder verrà reindirizzato nel proprio pannello. Altrimenti apparirà un messaggio di errore. In questo frame troviamo un secondo bottone con su scritto “registrai” che verrà descritto nel paragrafo successivo.

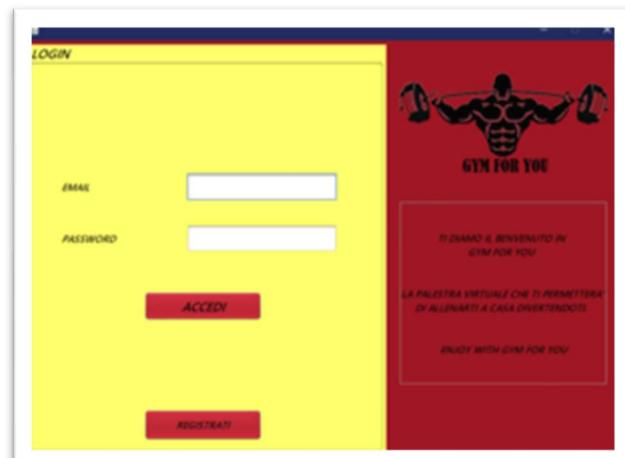


Figura 10 Frame login.

### Frame registrazione.

Questo frame (figura 10.1) viene utilizzato solamente dagli utenti quando vogliono registrarsi al nostro sistema. Tale frame si compone di diversi campi che dovranno essere compilati. Nel momento in cui l’utente clicca il bottone verranno attivate delle funzioni che

The screenshot shows a registration form titled 'REGISTRAZIONE'. It is divided into two main sections: 'CREDITIZIALI' (left) and 'FISICO' (right). The 'CREDITIZIALI' section contains fields for 'NOME', 'COGNOME', 'GENERE' (with radio buttons for Uomo, Donna, and Altro), 'EMAIL', 'PASSWORD', and 'CONFERMA PASSWORD'. The 'FISICO' section contains fields for 'ALTEZZA (CM)' and 'PESO (KG)'. At the bottom is a large red 'REGISTRATI' button.

Figura 10.1 Registrazione

controlleranno se gli input sono corretti prestando attenzione soprattutto sulle credenziali, infatti se l'email inserita è già utilizzata da qualche altro utente, oppure se le password inserite non rispettano la lunghezza minima (tra 8 e 15 caratteri) oppure non coincidono fra di loro, in tutti questi tre casi apparirà un alert e la registrazione non potrà essere effettuata.

### Pannello Utente.

Questo pannello (figura 10.2) si compone di diversi bottoni disposti su due file in maniera verticale. Il primo bottone della fila di sinistra ci porterà in un altro frame (figura 10.3), all'interno del quale troveremo dei *textField* che dovranno essere compilati dall'utente con le informazioni richieste. In basso troviamo due bottoni che fanno due operazioni diverse il primo "calcola massa grassa", andrà a prelevare altezza e peso inseriti nel momento della registrazione, dopodiché utilizzando i dati inseriti nei *textField* calcolerà la massa grassa dell'utente e la farà apparire mediante la funzione *setVisible*. La stessa cosa succederà con il bottone "calcola massa magra".

L'utente utilizzando l'apposito *button* potrà tornare indietro al pannello principale. Premendo sul bottone "compila questionario" apparirà un nuovo frame (figura 10.4), qui troveremo 13 domande a cui l'utente dovrà rispondere. Per le risposte (si o no) abbiamo utilizzato dei *radioButton* e per fare in modo che venga selezionata una sola opzione tra quelle presenti ci siamo serviti di un *buttonGroup*. Alla fine delle domande ci sarà un *button* "invia" che invierà le risposte. Ritornando al pannello principale l'utente non troverà più il bottone descritto in precedenza ma ci sarà una *label* che indicherà lo stato del questionario.

Se il questionario risulta:

- **in attesa oppure negativo:** l'utente potrà utilizzare solamente il servizio di calcolo delle masse. Abbiamo fatto in modo che fin quando l'esito non risulterà positivo l'utente non potrà usare i servizi che richiedono questo tipo di esito, per questo nel caso in cui verranno premuti gli appositi bottoni apparirà un alert;
- **Positivo:** l'utente potrà utilizzare gli altri servizi messi a sua disposizione. Tra cui "calcolatore peso corporeo e misurazioni massime muscolari" e "scheda for you".

Il secondo bottone di sinistra recita "calcolatore peso corporeo e misurazioni massime muscolari". Premendo su questo bottone apparirà un nuovo frame (figura 10.5), qui l'utente dovrà inserire negli appositi text field le informazioni richieste, per quanto riguarda il campo "massa grassa" esso verrà automaticamente compilato se l'utente ha già effettuato il calcolo di tale

grandezza. Quando l'utente premerà sul bottone “calcola” tramite le funzioni apposite, verranno calcolate delle grandezze che potranno essere visualizzate in un frame che apparirà (figura 10.6). Nel pannello utente troviamo un altro bottone con su scritto “scheda for you” che ci rimanda ad un nuovo frame (figura 10.7). Per poter accedere al frame in questione per prima cosa l'utente dovrà avere calcolato le misurazioni massime muscolari, se ciò non è stato fatto apparirà un alert. All'interno dell'interfaccia “scheda for you” l'utente troverà sulla sinistra dei campi in cui dovranno essere inserite le misure attuali espresse in centimenti dei muscoli in questione, sulla destra abbiamo i valori delle massime misure muscolari che indicano, in base alla composizione fisica del nostro utente quanto egli può accrescere i suoi muscoli, al centro troviamo le *progressBar* esse indicano quanto è stato fatto dall'utente per arrivare al massimo. Alla fine del frame sarà presente un bottone che una volta cliccato andrà ad assegnare degli allenamenti specifici per le parti del corpo che dovranno essere aumentate, creando così una scheda personalizzata (figura 10.8). Nel pannello utente troviamo un altro bottone con scritto “impostazioni account”, premendolo ci porterà in un nuovo frame (figura 10.9), all'interno l'utente può modificare la sua altezza e/o il peso (figura 10.10), l'utente invece premendo su “modifica password” potrà modificare la password del suo account (figura 10.11).

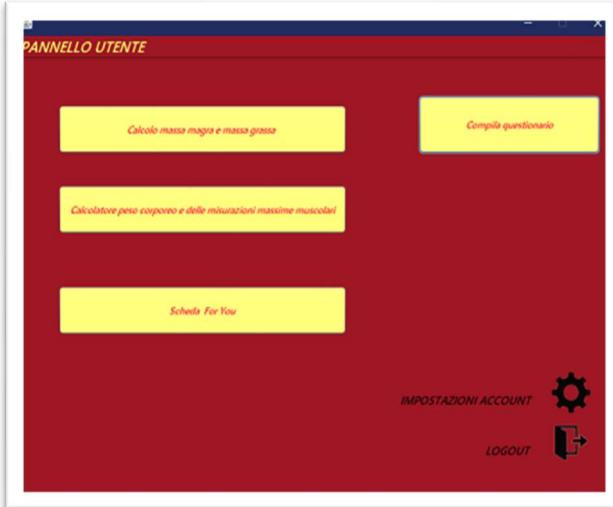


Figura 10.2 Pannello utente.

**CALCOLO MASSA GRASSA E MASSA MAGRA**

Qui puoi usufruire di un calcolatore che ti permette di calcolare la massa grassa e la massa magra del tuo corpo.

La massa grassa è importante per capire in che stato si trova il tuo corpo. Per questo una volta ottenuto il risultato puoi confrontarlo con la tabella che mettiamo a tua disposizione e comprendere lo stato del tuo corpo.

**Massa grassa**

Circonferenza vita:

Circonferenza fianchi:

Circonferenza collo:

**Calcola massa grassa**

**Massa magra**

ALTEZZA: 175.0 CM  
PESO: 68.0 KG

**Calcola massa magra**

VALUTAZIONE	UOMINI	DONNE
Peso minore	25-40%	10%-12%
Norma atletica	60-70%	141-150%
Buono stato di fitness	14%-17%	21%-24%
Sopra della media	18%-25%	29%-31%
Obesità	>= di 26%	>= del 32%

Figura 10.3 Frame calcolo masse.

**QUESTIONARIO UTENTE**

**Domanda 1**  
Provvi dolore al torace quando fai attività fisica?

Si       No

**Domanda 2**  
Provvi vertigini o giramenti di testa prima o dopo aver fatto attività fisica?

Si       No

**Domanda 3**  
Hai problemi alle ossa o alle articolazioni (per esempio schiena, anche ginocchia) che potrebbero peggiorare a causa dell'attività fisica?

Si       No

**Domanda 4**  
Hai mai avuto perdita dei sensi o quasi perdita dei sensi durante o dopo l'esercizio fisico?

Si       No

**Domanda 5**  
Hai mai avuto problemi respiratori durante o dopo lo sforzo fisico

Si       No

Figura 10.4 interfaccia compilazione questionario.

**PERFEZIONA CORPO**

Il seguente calcolatore fornisce una stima delle misurazioni muscolari che è probabile che il tuo corpo possa raggiungere.  
Si basa su delle equazioni che sono state studiate e sviluppate da una ricerca che è durata circa sei anni.

Altezza: 180.0 CM

Circonferenza polso:

Circonferenza caviglia:

Massa Grassa: 0.0 %

**CALCOLA**

**Informazioni utile**

- 1) Tutte le misurazioni devono essere espresse solo ed esclusivamente in centimetri.
- 2) Per misurare la circonferenza del polso, innervis di un metro da sartà, tenere l'avambraccio flesso ad angolo retto e il palmo della mano rivolto verso l'alto, mettere il metro da sartà sotto il radio e l'ulna e misurare.
- 3) Per misurare la circonferenza della caviglia, porre il metro da sartà nella parte più stretta della caviglia e misurare.
- 4) Per ottenere il valore della tua massa grassa ti rimandiamo al calcolatore dedicato presente nella tua dashboard.

Figura 10.5 Frame perfeziona corpo.



Figura 10.6 Risultato massime misure muscolari.

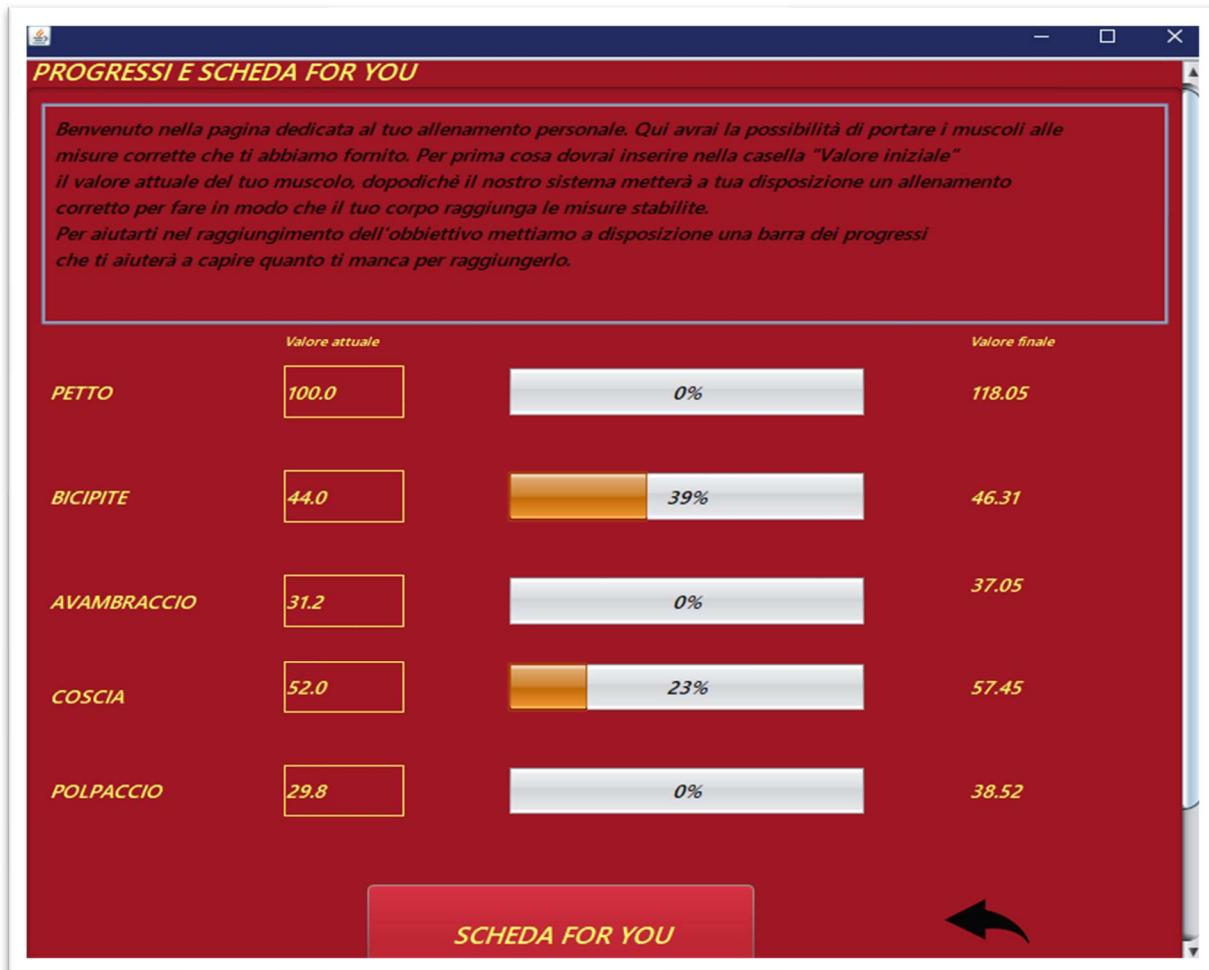


Figura 10.7 Interfaccia scheda for you.



Figura 10.8 Scheda for you.



Figura 10.9 Impostazioni account.

Figura 10.10 Modifica password utente.

Figura 10.11 Modifica peso o altezza.

## Pannello Medico.

Dopo aver fatto il login il nostro medico entrerà nel proprio pannello (figura 10.12), cliccando su “controlla questionari” verrà rimandato ad un altro frame. In questo frame (figura 10.13) sulla sinistra troverà una *jTable* all’interno della quale ci sarà l’elenco di tutti gli utenti che hanno compilato il questionario. Il medico cliccando su una delle righe della tabella scoprirà le risposte date dall’utente che ha selezionato, infatti esse appariranno nelle apposite *label*. Il medico avrà a sua disposizione due bottoni che permetteranno di dare un esito al questionario. Per far in modo che il medico possa leggere le domande che vengono sottoposte all’utente abbiamo predisposto un frame che comparirà nel momento in cui verrà cliccato il bottone “controlla questionari”.

Figura 10.12 Pannello medico.

	NO	EMAIL
DOMANDA 1	NO	mariorossi@gmail.com
DOMANDA 2	SI	
DOMANDA 3	NO	
DOMANDA 4	NO	
DOMANDA 5	SI	
DOMANDA 6	NO	
DOMANDA 7	NO	
DOMANDA 8	NO	
DOMANDA 9	NO	
DOMANDA 10	SI	
DOMANDA 11	NO	
DOMANDA 12	NO	
DOMANDA 13	NO	

Figura 10.13 Frame controllo questionari.

## Pannello admin.

Nel pannello dedicato all'admin (figura 10.14), abbiamo predisposto dei *textfield* che devono essere utilizzati per poter inserire un nuovo dipendente all'interno della piattaforma. Per fare l'inserimento inanzitutto devono essere compilati i campi con gli input corretti e poi premere il bottone "inserisci", se gli input inseriti non saranno corretti apparirà un alert. Sulla destra l'admin troverà davanti a se una *jTable* con tutti i dipendenti presenti nella piattaforma, cliccando sulla riga della tabella potrà eliminare il dipendente selezionato premendo il bottone "elimina". Se invece vuole modificare la password di un dipendente l'admin dovrà sceglierlo dalla tabella, in seguito dovrà compilare i campi relativi alla password se verranno compilati in modo corretto apparirà un messaggio che lo informerà dell'avvenuta modifica. Infine l'admin può anche modificare la propria password cliccando sul bottone "impostazioni account" che lo proietterà in un nuovo frame (figura 10.15) che gli permetterà di fare l'operazione di modifica della propria password.

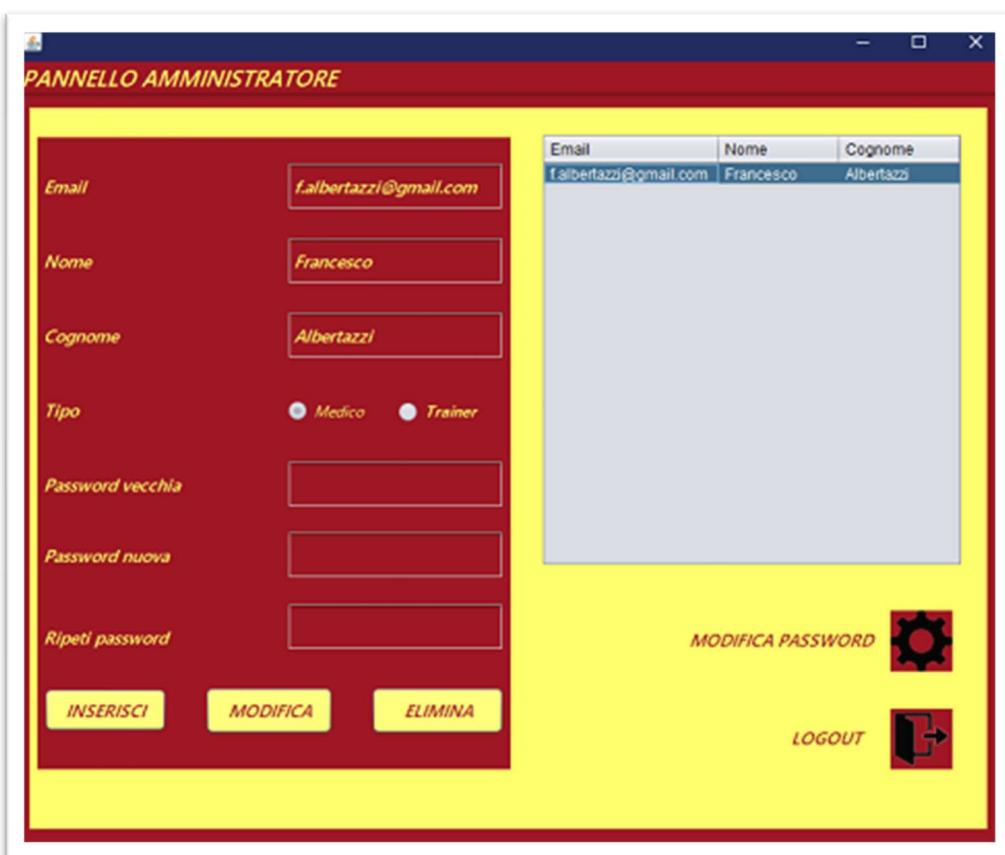


Figura 10.14 Pannello admin.

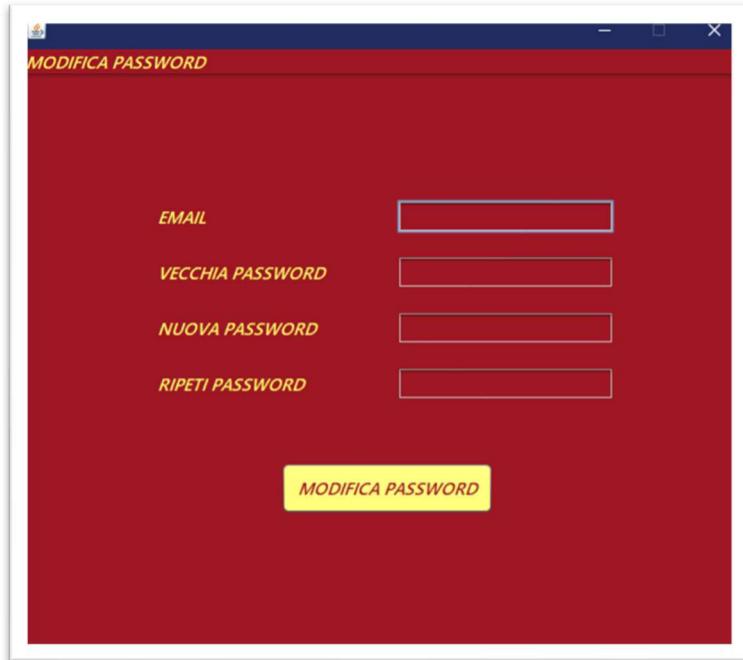


Figura 10.15 Modifica password admin.

## 10.1 Descrizione algoritmo.

Il compito del nostro algoritmo è fare in modo che l'utente possa allenarsi senza arrecare danni ai propri muscoli ed anche fare in modo che i muscoli del suo corpo abbiano una crescita naturale e controllata.

### Presupposti.

- Ogni utente ha la sua altezza e il suo peso;
- Ogni utente ha la propria struttura fisica;
- Ogni utente ha la propria composizione muscolare;
- Ogni allenamento viene fornito in modo diverso in base all'utente;
- Ogni muscolo ha propri esercizi;
- Ogni utente ha diversi tempi di crescita muscolari.

### Scopo.

Lo scopo del nostro algoritmo è quello di far accrescere in modo controllato i muscoli dei nostri utenti, far capire all'utente in che condizioni si trova il suo fisico fornendo le informazioni relative alla massa grassa, magra ecc.

### Criterio.

Il criterio utilizzato si basa sull'inserimento di determinate informazioni che serviranno da input per la creazione della scheda personalizzata. Il software analizzerà il peso, l'altezza, la massa grassa, la massa magra e tutti gli altri valori ottenuti dai calcoli per creare una scheda adatta all'utente.

## Calcolo massa grassa e massa magra.

Per prima cosa il nostro algoritmo va a prelevare i valori del peso e dell'altezza inseriti dall'utente nel momento della registrazione. Poi verranno prelevati i valori delle circonferenze richieste. I calcoli cambiano in base al genere del nostro utente perciò avremmo:

Massa grassa UOMO

$$495/(1.0324-0.19077*\log(crf_vita-crf_collo)+0.15456*\log(altezza))-450$$

Massa grassa DONNA

$$495/(1.29579-0.35004*\log(crf_vita+crf_fianchi-crf_collo)+0.22100*\log(altezza))-450$$

Massa magra UOMO

$$(1.10*peso)-128*(peso^2/altezza^2)$$

Massa magra DONNA

$$(1.07*peso)-148*(peso^2/altezza^2)$$

## Calcolo pesi.

Per calcolare i diversi pesi utilizziamo le circonferenze richieste e preleviamo i valori di altezza e massa grassa. In questo caso i calcoli vanno bene per qualsiasi genere. I calcoli che facciamo sono i seguenti:

Peso corporeo muscolare massimo:

$$(altezza^{1.5})*(\frac{\sqrt{polso}}{22.6670} + \frac{\sqrt{caviglia}}{17.0104})*(\frac{massagrassa}{224} + 1)$$

Peso corporeo:

$$\left( \frac{pesomuscolaremax}{100 - massagrassa} \right) * 100$$

Peso corporeo ammassato massimo:

$$pesocorporeo * 1.04$$

## Massime misure muscolari.

Per ottenere le massime misurazioni muscolari preleviamo le circonferenze richieste ed utilizziamo la massa grassa calcolata in precedenza e l'altezza inserita nel momento della registrazione. Qui non abbiamo differenze di calcoli dovuti al genere.

Massimo petto:

$$(1.6817 * polso) + (1.3759 * caviglia) + (0.3314 * altezza)$$

Massimo bicipite:

$$(1.2033 * \text{polso}) + (0.1236 * \text{altezza})$$

Massimo avambraccio:

$$(0.9626 * \text{polso}) + (0.0989 * \text{altezza})$$

Massimo cosce:

$$(1.3868 * \text{caviglia}) + (0.1805 * \text{altezza})$$

Massimo polpacci:

$$(0.9298 * \text{caviglia}) + (0.1210 * \text{altezza})$$

### Scheda for you.

Una volta ottenute le misurazioni massime esse vengono confrontate con le misurazioni attuali. Dopo aver fatto il confronto l'algoritmo è in grado di stabilire quale parte del corpo deve essere allenata di più rispetto alle altre quindi andrà a scegliere in modo casuale gli esercizi e gli stretching da assegnare. Una volta che l'utente inizia ad allenarsi i muscoli aumenteranno quindi i valori attuali dei muscoli subiranno una modifica che dovrà essere comunicata al sistema in questo modo l'algoritmo stabilirà nuovamente quali sono i muscoli che devono essere allenati con maggiore priorità e assegnerà nuovi esercizi e nuovi stretching. Il nostro sistema sfruttando la funzione *random()* e la miriade di esercizi e stretching che gli abbiamo fornito farà in modo che all'utente vengano assegnati sempre nuove schede di allenamento. Così facendo l'utente si troverà sempre a fare nuovi esercizi.

## 11. Quarta attività: IMPLEMENTAZIONE.

L'attività di implementazione segue quella di progettazione e trasforma il modello di progettazione in un sistema eseguibile. Non esiste un vero e proprio modello d'implementazione e spesso esso è lasciato alle capacità del team di sviluppo. Tuttavia un modello di implementazione trasforma elementi del modello di design in componenti e organizza le componenti secondo meccanismi di strutturazione e modularizzazione. Durante questa attività il sistema prende vita: da semplici diagrammi si passa a programmi eseguibili. Il modello di progettazione per essere tale deve comprendere almeno una delle seguenti documentazioni:

- Piano d'integrazione;
- Diagramma dei componenti;
- Diagramma di Deployment.

La scelta del team è stata quella di pubblicare il diagramma dei componenti.

### 11.1 Diagramma dei componenti.

Legenda:

Elemento	Simbolo/Notazione	Spiegazione
Componenti ("component")		Simbolo per i moduli di un sistema (interazione e comunicazione avvengono tramite interfaccia)
Interfaccia offerta		Simbolo di una o più interfacce definite in modo chiaro che mettono a disposizione funzioni, servizi o dati verso l'esterno (il semicerchio viene anche chiamato socket).
Interfaccia necessaria		Simbolo di un'interfaccia necessaria che riceve funzioni, servizi o dati dall'esterno (il cerchio con notazione con bastoncino viene chiamato anche notazione lollipop).
Porta		Il simbolo rappresenta un punto di interazione separato tra un componente e il proprio ambiente.
Relazione		Le linee fungono da connettori e indicano le relazioni tra componenti.

Figura 11 Legenda diagramma dei componenti.

## Diagramma dei componenti client-server.

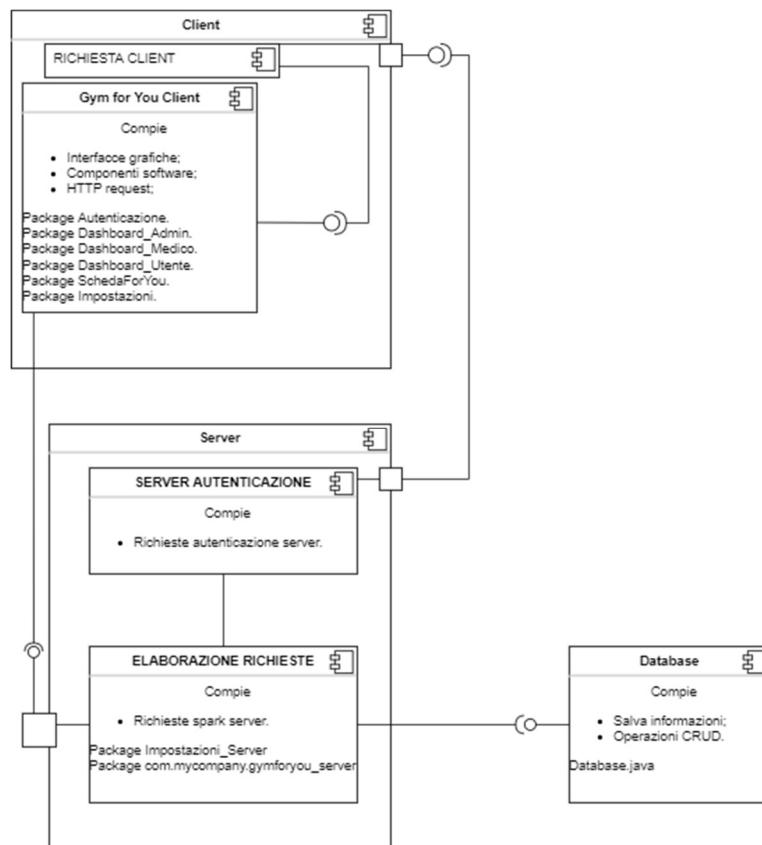


Figura 11.1 Diagramma dei componenti client-server.

## Diagramma dei componenti: Gym for you client.

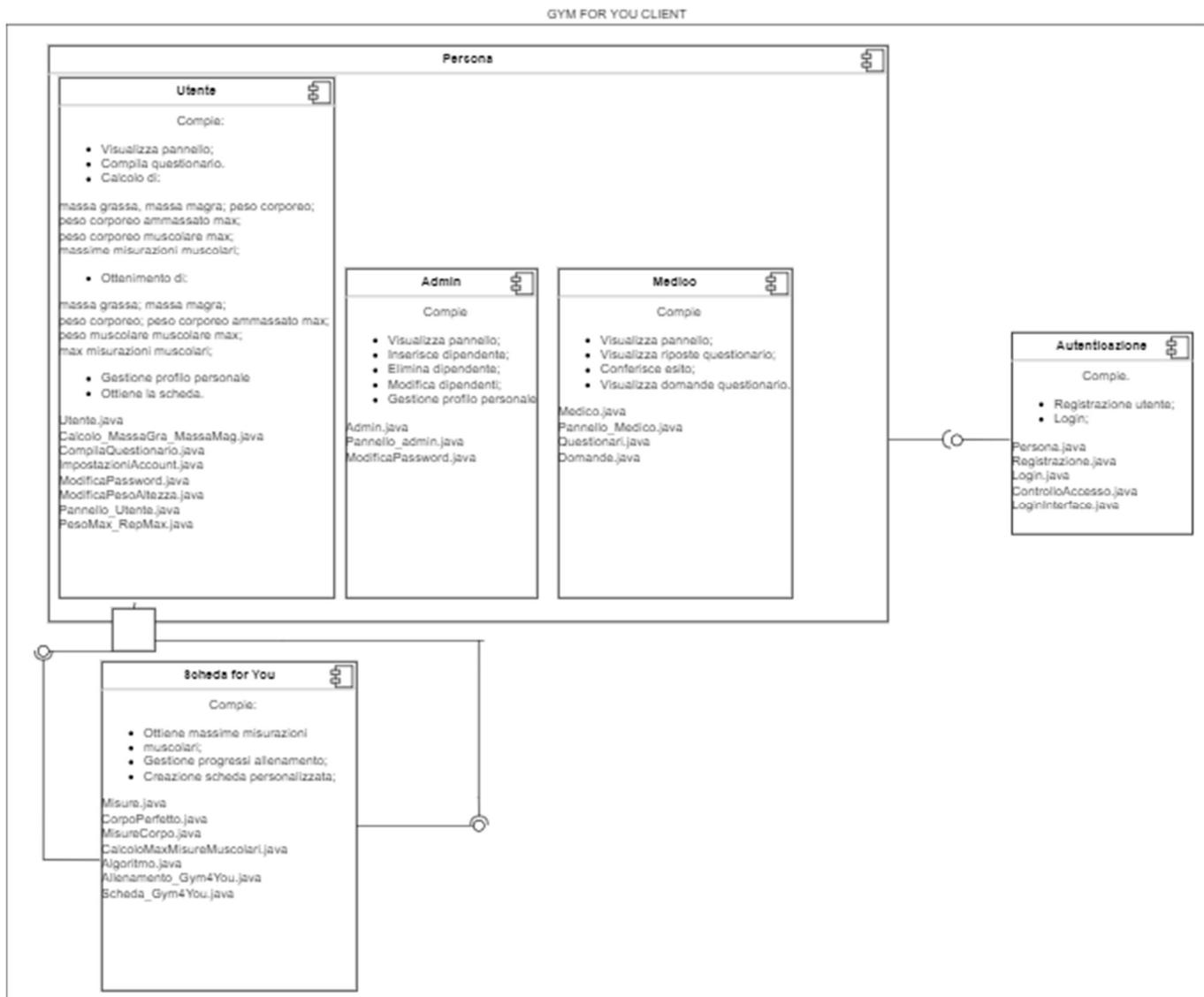


Figura 11.2 Diagramma dei componenti gym for you client.

## 12. Quinta attività: testing.

Quest'ultima fase prevede che il sistema venga testato per vedere se il rispetta i requisiti funzionali ed anche per capire in che modo si comporta il software in situazioni anomale. Per testare il software il team ha deciso di utilizzare lo **unit-testing**.

### Unit testing.

Unit testing è un tipo di test del software in cui vengono testate singole unità o componenti di un software. Lo scopo è convalidare che ogni unità del codice software funzioni come previsto. Lo Unit Testing viene eseguito durante lo sviluppo (fase di codifica) di un'applicazione da parte degli sviluppatori. I test unitari isolano una sezione di codice e ne verificano la correttezza. Un'unità può essere una singola funzione, metodo, procedura,

modulo o oggetto. Il team ha deciso di utilizzare lo unit testing perché presenta diversi vantaggi:

- 1) I test unitari aiutano a correggere i bug nelle prime fasi del ciclo di sviluppo e a risparmiare sui costi.
- 2) Aiuta gli sviluppatori a comprendere la base del codice di test e consente loro di apportare modifiche rapidamente
- 3) I test unitari aiutano con il riutilizzo del codice.
- 4) A causa della natura modulare del test unitario, possiamo testare parti del progetto senza attendere che altre vengano completate.

Lo unit testing quindi suddivide in moduli i componenti del sistema. Il team ha quindi deciso di testare il software facendo riferimento alla suddivisione che è stata fatta nell'analisi dei casi d'uso (si rimanda al paragrafo 6.1). Il team avendo sviluppato il software con un approccio modulare, attraverso unit testing ha potuto fare anche un tipo di testing modulare. Il nostro sistema si basa su moduli, classi, servizi, che richiedono la presenza di altri, per questo durante lo sviluppo del sistema si è potuto prima sviluppare, testare un modulo precedente e solo dopo avere ottenuto i risultati attesi, si è potuto sviluppare il modulo successivo. Grazie a questo tipo di approccio il team è riuscito a contenere i costi e i tempi di sviluppo inoltre ha evitato la creazione di grandi e irrisolvibili bug.

Il team essendo composto da due uniche persone ha anche deciso di utilizzare insieme a unit testing, un altro approccio che prende il nome di **test di coppia**. Nel test di coppia ai componenti del team vengono assegnati moduli, condividono idee e lavorano sulle stesse macchine per trovare i difetti. Una persona può eseguire i test (nel nostro caso Infortuna) e un'altra persona può prendere appunti sui risultati (nel nostro caso Primerano).

Per una maggiore comprensione adesso andremo ad analizzare come sono stati eseguiti i test e il comportamento che è stato assunto dal nostro software.

### Gestione accesso.

Questa parte del nostro sistema permette agli utenti di registrarsi e agli altri attori di poter effettuare il log-in. In questa fase il team ha provato l'inserimento di molteplici input per testare la loro validità.

- **Registra utente:** per avere una corretta registrazione devono essere rispettate diverse regole. Non usare un'email già utilizzata. L'utente deve inserire una password con una lunghezza compresa tra 8 - 15 caratteri. Le due password inserite negli appositi campi devono coincidere. Non possono essere lasciati campi vuoti oppure non si

possono compilare determinati campi con input non corretti. Il team ha provato molte volte con input diversi andando a stressare il sistema e in tutti i casi il sistema ha risposto secondo le aspettative. Inoltre nel caso in cui una delle regole descritte non viene rispettata il sistema fa apparire un alert che indica cosa si sta facendo di sbagliato.

- **Login:** qui il team ha provato più volte a immettere credenziali errate e in tutti i casi il sistema ha rispettato i requisiti. Il team ha poi provato ad entrare nel sistema con credenziali non registrate e il sistema ha rispettato i requisiti non consentendo l'accesso.
- **Logout:** qui il team ha voluto testare che nel momento del logout non si veniva reindirizzati in pagine errate, ciò non accade quindi il sistema rispetta i requisiti.

### Servizi senza il consenso del medico.

- **Inserire circonferenze:** in questo caso il team ha provato più volte ad inserire input non attesi dal sistema, e in tutti i casi il sistema ha risposto come dovuto non accettando tali input.
- **Compila questionario:** per fare in modo che il questionario venga inviato l'utente deve aver compilato tutte le domande, per quanto il team ha più volte provato a inviare il questionario tralasciando una o più risposte in tutti i casi il sistema non ha inviato il questionario e fatto apparire un messaggio di errore.
- **Calcolo masse:** qui il team ha riscontrato un problema, ovvero i risultati che venivano presentati erano errati. Dopo aver controllato le funzioni che hanno il compito di svolgere i calcoli il team si è accorto della mancanza di alcune parentesi fondamentali. Un altro problema che è stato riscontrato riguarda i dati che vengono usati per fare i calcoli, infatti inserendo i numeri e separando le cifre decimali da quelle intere tramite la virgola i dati venivano salvati proprio con la virgola il che portava a dei problemi. Per risolvere tale problema il team si è avvalso dell'uso della funzione *replace* che va a sostituire tutte le virgole con i punti decimali ogni volta che se ne presenta l'occasione.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Servizi senza il consenso del medico.

- **Inserimento circonferenze:** anche qui l'inserimento di valori non corretti viene prontamente rifiutato dal software.
- **Inserimento e modifica misure:** anche qui l'inserimento o la modifica di valori non corretti viene prontamente rifiutato dal software.

- **Ottenimento pesi e max misurazioni muscolari:** il team ha verificato che i calcoli venivano fatti in modo corretto e ciò in un primo momento non succedeva, anche in questo caso per colpa della presenza delle virgole, perciò si è prontamente intervenuto ad inserire la funzione *replace()* dove ritenuto necessario. Il team ha poi visionato che gli output fossero corretti e ciò accade.
- **Scheda for you:** in questo caso il team ha provato diversi scenari immedesimandosi in diverse tipologie di utente e ha verificato che il sistema risponde in maniera corretta a tutte le esigenze. Nella stampa della scheda si è testato che le immagini si muovessero tutte e ciò accade, infine si è anche testato che i nomi degli esercizi non si sovrapponessero e si è ottenuto un responso positivo da parte del sistema.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Gestione salute.

- **Visualizza risposte:** il team ha testato e verificato che le risposte visualizzate dal medico siano uguali a quelle inserite dall'utente e ciò accade. Dal punto di vista grafico le risposte vengono presentate in maniera corretta senza sovrapposizioni che renderebbero difficile la loro lettura. Inoltre il questionario a cui viene dato l'esito scompare dalla tabella e dopo diversi test il team non ha riscontrato errori riguardo ciò.
- **Valuta questionario:** il team ha testato che il medico riesca a dare il suo esito senza problemi cliccando l'apposito bottone.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Gestione sistema.

- **Visualizzazione dipendenti:** il team ha testato che all'interno della tabella predisposta per la visualizzazione non ci siano problemi grafici. Inoltre si è testato che non vengano mostrate righe errate. È stata testata la selezione della riga. Infine il dipendente che viene eliminato deve scomparire dalla tabella e dopo diversi test il team non ha riscontrato errori riguardo ciò.
- **Inserisci:** anche qui l'inserimento di valori non corretti viene prontamente rifiutato dal software.
- **Elimina:** il team ha testato l'eliminazione di un dipendente e l'azione avviene senza nessun problema.

- **Modifica:** anche qui l'inserimento di valori non corretti viene prontamente rifiutata dal software. Infatti se viene inserita la password vecchia in modo errato il sistema fa apparire un messaggio di errore, la stessa cosa accade se le due password inserite non coincidono.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### 13. Verifica.

Dopo aver testato il sistema e aver compreso che i requisiti sono stati rispettati è giusto andare a verificare che il software sia privo di errori.

#### Test 1: registrazione.

<b>Test</b>	<b>Input</b>	<b>Output</b>
Inserimento valori nome e cognome	Caratteri alfabetici	Inserimento corretto.
	Caratteri numerici.	Inserimento errato.
	Caratteri alfanumerici.	Inserimento errato.
Inserimento valori password	Caratteri alfanumerici	Inserimento corretto.
	Inserimennto caratteri minore lunghezza richiesta	Inserimento errato.
Inserimento valori peso e altezza	Caratteri alfabetici	Inserimento errato.
	Caratteri numerici con virgola positivi.	Inserimento corretto.
	Numeri negativi.	Inserimento errato.

#### Test 2: login.

<b>Test</b>	<b>Input</b>	<b>Output</b>
Inserimento valori email password	Credenziali corrette.	Accesso effettuato.
	Credenziali errate.	Accesso rifiutato.
	Campi vuoti.	Accesso rifiutato.

#### Test 3: inserimento circonferenze, inserimento e modifica misure.

<b>Test</b>	<b>Input</b>	<b>Output</b>
Inserimento circonferenze, inserimento e modifica misure.	Numeri positivi, sia interi che decimali con virgola o punto	Inserimento corretto.
	Numeri negativi.	Inserimento errato.
	Campi vuoti.	Errore.

I test condotti non hanno dunque rilevato apparenti casi di malfunzionamenti dovuti ad inserimenti errati di dati di input.

### 14. Ultima attività: DEPLOYMENT.

#### Diagramma di deployment.

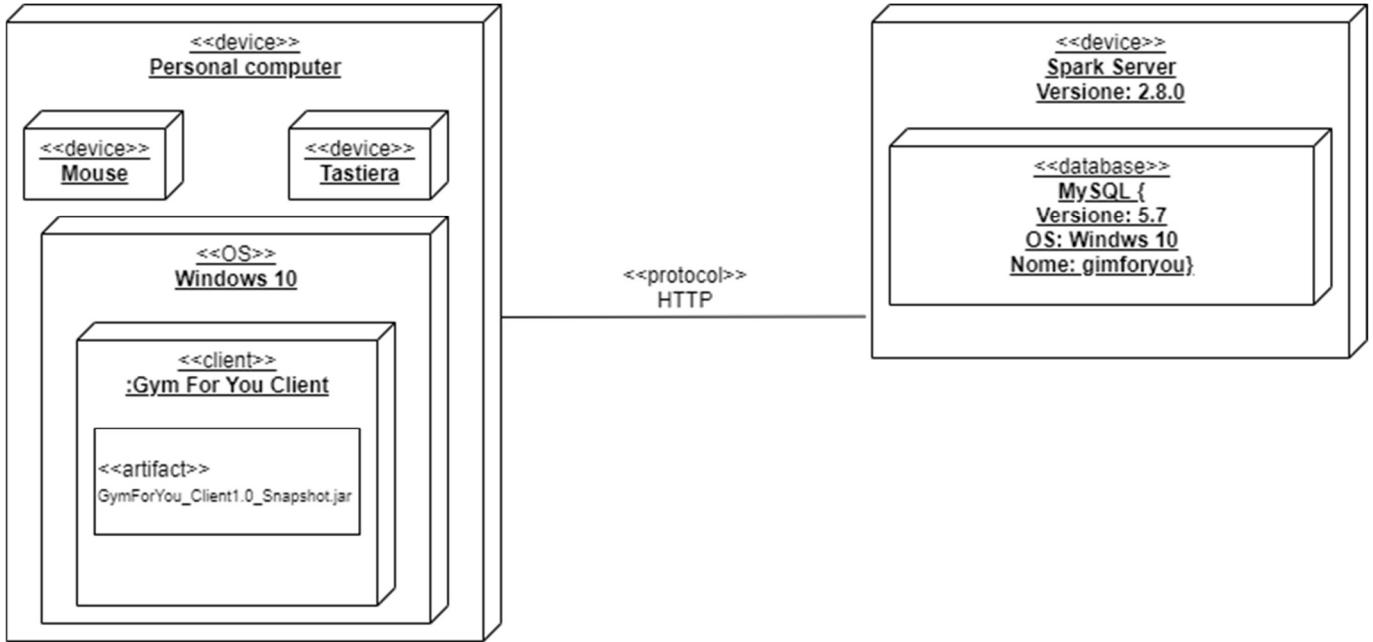


Figura Deployment diagram.

## Conclusioni.

Dopo che il primo prototipo è stato rilasciato, il cliente si dichiara abbastanza soddisfatto del risultato raggiunto. Il prototipo si dimostra già potenzialmente pronto per l'utilizzo. Nonostante la necessità di inserimento di molteplici parametri, il software appare semplice e intuitivo. Le diverse prove effettuate hanno mostrato un'ottima presentazione e distribuzione degli allenamenti, è stato apprezzato l'alto grado di personalizzazione che viene fornito all'utente. Il punto di forza del software risiede nella possibilità di fornire diverse schede di allenamento personalizzate, potendo variare a proprio volgìa l'allenamento che si vuole fare. Tuttavia il cliente richiede che vengano inserite ulteriori funzionalità che verranno presentate nel secondo prototipo.

## SECONDO PROTOTIPO.

Dopo aver presentato il primo prototipo al nostro cliente ed avere ricevuto dei feedback positivi, il team è partito con lo sviluppo delle nuove funzionalità. Le nuove funzionalità consistono in un sistema di comunicazione che metterà in contatto gli utenti con i medici ed i trainer. Il sistema di comunicazione si basa su un meccanismo di domanda e risposta. Gli utenti dovranno porre le domande e invece gli altri due attori dovranno rispondere, naturalmente gli utenti possono fare una o più domande allo stesso attore. Per analizzare l'operato dei medici e dei trainer il nostro cliente ha richiesto un servizio di valutazione, tale servizio si baserà su un giudizio che potrà essere espresso dagli utenti sulle risposte

ricevute e sui singoli dipendenti che fanno parte di *Gym For You*. Il giudizio verrà dato in forma di valutazione numerica che va da 0 a 5. Dalle valutazioni che verranno fatte sui singoli dipendenti verrà fatta una media voto che sarà visibile nel pannello personale di ogni attore. Un altro servizio che ci è stato richiesto riguarda il calcolo del peso massimo e delle ripetizioni massime che un singolo utente può fare utilizzando un peso. Questo servizio è utile soprattutto per gli utenti che praticano il sollevamento dei pesi in modo amatoriale. Quest'ultimo servizio è stato implementato grazie all'utilizzo della formula di Brzycki trovata all'interno del libro presente nella bibliografia.

## 15. Prima fase INCEPTION.

### 15.1 Analisi dei requisiti.

Descriviamo i nuovi requisiti del sistema che si aggiungono a quelli descritti nel paragrafo 5.1.1.

### 15.2 Descrizione stakeholder.

Oltre agli stakeholder descritti nel paragrafo 5.1.2, qui andremo a descrivere quelle che saranno le nuove funzionalità che si aggiungono a quelle già descritte:

- **Utente:** oltre alle funzionalità già descritte l'utente adesso potrà comunicare con i medici e i trainer e valutare le risposte ricevute da essi, potrà anche valutare il loro operato. Potrà sapere quanto peso riesce ad alzare e quante ripetizioni può fare con un determinato peso.
- **Trainers:** potranno comunicare con gli utenti se quest'ultimi necessitano di un loro parere o consiglio. Possono ricevere una valutazione personale oppure sulle risposte date, valutazione che verrà fatta dagli utenti.
- **Medici:** potranno interagire con gli utenti per rispondere alle loro domande e potranno ricevere delle valutazioni sulle riposte date e sul loro lavoro compiuto.

Come si può vedere è stato aggiunto un nuovo stakeholder (il trainer) che va ad arricchire la famiglia di *Gym For You*. Il suo principale compito sarà quello di consigliare gli utenti sull'allenamento e di rispondere alle domande che essi gli pongono.

### 15.3 Specifica dei requisiti.

Oltre ai requisiti descritti nel paragrafo 5.1.3 aggiungiamo i seguenti requisiti:

### 15.4 Requisiti funzionali.

#### **Utente può:**

- Chiedere un parere al medico e al trainer.
- Valutare la risposta ottenuta.
- Valutare i medici e i trainer.
- Calcolare il peso massimo che riesce ad alzare.
- Calcolare le massime ripetizioni che riesce a fare con un determinato peso.

#### **Medico può:**

- Comunicare con l'utente.
- Ricevere delle valutazioni.

#### **Trainer può:**

- Autenticarsi.
- Comunicare con gli utenti.
- Ricevere delle valutazioni.

### 15.5 Indagine sul sistema.

### 15.6 Analisi del dominio.

Il dominio applicativo del nostro software è immutato da quello descritto nel paragrafo 5.2.1.

Per comprendere al meglio il nuovo dominio applicativo il team ha ampliato il context diagram (figura 15) già presente nel sopracitato paragrafo (figura 5). Ecco il nuovo diagramma:

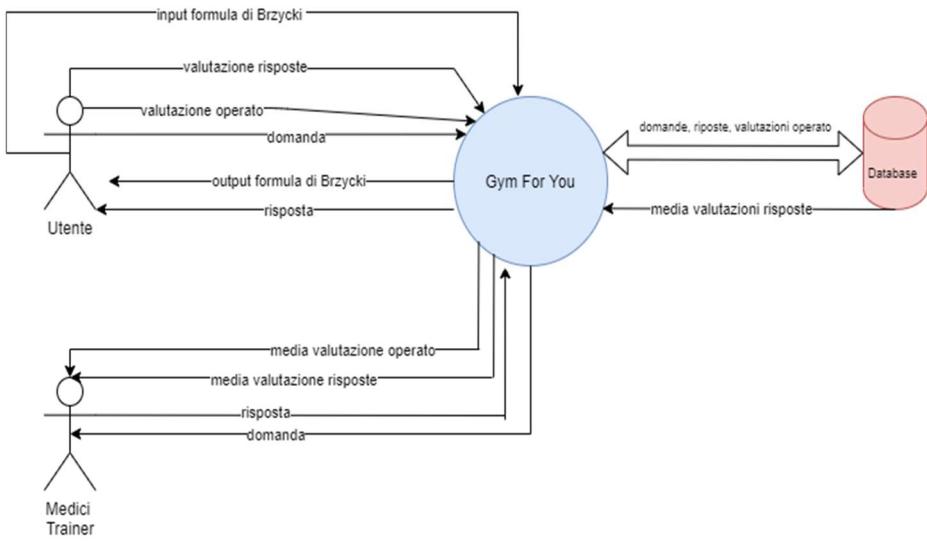


Figura 15 Context diagram aggiornato.

## 15.7 Studio fattibilità.

Per lo studio di fattibilità si rimanda la paragrafo 5.2.2 in quanto le caratteristiche fondamentali del sistema sono rimaste invariate.

# 16. Analisi dei rischi.

## 16.1 Rischi di progetto.

I rischi che possono capitare quando un utente vuole iniziare una comunicazione, nascono dai problemi legati all'instabilità della connessione. Questo rischio non può essere gestito dal team in quanto è una problematica relativa ai provider del servizio. Un altro rischio che potrebbe sussistere riguarda il sovraccarico dei server, anch'esso non gestibile dal team. Il team per sviluppare il servizio di comunicazione si è avvalso dell'utilizzo delle migliori tecnologie, infatti è stato utilizzato il protocollo HTTP che garantisce sicurezza e affidabilità, inoltre il nostro sistema si basa su un server strutturato tramite le tecnologie messe a disposizione da Spark. Quindi il team auspica che i rischi che potranno colpire il nostro sistema sono di natura esterna al sistema stesso. Per quanto riguarda i rischi legati al calcolo della formula di Brzycki sono molto bassi quasi inesistenti, in quanto i calcoli da fare risultano modesti.

## 16.2 Rischi di prodotto.

I rischi di prodotto rimangono immutati a quelli presenti nel paragrafo 5.3.2 (pagina 31) perciò si rimanda al suddetto paragrafo.

## 16.3 Tabella valutazione rischi.

La seguente tabella riprende quella presente nel paragrafo 5.3.3 e aggiunge i seguenti campi:

RISCHI	VALUTAZIONE RISCHIO	PROBABILITA' CHE ACCADA	CONSEGUENZE	METODO DI RISOLUZIONE
Impossibilità di inviare una domanda al medico o al trainer.	Medio.	Abbastanza Probabile.	L'utente non riesce a comunicare con il medico oppure con il trainer.	Il team utilizza le migliori tecnologie per la comunicazione per evitare la presenza di questo rischio. Ma è un rischio che potrebbe presentarsi nel momento in cui le infrastrutture di rete vengono meno.
Impossibilità di ricevere una risposta da un medico o da un trainer.	Medio.	Abbastanza Probabile.	I medici e i trainer non riescono a comunicare con gli utenti.	Anche in questo caso il team garantisce le migliori tecnologie però il rischio potrebbe sempre nascere a causa di problematiche relative al provider dei servizi di rete.
Impossibilità di valutare un medico o un trainer.	Medio.	Abbastanza Probabile.	Gli utenti non riescono a valutare una risposta ottenuta oppure l'operato dei medici o dei trainer.	Rischio anche in questo caso dovuto alle problematiche della rete. Non gestibile dal team.
Impossibilità di ottenere il risultato dalla formula di Brzycki.	Basso.	Poco probabile.	L'utente non riesce a visualizzare il peso e le ripetizioni che deve fare.	Il team ha predisposto dei controlli sugli input nel caso in cui essi siano non validi vengono subito rifiutati.

## SECONDO PROTOTIPO.

## 17. Seconda fase ELABORATION.

### 17.1 Prima attività requisiti.

Oltre ai requisiti già presentati nel capitolo 6 (pagina 33) andremo a descrivere i nuovi requisiti inerenti a questo nuovo prototipo. Anche qui abbiamo deciso di suddividere il nostro sistema in 4 macro-parti:

- 1) Invio domande.
- 2) Invio risposte.
- 3) Valutazione risposte e dipendenti.
- 4) Gestione accesso.
- 5) Formula di Brzycki.

#### 17.1.2 Requisiti funzionali.

##### **Requisito A: Invio domande.**

**ID A.A.A:** Visualizzazione trainer.

**ID A.A.B:** Visualizzazione medici.

**ID A.A.C:** Invio domanda.

**ID A.A.D:** Visualizzazione domande inviate e risposte ricevute.

## **Requisito B: Invio risposte.**

**B.A.A:** Visualizzazione domande ricevute.

**B.A.B:** Invio risposta.

## **Requisito C: Valutazione risposte e dipendenti.**

**C.A.A:** Valutazione risposte ricevute.

**C.A.B:** Visualizzazione medici.

**C.A.C:** Valutazione medico.

**C.A.D:** Visualizzazione trainer.

**C.A.E:** Valutazione trainer.

## **Requisito D: Gestione accesso TRAINER.**

**D.A.A:** Log-in.

**D.A.B:** Log-out.

## **Requisito E: Formula di Brzycki.**

**E.A.A:** Inserimento valori.

**E.A.B:** Visualizzazione risultati formula.

### **17.1.3 Requisiti non funzionali.**

- 1) Utilizzo di una database per la memorizzazione delle domande e delle risposte.
- 2) Trattamento dei dati nel maggior rispetto della privacy e rispetto delle leggi vigenti a riguardo.
- 3) Compatibilità con diverse piattaforme.

### **17.1.4 Documento SRS.**

NOME REQUISITO	ID	PRIORITA'	DESCRIZIONE	REQUISITO PADRE	REQUISITO FIGLIO
Invio domande	A	Indispensabile.	Permette all'utente di inviare delle domande ai trainer o ai medici per ottenere un loro parere.		A.A.A; A.A.B A.A.C; A.A.D
Visualizzazione trainer.	A.A.A	Indispensabile	Permette di visualizzare tutti i trainer presenti a servizio degli utenti.	A	
Visualizzazione medici.	A.A.B	Sarebbe meglio averlo.	Permette di visualizzare tutti i	A	

			medici presenti a servizio degli utenti.		
Invio domanda.	A.A.C	Indispensabile	Permette l'invio della domanda.	A	
Visualizzazione domande inviate e risposte ricevute.	A.A.D	Sarebbe meglio averlo.	Permette all'utente di visualizzare tutte le domande inviate e le risposte ricevute così da avere uno storico.	A	
Invio risposte.	B	Indispensabile	Permette ai medici e ai trainer di rispondere ad una domanda dell'utente.	B	B.A.A; B.A.B
Visualizzazione domande ricevute.	B.A.A	Sarebbe meglio averlo.	Permette ai medici e ai trainer di visualizzare le domande ricevute.	B	
Invio risposta.	B.A.B	Indispensabile	Permette ai medici e ai trainer di rispondere alle domande.	B	
Valutazione risposte e dipendenti	C	Indispensabile	Permette agli utenti di votare le risposte ricevute i trainer e i medici..		C.A.A; C.A.B C.A.C; C.A.D C.A.E
Valutazione risposte ricevute	C.A.A	Indispensabile	Permette di valutare le risposte ricevute.	C	
Visualizzazione medici	C.A.B	Sarebbe meglio averlo.	Permette di visualizzare i medici.	C	
Valutazione medico	C.A.C	Indispensabile	Permette di valutare i medici.	C	
Visualizzazione trainer	C.A.D	Sarebbe meglio averlo.	Permette di visualizzare i trainer.	C	
Valutazione trainer	C.A.E	Indispensabile	Permette di valutare i trainer.	C	
Gestione accesso.	D	Indispensabile.	Permette ai trainer di accedere ed uscire dalla propria pagina personale.		D.A.A; D.A.B
Log-in Trainer	D.A.A	Indispensabile.	Permette ai trainer di accedere al sistema.	D	
Log-out Trainer.	D.A.B	Indispensabile.	Permette ai trainer di uscire dal sistema.	D	
Formula di Brzycki.	E	Sarebbe meglio avere.	Permette il calcolo della formula indicata.	E	E.A.A; E.A.B.
Inserimento valori.	E.A.A	Sarebbe meglio avere.	Permette di inserire gli input per il calcolo della formula	E	
Visualizzazione risultati.	E.A.B	Sarebbe meglio avere.	Permette di visualizzare i risultati della formula di Brzycki.	E	

### 17.1.5 Diagramma dei casi d'uso.

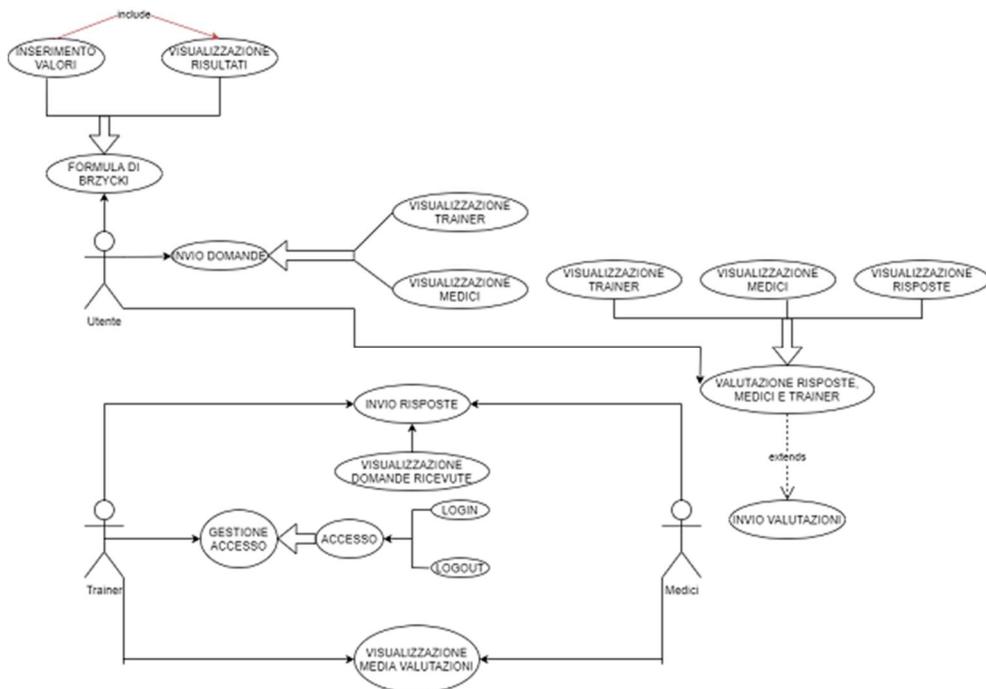


Figura 17 Use-Case diagram secondo prototipo.

### 17.1.6 Descrizione attori.

<b>ATTORE</b>	Utente.
<b>ID</b>	1
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Egli è colui il quale potrà chiedere pareri ai trainer o ai medici e potrà anche valutare il loro operato e le risposte ricevute.

<b>ATTORE</b>	Medico.
<b>ID</b>	2
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Ha il compito di rispondere alle domande che gli vengono poste dagli utenti.

<b>ATTORE</b>	Trainer.
<b>ID</b>	3
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Ha il compito di rispondere alle domande che gli vengono poste dagli utenti.

### 17.1.7 Descrizione casi d'uso.

Caso d'uso	Invio domande.
ID	A.
Descrizione	Permette agli utenti di visualizzare tutti i medici e i trainer che fanno parte di <i>Gym For You</i> . Dopo averli scelti l'utente può inviare loro una o più domande.
Attori	Utente.
Pre-condizioni	Avere effettuato il log-in.
Flusso principale	Accede al suo pannello.
Scenario secondario.	Errore durante il log-in. Impossibilità di accedere al proprio pannello. Superamento dei caratteri consentiti per la domanda.
Flusso alternativo.	Non accede al pannello. Non viene inviata la domanda. In tutti e due i casi apparirà un pop-up con l'errore.
Post-condizioni	Accesso al pannello e invio domanda.

Caso d'uso	Invio risposta.
ID	B.
Descrizione	I medici e i trainer dopo aver effettuato l'accesso visualizzano tutte le domande che gli sono state poste e hanno il compito di rispondere.
Attori	Medici, Trainer.
Pre-condizioni	Avere effettuato l'accesso, avere ricevuto almeno una domanda.
Flusso principale	Accedono al loro pannello e rispondono alle domande.
Scenario secondario.	Errore di accesso. Superamento dei caratteri consentiti per l'invio della risposta.
Flusso alternativo.	Non accede al pannello. Non viene inviata la risposta. In tutti e due i casi apparirà un pop-up con l'errore.
Post-condizioni	Accesso al pannello e invio risposta.

Caso d'uso	Valutazione risposte e dipendenti.
ID	C.
Descrizione	L'utente deve effettuare l'accesso. Se ha inviato una domanda e ricevuto una risposta può dare una valutazione alla risposta ricevuta. Può valutare l'operato dei medici e dei trainer.

Attori	Utente.
Pre-condizioni	Avere effettuato l'accesso. Avere ricevuto almeno una risposta per poter essere valutata.
Flusso principale	Accesso al pannello e valutazione.
Scenario secondario.	Errore durante il log-in.
Flusso alternativo.	Appare un pop-up che comunica l'errore durante la fase di accesso al sistema.
Post-condizioni	Invio valutazione per la risposta e/o per il dipendente.

Caso d'uso	Gestione accesso.
ID	D.
Descrizione	Il trainer utilizza questa operazione per accedere alla piattaforma. Abbiamo delle altre sotto-operazioni che permettono di entrare nella piattaforma log-in e di uscirne log-out.
Attori	Trainer.
Pre-condizioni	Avere effettuato l'accesso. Avere ricevuto almeno una risposta per poter essere valutata.
Flusso principale	Accede al suo pannello.
Scenario secondario.	Il trainer non inserisce bene le sue credenziali e non entra nel suo pannello.
Flusso alternativo.	Non accede al pannello e viene visualizzato un pop up con l'errore.
Post-condizioni	Accesso al pannello medico.

Caso d'uso	Formula di Brzycki.
ID	E.
Descrizione	L'utente inserisce gli input richiesti, il sistema fa i calcoli dovuti e restituisce i risultati.
Attori	Utente.
Pre-condizioni	Avere effettuato l'accesso. Inserimento dei valori richiesti.
Flusso principale	Visualizzazione risultato.
Scenario secondario.	L'utente non inserisce in modo corretto gli input richiesti.
Flusso alternativo.	Impossibili visualizzare i risultati della formula.
Post-condizioni	Visualizzazioni risultati formula.

## 17.1.8 Diagramma delle attività.

Nei prossimi paragrafi presentiamo i diagrammi di attività relativi al nuovo prototipo. Per evitare la ridondanza di seguito sono riportati i diagrammi relativi alle nuove attività e funzionalità del nostro sistema.

### Diagramma delle attività utente.

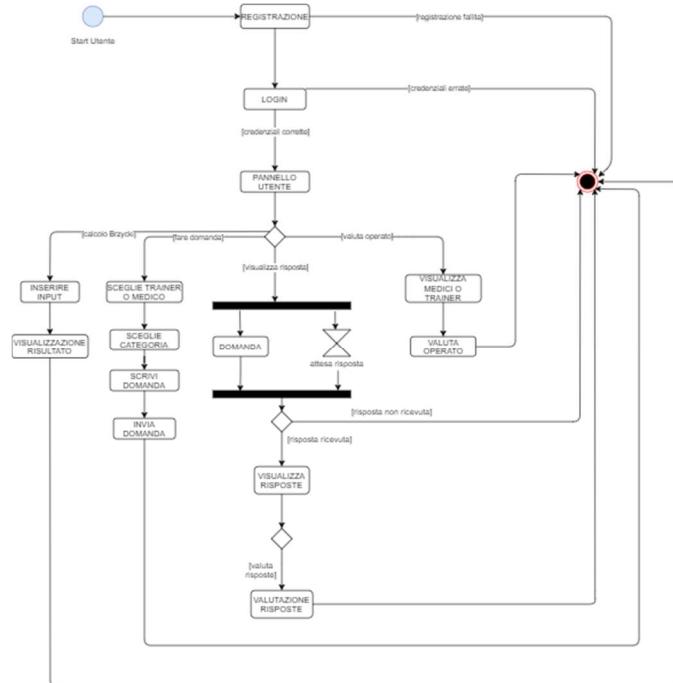


Figura 17.1 Activity diagram utente.

### Diagramma della attività trainer e medico.

Nel seguente diagramma (figura 17.2) è riportato il diagramma delle attività sia dei trainer che dei medici, il team ha deciso di unificare i diagrammi in quanto lo svolgimento delle attività è il medesimo per ambedue gli attori.

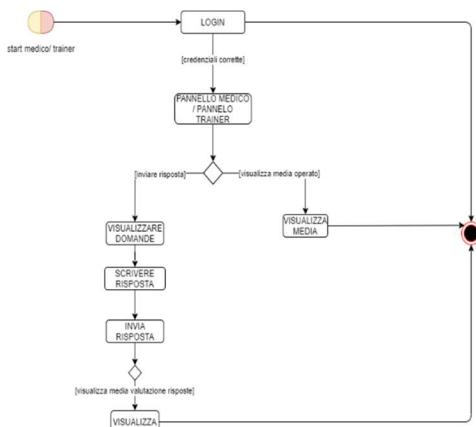


Figura 17.2 Activity diagram trainer e medico

## SECONDO PROTOTIPO.

### 18. Seconda attività: ANALISI.

Analisi architetturale.

#### 18.1 Diagramma dei package.

Il nostro sistema oltre ai package descritti nel paragrafo 7.1 è composto da 2 nuovi package:

- 1) Dashboard Trainer;
- 2) Consultazioni.

Per una questione di ordine abbiamo deciso di suddividere quest'ultimo package in tre diversi package figli in cui sono state inseriti dei file relativi alle operazioni che riguardano gli utenti, quelle che riguardano i trainer e infine quelli che riguardano i medici.

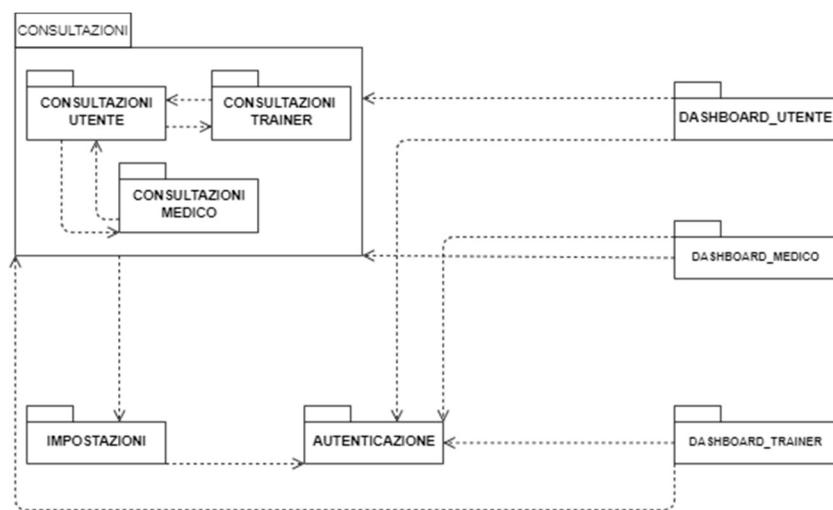
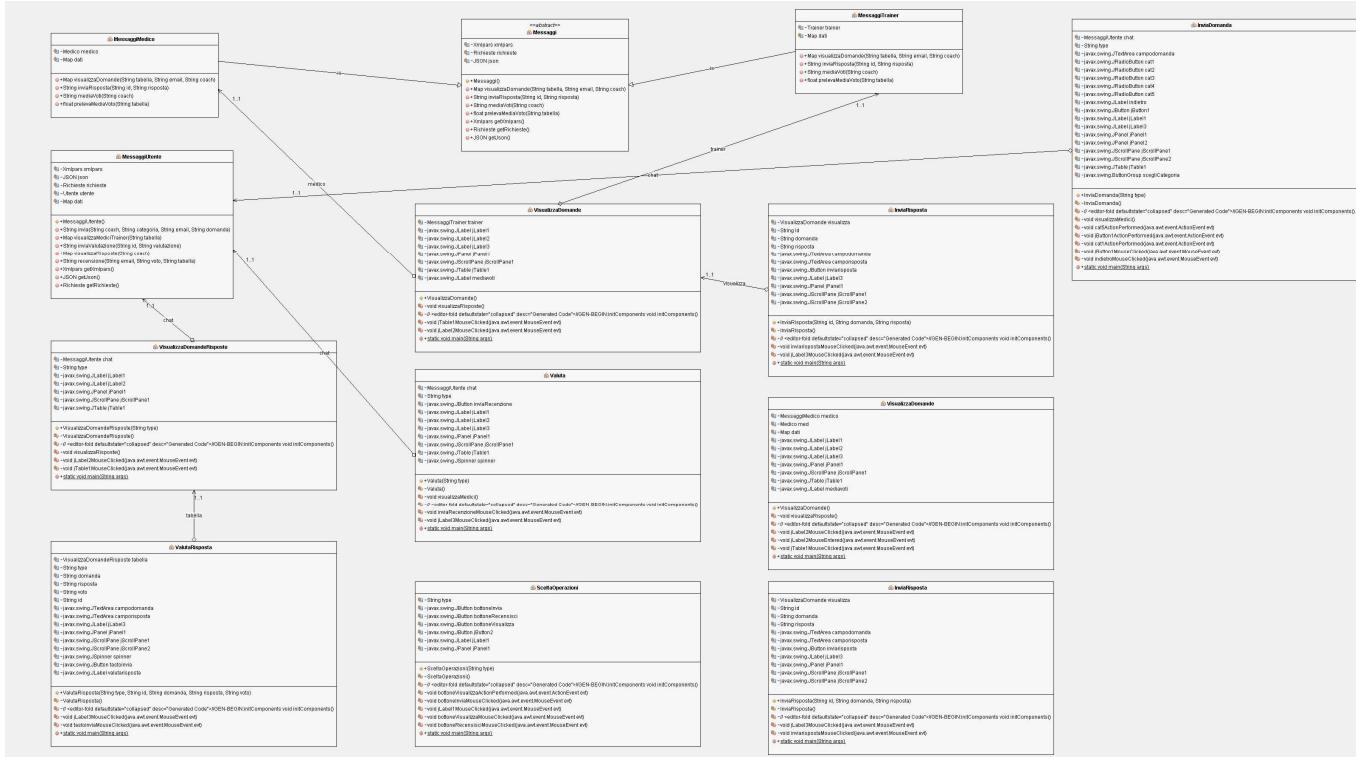


Figura 18 Package diagram secondo prototipo.

#### 18.2 Individuazione delle classi di analisi.

Oltre alle classi che sono state descritte nel paragrafo 7.3 in seguito descriviamo le nuove classi che sono state create per questo secondo prototipo. Anche in questo caso per ottenere i diagrammi delle classi abbiamo utilizzato il plug-in di Netbeans, EasyUML.

## Diagramma delle classi: CONSULTAZIONI.



*Figura 18.1 Class diagram consultazioni*

# Diagramma delle classi: Dashboard Utente.

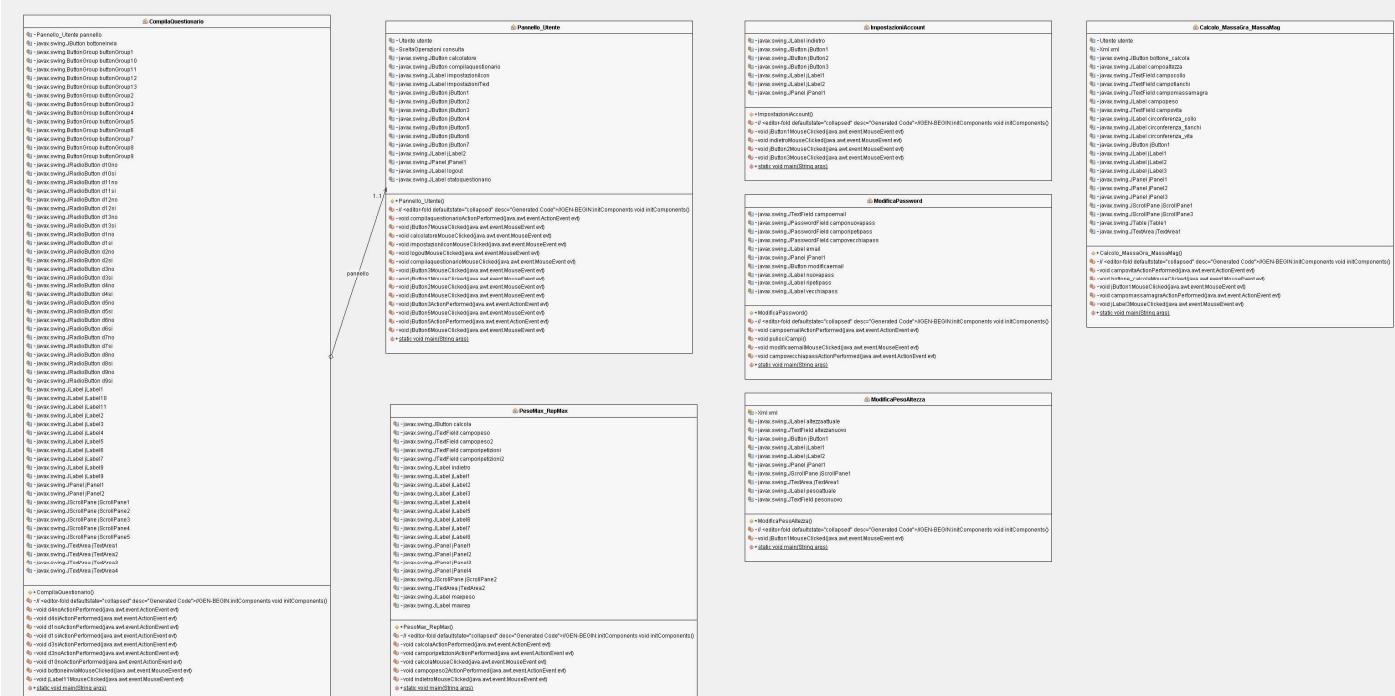
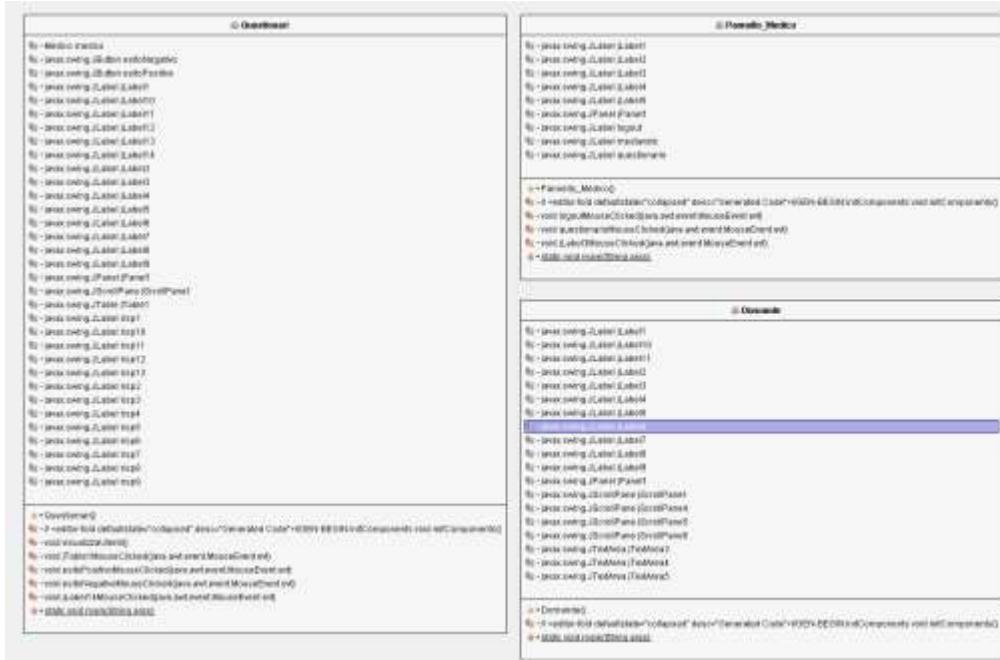


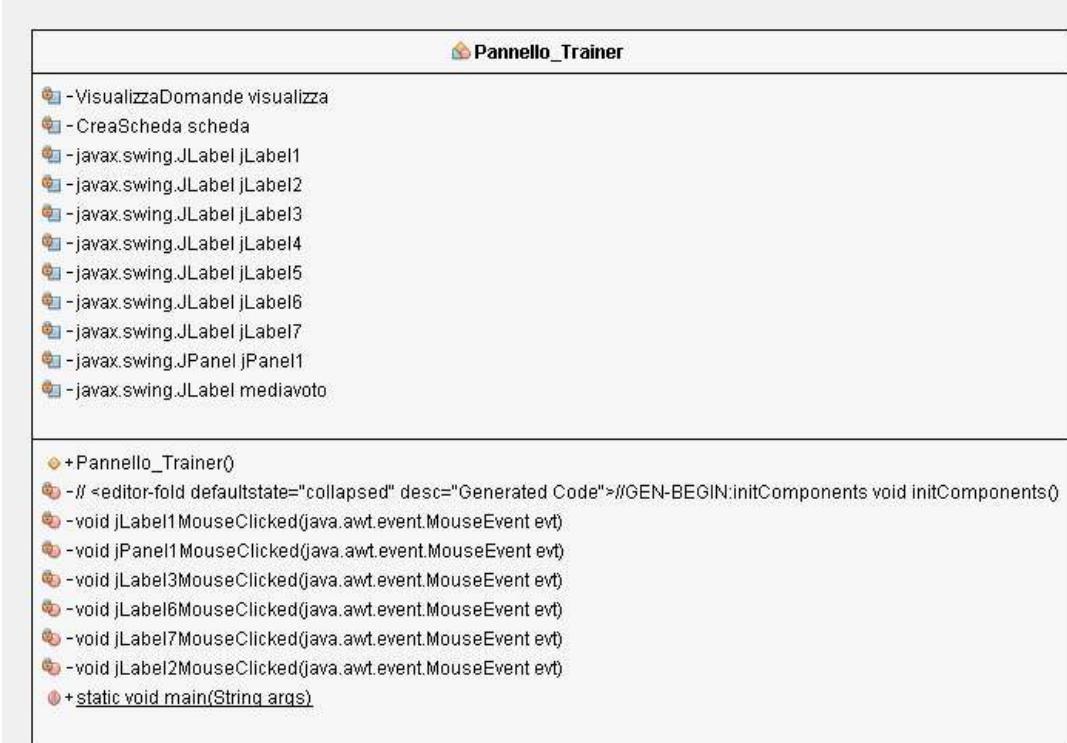
Figura 18.2 Class diagram Dashboard dashboard Utente.

## Diagramma delle classi: Dashboard Medico.



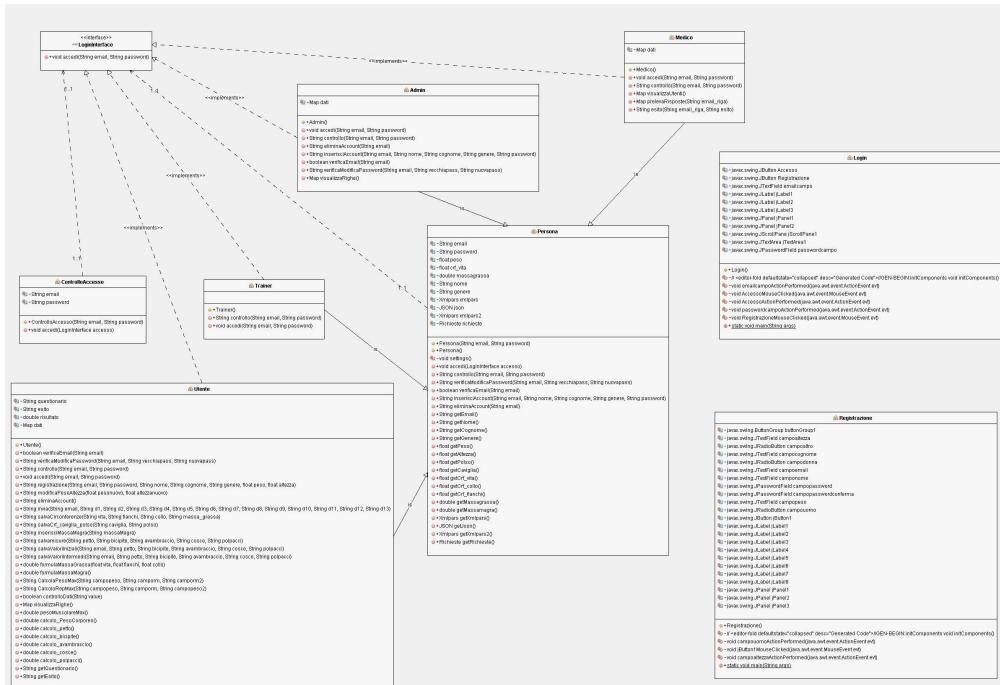
*Figura 18.3 Class diagram dashboard Medico.*

# Diagramma delle classi: Dashboard Trainer.



*Figura 18.4 Class diagram dashboard Trainer.*

## Diagramma delle classi: Autenticazione.



*Figura 18.5 Class diagram Autenticazione.*

## Analisi dei casi d'uso.

## 19. Diagrammi di sequenza.

Anche per questo secondo prototipo abbiamo utilizzato l'architettura client-server. Quindi il funzionamento dei servizi proposti è uguale a quello descritto nel paragrafo 8.

## Diagramma di sequenza Utente.

Il diagramma in figura 19, mostra le operazioni che può fare l'utente. Per non essere ripetitivi evitiamo di riscrivere i procedimenti che devono essere eseguiti per la registrazione e per l'accesso al nostro software perciò si rimanda al diagramma di sequanza presente nel capitolo dedicato al primo prototipo. In questo nuovo diagramma di sequenza il server riceve dal database la lista di tutti i medici e i trainer, il server quindi li passerà al client e l'utente potrà scegliere a chi inviare la domanda. L'invio della domanda viene fatto in modo che il client invii la domanda al server e in seguito essa viene inviata al database e inserita nella tabella dedicata. Se la domanda supera 250 caratteri il client riceverà un alert. Per visualizzare le risposte l'utente comunicherà al client quale risposta visualizzare, la risposta scelta sarà presa dal database passata server e poi rimandata al

client. L'utente può esprimere un voto per ogni risposta ricevuta, il voto verrà inviato dal client al server per poi essere memorizzato all'interno del db. Il procedimento di invio del voto è uguale quando l'utente vuole valutare un dipendente, l'unica differenza sta nel fatto che le risposte possono essere valutate solo una volta invece i lavoratori possono essere valutati quante volte si vuole. Per il calcolo della formula di Brzycki il procedimento consiste nell'inserire gli input richiesti dalla formula all'interno del frame dedicato dopodiché se gli input sono corretti il client dopo aver fatto i calcoli del caso, restituisce all'utente il risultato della formula.

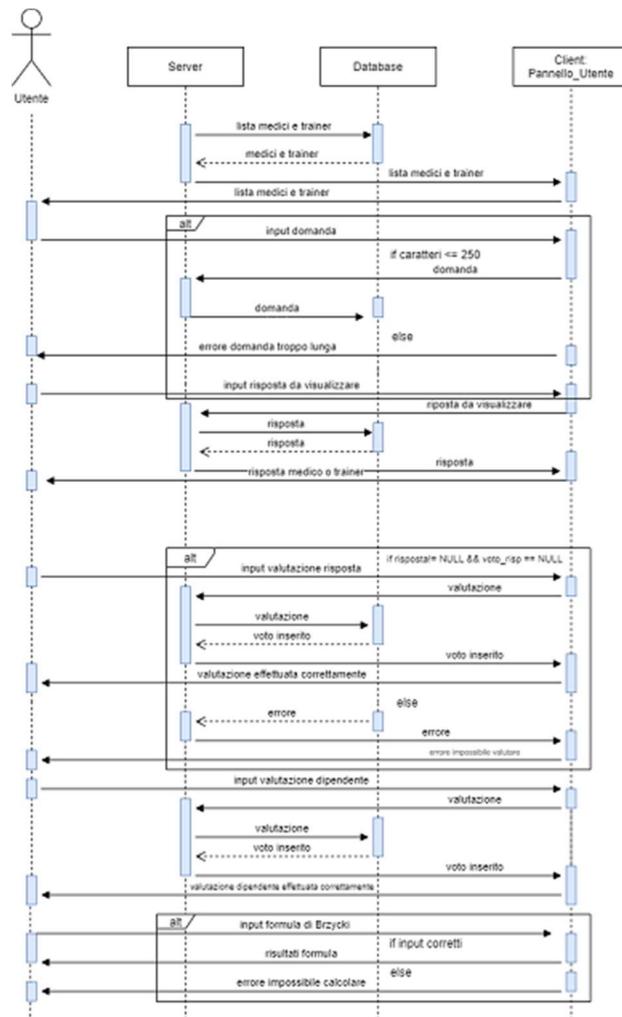


Figura 19 Sequence diagram utente.

## Diagramma di sequenza Medico.

Come si può vedere nel diagramma di sequenza del medico (figura 19.1) la prima operazione che deve essere fatta è quella del login (siccome il diagramma dei sequenza relativo al login è stato già realizzato si rimanda al capitolo dedicato), se le condizioni vengono rispettate si può entrare nella piattaforma dedicata. Una volta entrato il medico visualizzerà tutte le

domande che gli sono state inviate e sceglierà quella da visualizzare, per fare ciò il server fa una chiamata al db che restituisce tutte le domande per ogni singolo medico in base all'email, il server le passa al client per mostrarle all'interno del pannello dedicato al medico. Lo specialista sceglierà la domanda a cui rispondere e scriverà la risposta, questa risposta verrà inviata dal client al server, quest'ultimo la passerà al database che avrà il compito di memorizzarla. Se il medico non rispetta la lunghezza di 250 caratteri visualizzerà un messaggio di errore e non potrà inviare la risposta. Per ottenere una valutazione media sulle risposte date da ogni singolo medico, il server fa una chiamata al database che restituisce un valore che esprime la valutazione media delle risposte date, il server riceverà questo valore e lo invierà al client per farlo vedere all'interno del pannello dedicato allo stakeholder in questione. Per ottenere una valutazione media sull'operato di ogni singolo medico verrà utilizzato un procedimento uguale a quello descritto in precedenza.

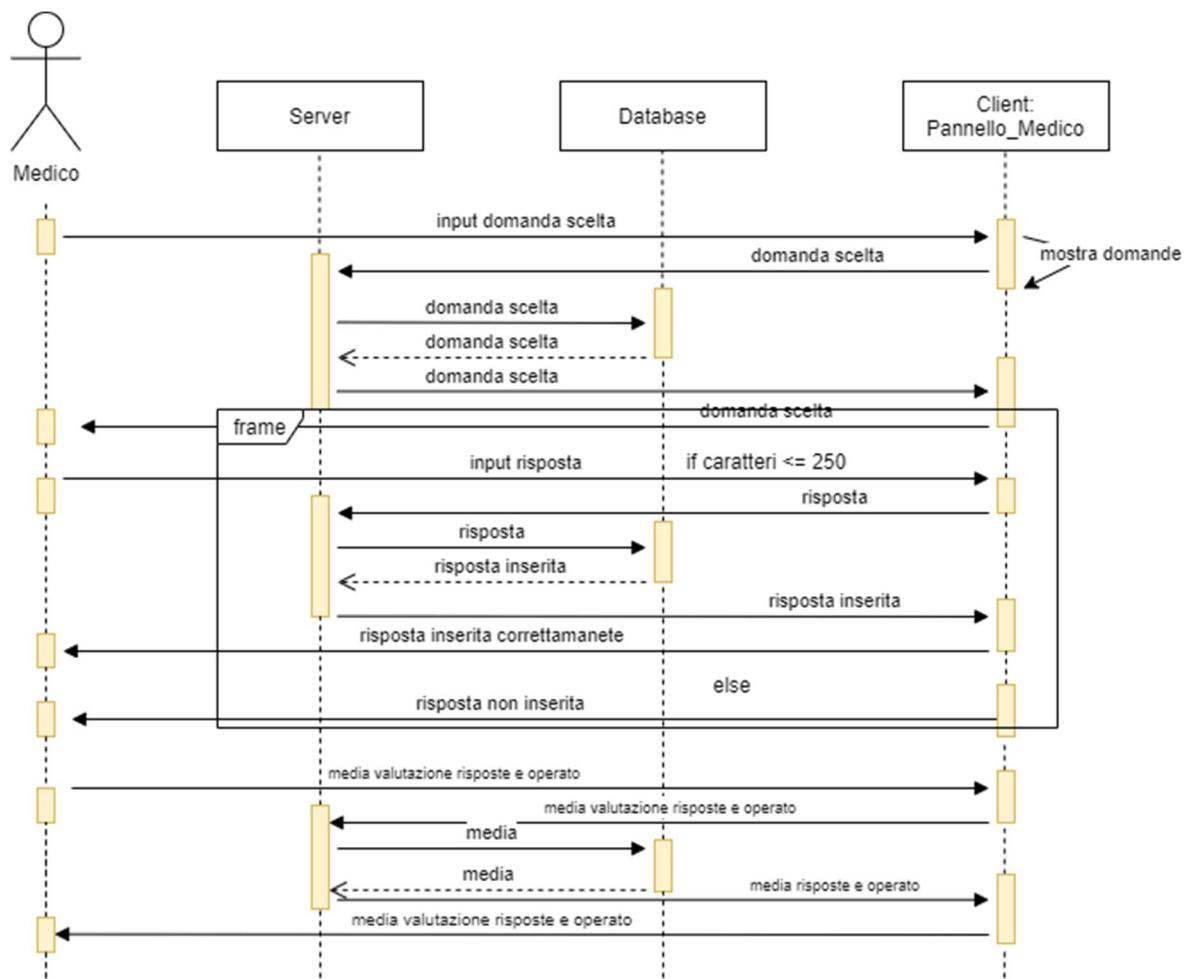


Figura 19.1 Sequence diagram medico.

## Diagramma di sequenza Trainer.

Come si può vedere nel diagramma di sequenza del trainer (figura 19.2) la prima operazione che deve essere fatta è quella del login, se le condizioni vengono rispettate si può entrare nella piattaforma dedicata. Una volta entrato il trainer visualizzerà tutte le domande che gli sono state inviate, per fare ciò il server fa una chiamata al db che restituisce tutte le domande per ogni singolo trainer in base all'email, il server le passa al client per mostrarle all'interno del pannello dedicato. L'allenatore sceglierà la domanda a cui rispondere e scriverà la risposta, questa risposta verrà inviata dal client al server, quest'ultimo la passerà al database che avrà il compito di memorizzarla. Se il trainer non rispetta la lunghezza di 250 caratteri visualizzerà un messaggio di errore e non potrà inviare la risposta. Per ottenere una valutazione media sulle risposte date da ogni singolo trainer, il server fa una chiamata al database che restituisce un valore che esprime la valutazione media delle risposte date, il server riceverà questo valore e lo invierà al client per farlo vedere all'interno del pannello dedicato allo stakeholder in questione. Per ottenere una valutazione media sull'operato di ogni singolo trainer verrà utilizzato un procedimento uguale a quello descritto in precedenza.

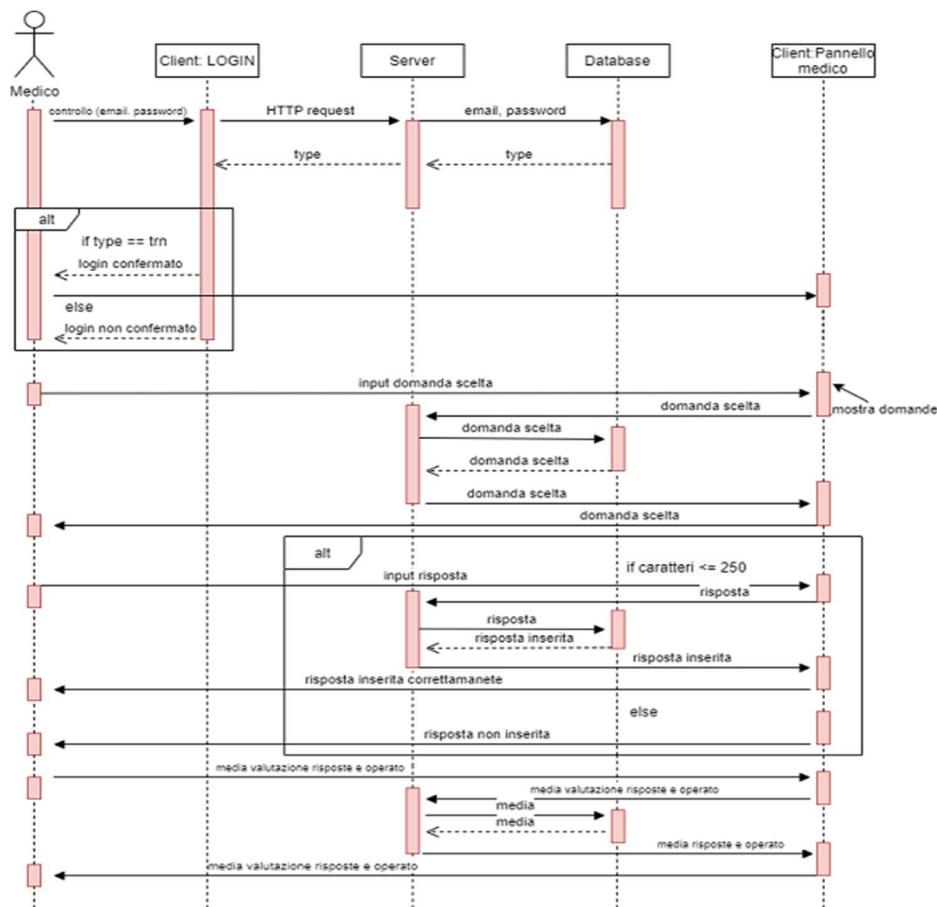


Figura 19.2 Sequence diagram trainer.

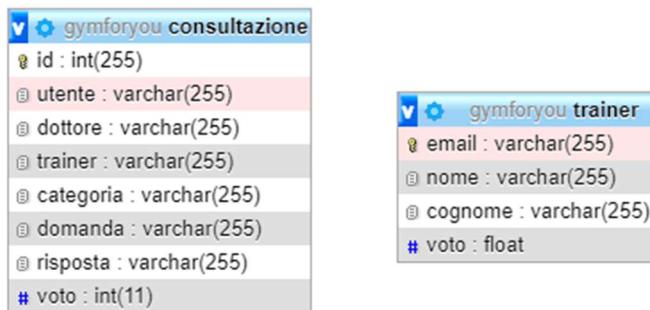
## 20. Terza attività: PROGETTAZIONE.

### 20.1 Architettura di sistema.

L'architettura del nostro sistema rimane immutata a quella presentata nel primo prototipo perciò per una maggiore comprensione si rimanda al capitolo 9. I ruoli di client e server rimangono uguali a quelli già descritti.

### 20.2 Database (db).

Il database per queste nuove funzionalità verrà utilizzato per immagazzinare tutte le domande e le risposte. In questo modo tutti gli stakeholder potranno avere uno storico della comunicazione. Il database viene anche utilizzato per salvare le votazioni che sono state fatte dagli utenti. Grazie a questo è possibile avere una valutazione media sulle risposte date da medici e trainer e sulle valutazioni date dagli utenti ai singoli attori. Ripetiamo inoltre che l'utilizzo del database permette a tutti gli stakeholder di accedere e usare il nostro sistema in qualsiasi momento e su qualsiasi piattaforma. Al nostro database per gestire le nuove funzionalità sono state aggiunte queste due nuove tabelle:



gymforyou consultazione	
#	id : int(255)
✉	utente : varchar(255)
✉	dottore : varchar(255)
✉	trainer : varchar(255)
✉	categoria : varchar(255)
✉	domanda : varchar(255)
✉	risposta : varchar(255)
#	voto : int(11)

gymforyou trainer	
#	email : varchar(255)
✉	nome : varchar(255)
✉	cognome : varchar(255)
#	voto : float

Figura 20 Database aggiornato

Le nuove operazioni che possono essere fatte sul nostro database sono le seguenti:

- 1) Visualizzazione medici e trainer;
- 2) Invio domanda;
- 3) Invio risposta;
- 4) Visualizzazione domande e risposte;
- 5) Invio valutazione relativa alla risposta;
- 6) Invio valutazione per il singolo attore.

Tutte le operazioni sopra indicate sono presenti all'interno della classe database, contenuta nel progetto **GymForYou\_server**.

## 20.3 Diagrammi degli stati.

### Diagramma degli stati: Invio domande.

L'utente deve aver effettuato il login. Nella pagina dedicata l'utente potrà scegliere una delle 5 categoria messe a sua disposizione, dovrà poi scegliere a chi inviare la domanda e infine inviarla.

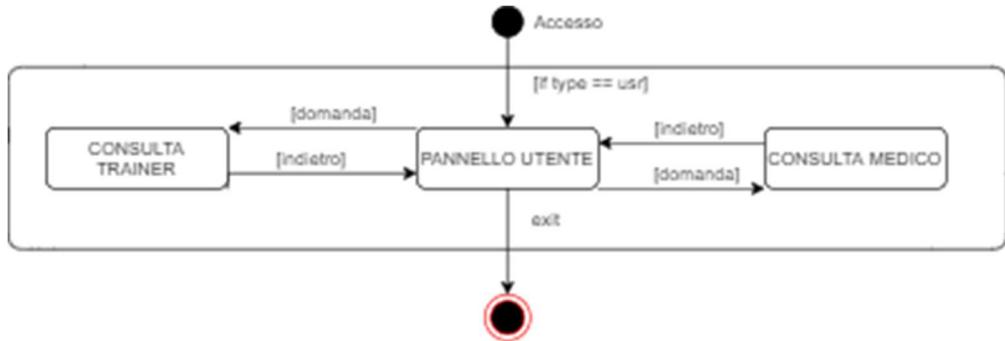


Figura 20.1 State diagram invio domande.

### Diagramma degli stati: Invio risposte.

Il medico o il trainer dopo aver effettuato l'accesso, nell'apposita sezione “controlla domande” potranno visualizzare tutte le domande che gli sono state poste dagli utenti. Dopo avere scelto la domanda dovranno rispondere.

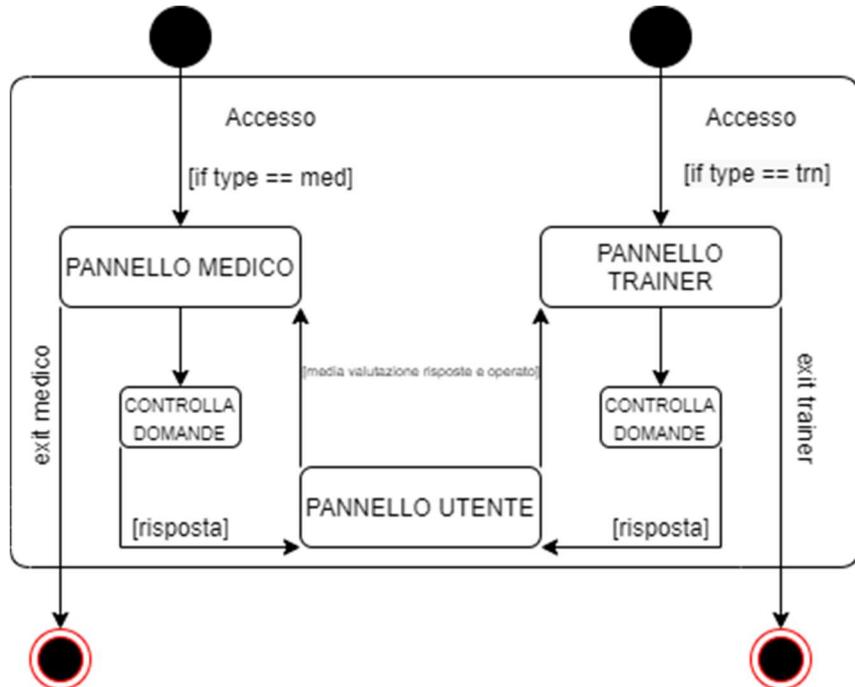


Figura 20.2 State diagram invio risposte.

### Diagramma degli stati: Valutazione risposte e dipendenti.

Se l'utente ha ricevuto una risposta può valutare la risposta ricevuta. In un altro pannello dedicato l'utente potrà valutare i medici e i trainer che fanno parte di *Gym For You*.

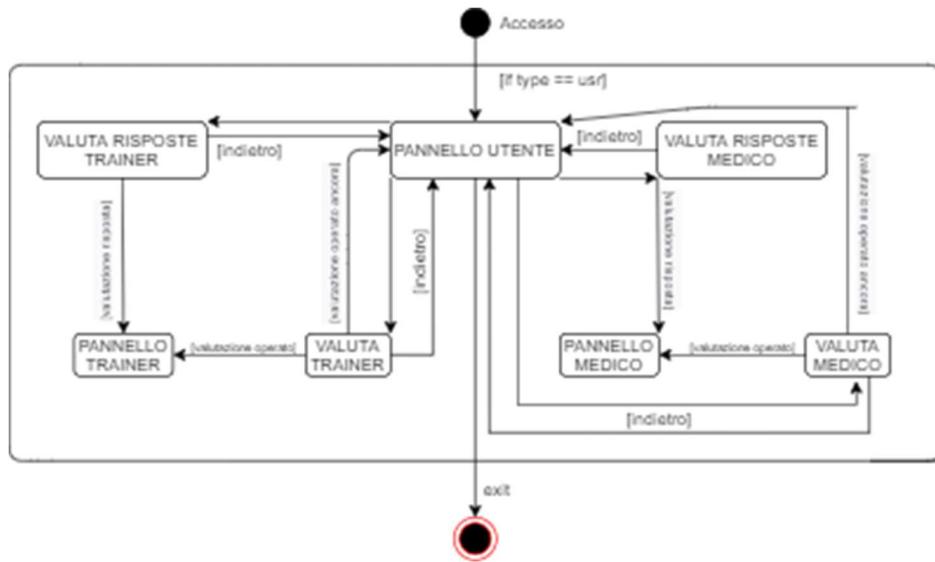


Figura 20.3 State diagram valutazione.

### Diagramma degli stati: Gestione accesso.

In figura 20.4 possiamo notare come sia cambiato il diagramma degli stati relativo all'accesso.

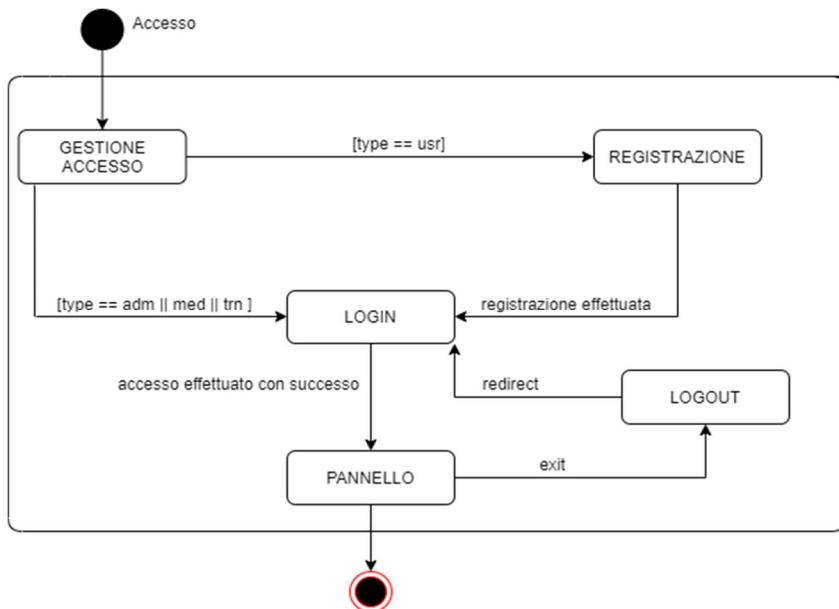


Figura 20.4 State diagram gestione accesso.

## Diagramma degli stati: Formula di Brzycki.

Dopo aver effettuato l'accesso l'utente avrà a sua disposizione un frame grazie al quale l'utente potrà inserire i valori richiesti dalla formula di Brzycki e sempre nello stesso frame visualizzerà il risultato.

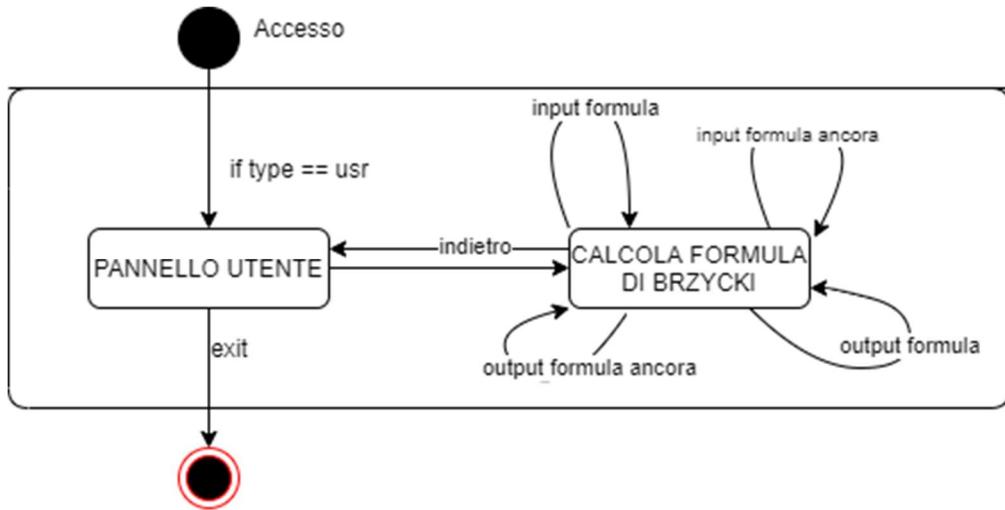


Figura 20.5 State diagram formula di Brzycki.

## 21. Descrizione delle GUI.

### Nuovo pannello utente.

In questo nuovo pannello (figura 21) sono stati aggiunti due nuovi buttoni che permettono all'utente di visualizzare i nuovi frame creati per gestire le funzionalità aggiunte in questo secondo prototipo.

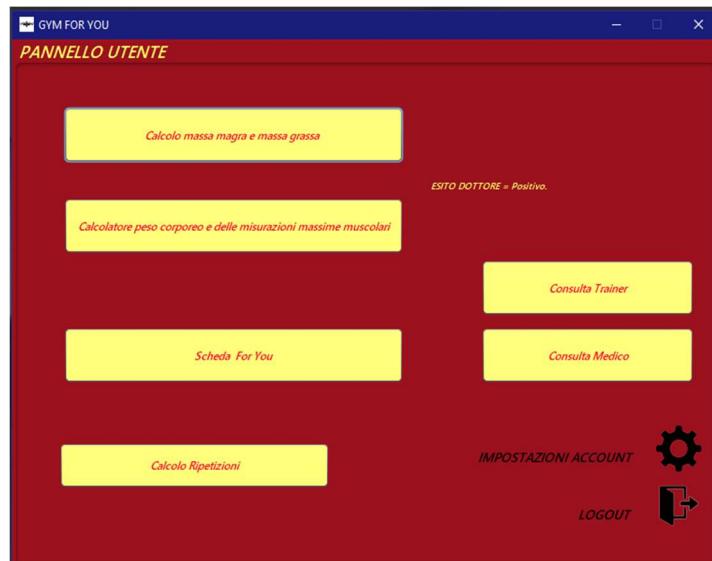


Figura 21 Pannello utente aggiornato.

## Frame scelta operazioni.

In questa interfaccia (figura 21.1) l'utente ha davanti a se tre bottoni che rappresentano le funzionalità di cui l'utente può servirsi. L'interfaccia per la scelta delle operazioni in cui è coinvolto il medico è uguale.



Figura 21.1 Scelta operazioni.

## Frame invia domanda.

L'utente trova davanti a se un insieme di *radioButton* che permetteranno di scegliere la categoria della domanda. A destra abbiamo predisposto una *jTable* con all'interno tutti i trainer o i medici, l'utente cliccando su una riga di tale tabella sceglierà a chi inviare la domanda. In basso troviamo una *textArea* che servirà per scrivere la domanda al suo interno. Infine cliccando sul bottone “invia domanda” verranno attivate tutte le funzioni che controlleranno se è stata scelta la categoria, il dipendente e infine se la domanda non supera 250 caratteri. Se il controllo verrà superato la domanda verrà inviata. Figura 21.2.

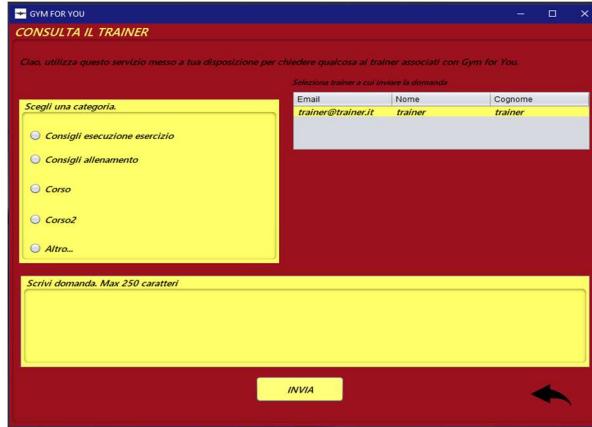


Figura 21.2 Frame invio domanda.

### Frame visualizza risposte.

Anche qui abbiamo una *jTable* con tutte le domande inviate dall'utente. Cliccando su una riga di questa tabella apparirà un nuovo frame (figura 21.3) che mostrerà la domanda e la risposta per esteso. In basso l'utente troverà un *jSpinner* impostato da 0 a 5 che gli permetterà di valutare la risposta ricevuta. Il *jSpinner* apparirà solamente se l'utente non ha valutato la risposta. In basso troviamo un bottone che una volta premuto attiverà tutti i metodi che serviranno ad inviare la valutazione. Figura(21.4)



Figura 21.3 Visualizza risposte.

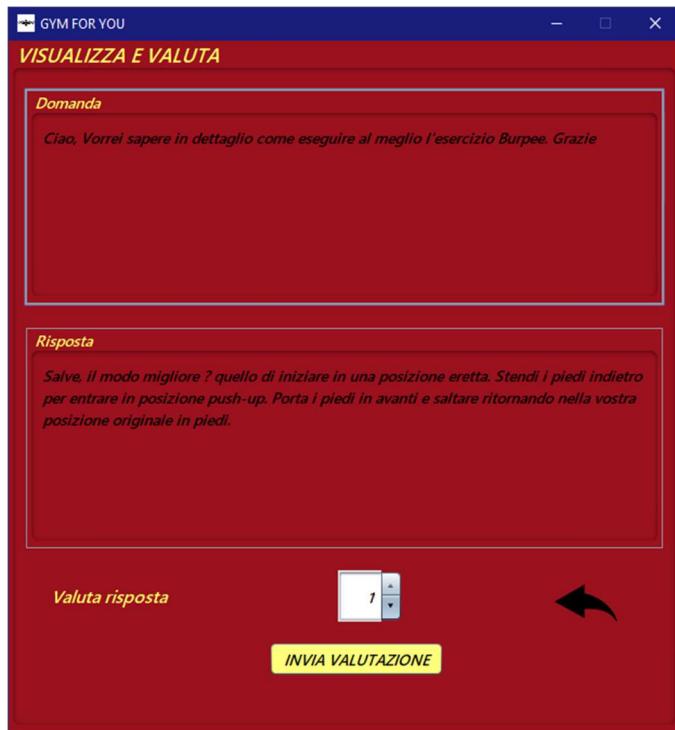


Figura 21.4 Valuta risposte.

### Frame valuta medico o trainer.

In base alla scelta che l'utente ha fatto nel suo pannello personale potrà valutare i medici o i trainer. In questo frame (figura 21.5) l'utente troverà davanti a sé una *jTable* e cliccando su una riga sceglierà il dipendente da valutare. La valutazione verrà espressa tramite uno *jSpinner* impostato da 0 a 5. Premendo il bottone “invia valutazione” la valutazione verrà inviata. A differenza delle domande l'utente potrà valutare quante volte vuole un singolo dipendente.

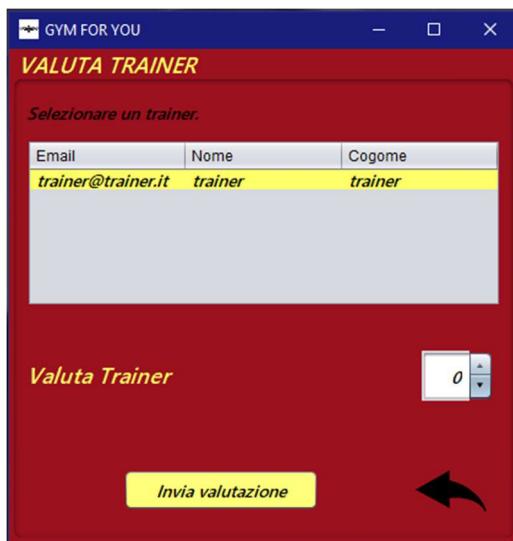


Figura 3Valuta operato.

## Nuovo pannello medico.

In questa nuovo pannello (figura 21.6) è stato aggiunta una nuova sezione chiamata “controlla domande”. Abbiamo aggiunto una *jLabel* che servirà a visualizzare la valutazione media calcolata in base ai voti dati dagli utenti.



Figura 21.6 Pannello medico aggiornato.

## Frame visualizza domande medico.

Il medico troverà davanti a se una *jTable* con tutte le domande che gli sono state inviate, abbiamo deciso di mantenere anche le domande a cui il medico ha risposto per fare in modo che sia presente uno storico delle conversazioni (figura 21.7). In basso a sinistra è presente una *jLabel* che indica la valutazione media delle risposte date dal medico in questione. Cliccando su una riga della tabella il medico visualizzerà un nuovo frame (figura) con all'interno la domanda per esteso e una *textArea* in cui dovrà essere scritta la risposta. Cliccando sul bottone “invia risposta” se la lunghezza dei caratteri (250) è stata rispettata la risposta verrà inviata.



Figura 21.7 Visualizza domande.

### Pannello trainer.

La grafica e le funzionalità sono uguali a quelle descritte nel paragrafo “nuovo pannello medico” (vedi sopra). L’unica differenza è data dall’attore che utilizza tali funzionalità.



Figura 21.8 Pannello trainer.

### Frame visualizza domande trainer.

La grafica e le funzionalità sono uguali a quelle descritte nel paragrafo “visulizza domande medico” (vedi sopra). L’unica differenza è data dall’attore che utilizza tali funzionalità.

**N.B.** Le interfacce e le funzionalità del login sono rimaste uguali a quelle descritte a pagina 64 pertanto si ripercuotono anche sul **trainer**.



Figura 21.9 Visualizza domande trainer.

### Frame formula di Brzycki.

In questo nuovo frame come si può vedere in figura 21.10, abbiamo predisposto 4 campi d'inserimento in cui l'utente dovrà inserire gli input richiesti. In seguito al click sul bottone *Calcola* nella parte destra del nostro frame appariranno i risultati ottenuti dal calcolo. Nel caso in cui gli input risultano errati il nostro sistema è predisposto per far apparire dei pop-up che avvisano dell'errore commesso.

Figura 21.10 Calcolo formula.

## 22. Quarta attività: IMPLEMENTAZIONE.

### 22.1 Diagramma dei componenti client-server (aggiornato).

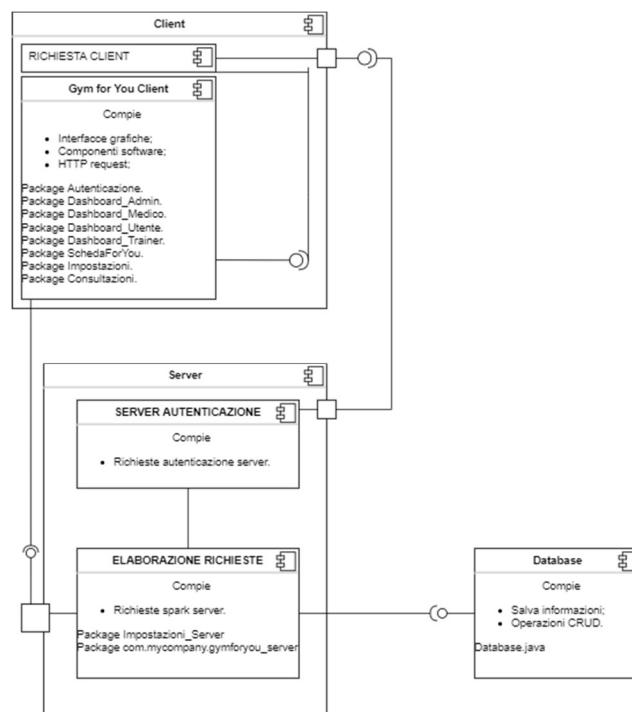


Figura 22 Component diagram client-server aggiornato

## 22.2 Diagramma dei componenti: Gym for you client (aggiornato).

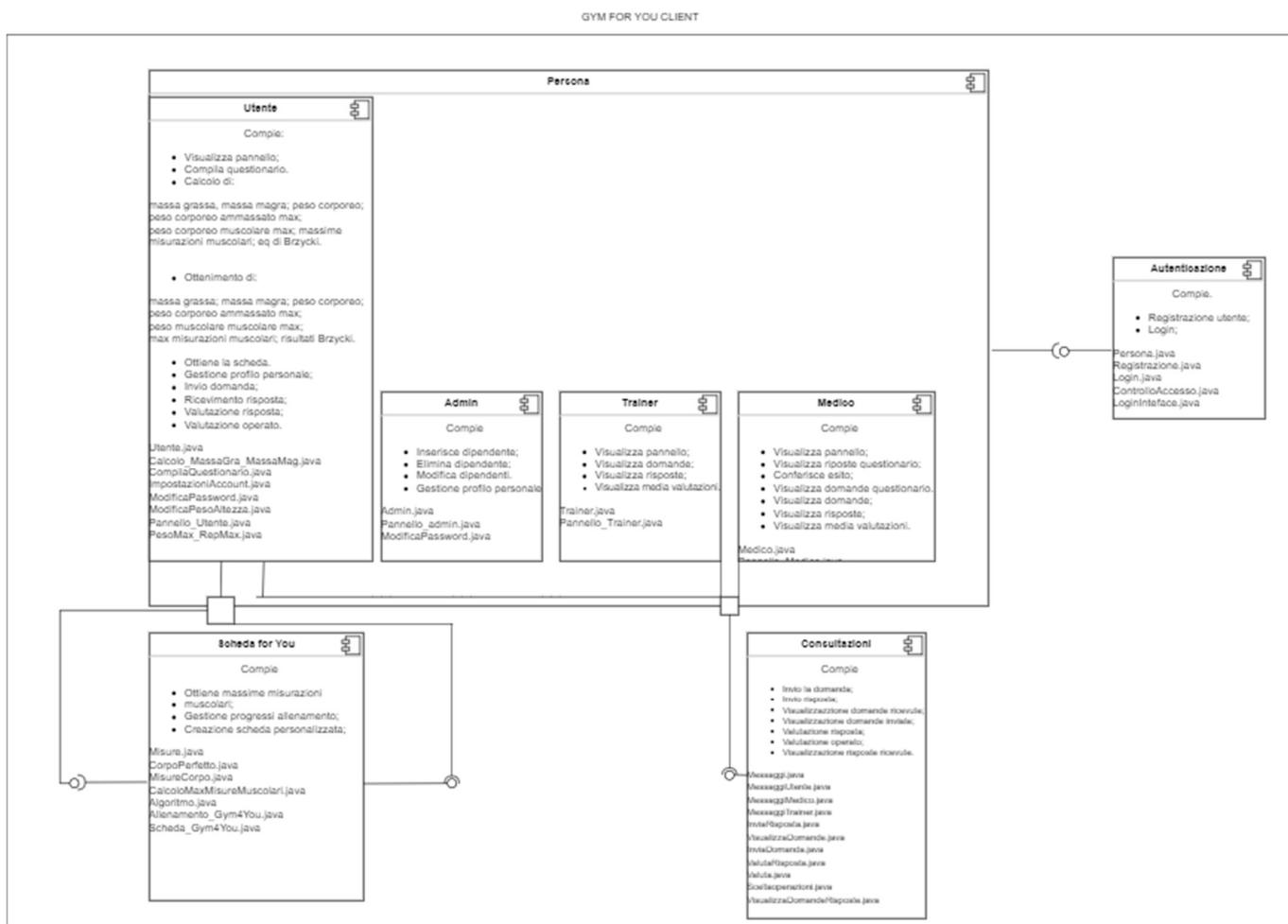


Figura 22.1 Component diagram Gym for you client aggiornato

## 23. Quinta attività: testing.

Anche in questo secondo prototipo per effettuare il testing ci siamo serviti della metodologia unit testing.

### Invio domande.

- **Visualizzazione medici:** il team ha testato che i medici vengano presentati all'utente in modo corretto senza problemi grafici e ciò accade. Il team ha poi più volte stressato il sistema per verificare che i dati vengano prelevati in modo corretto e dopo diverse prove i dati restituiti sono sempre quelli attesi.
- **Visualizzazione trainer:** essendo una funzionalità identica a quella descritta in precedenza sono stati condotti gli stessi test e i risultati sono sempre stati quelli attesi senza nessun tipo di problematica.
- **Invio domanda:** il team ha più volte provato ad inviare al server delle domande contenenti dei caratteri non accettabili e il software si è

sempre comportato nella maniera aspettata ovvero non accettando tali domande. Il team ha poi testato che non vengano accettate delle domande con un numero di caratteri superiore a quello consentito e anche in questo caso il software ha prontamente rifiutato tali domande. Il sistema è predisposto per non accettare delle domande in cui non è stata specificata la *categoria* e/o il destinatario, perciò il team ha più volte testato quest'aspetto ricevendo sempre dei risultati positivi da parte del software.

- **Visualizzazione domande inviate e risposte:** anche qui il team non ha riscontrato nessun tipo di problema né dal punto di vista grafico e tantomeno dal punto di vista funzionale, infatti sia le domande che le risposte vengono prelevate e presentate nel modo corretto.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Invio risposte.

- **Visualizzazione domande ricevute:** il team in questo caso ha dovuto testare che le domande ricevute vengano prelevate e visualizzate in modo corretto. Dopo una attenta e meticolosa analisi si è constatato che il software risponde in modo corretto sia nella parte dedicata al medico che in quella dedicata al trainer.
- **Invio risposta:** la stessa metodologia di test applicata per l'invio della domanda è stata applicata anche in questo caso e il sistema ha risposto nel modo corretto.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Valutazione risposte e dipendenti.

- **Visualizzazione medici e trainer:** sono stati condotti gli stessi test della sezione “visualizzazione medici” e “visualizzazione trainer” del paragrafo “invio domanda”. Allo stesso modo i risultati ottenuti sono corretti.
- **Valutazione medico e trainer:** il team ha condotto dei test per vedere che la valutazione viene inviata e immagazzinata con successo, e la cosa accade senza problematiche. Il team ha poi testato che all'interno dello spinner non vengano inseriti dei caratteri diversi dai numeri oppure che non venga superato il range di valutazione prefissato (da 0 a 5) grazie alle caratteristiche intrinseche dell'oggetto *jSpinner* utilizzato, la cosa non accade.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Gestione accesso.

- **Login:** qui il team ha provato più volte a immettere credenziali errate e in tutti i casi il sistema ha rispettato i requisiti. Il team ha poi provato ad entrare nel sistema con credenziali non registrate e il sistema ha rispettato i requisiti non consentendo l'accesso.
- **Logout:** qui il team ha voluto testare che nel momento del logout non si veniva reindirizzati in pagine errate, ciò non accade quindi il sistema rispetta i requisiti.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Formula di Brzycki.

- **Inserimento input:** nell'inserimento l'unico problema poteva presentarsi era quello in cui l'utente immetteva degli input errati. Per ovviare al problema il team ha creato dei meccanismi di controllo che evitano il presentarsi del problema.
- **Visualizzazione risultati:** la visualizzazione dei risultati dipende da ciò che viene inserito come input, considerando che gli input inseriti vengono controllati se sono corretti il risultato verrà visualizzato altrimenti apparirà un pop-up con l'errore.

## 24. Verifica.

### Test 1: login.

Test	Input	Output
Inserimento valori email password	Credenziali corrette.	Accesso effettuato.
	Credenziali errate.	Accesso rifiutato.
	Campi vuoti.	Accesso rifiutato.

### Test 2: invio domanda e invio risposta.

Test	Input	Output
Inserimento di caratteri non consentiti.	Caratteri speciali.	Domanda non inviata.
	Campi vuoti.	Domanda non inviata.
Superamento dei caratteri consentiti.	Domanda con più di 250 caratteri.	Domanda non inviata.

### Test 3: invio valutazione risposte e invio valutazione medici e trainer.

Test	Input	Output
Inserimento di valori non consentiti.	Caratteri speciali.	Valutazione non inviata.
	Numeri superiori a 5.	Valutazione non inviata.
	Numeri negativi.	Valutazione non inviata.
	Numeri decimali.	Valutazione non inviata.

### Test 4: formula bi Brzycki.

Test	Input	Output
Inserimento di valori non consentiti.	Caratteri speciali.	Impossibile visualizzare risultato.
	Ripetizioni da raggiungere > 10.	Impossibile visualizzare risultato.
	Numeri negativi.	Impossibile visualizzare risultato
	Peso che si vuole alzare minore di quello che si riesce ad alzare.	Impossibile visualizzare risultato

## 25. Ultima attività: DEPLOYMENT.

### Diagramma di deployment.

Il diagramma di deployment rimane identico a quello presentato a pagina 83.

## 26. Diagramma delle fasi-workflow RUP.

Una semplice enumerazione di tutti le attività e artefatti non costituisce del tutto un processo. Abbiamo bisogno di un modo per descrivere sequenze significative di attività che producono risultati di valore e per mostrare le interazioni tra i lavoratori. Un flusso di lavoro (workflow) è una sequenza di attività che produce un risultato di valore osservabile.

Non è sempre possibile o pratico rappresentare tutte le dipendenze tra le attività. Spesso, due attività sono più strettamente intrecciate di quanto mostrato, soprattutto quando coinvolgono lo stesso lavoratore o lo stesso individuo. Le persone non sono macchine e il flusso di lavoro non può essere interpretato letteralmente come un programma che le persone devono seguire esattamente e meccanicamente.

UML per rappresentare i flussi di lavoro mette a disposizione il diagramma dei workflow (figura 26). Il suddetto diagramma rappresenta sull'asse verticale le attività, sull'asse orizzontale le 4 fasi che compongono il modello RUP e soprattutto il **tempo**. In basso troviamo le due iterazioni che abbiamo

prodotto finora. Le parti colorate indicano il tempo impiegato per ciascuna attività e fase in base all'iterazione.

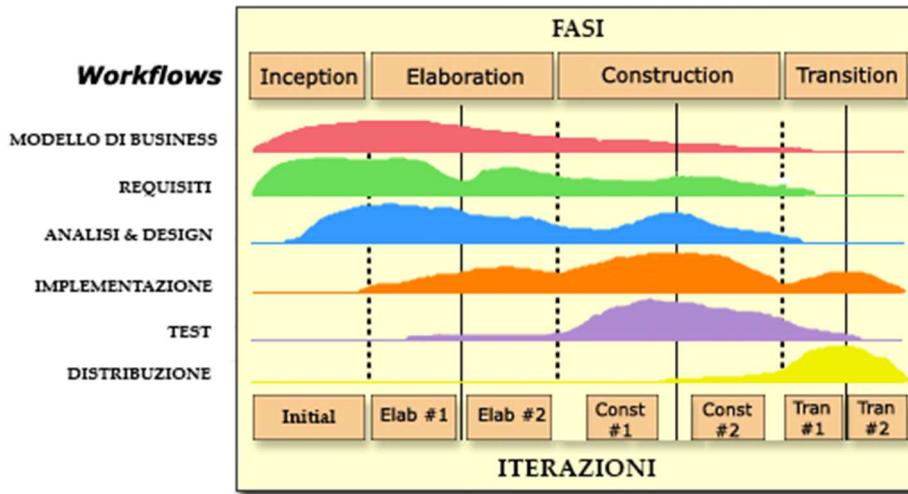


Figura 26 Workflow diagram.

## TERZO PROTOTIPO.

Dopo aver presentato il secondo prototipo al nostro cliente ed avere ricevuto dei feedback positivi, il team è partito con lo sviluppo delle nuove funzionalità. Le nuove funzionalità consistono nel permettere ai nostri utenti la creazione di un diario di allenamento personalizzato. Gli utenti dovranno personalizzare e parametrizzare il proprio diario. Gli utenti nel momento della creazione del diario di allenamento potranno scegliere se gli esercizi che dovranno svolgere sono degli esercizi standard oppure esercizi proveniente dai corsi che sono presenti in *Gym for You*. Il diario verrà poi inviato ai nostri allenatori che potranno visionarlo. Dopo essere stato visionato entrerà in gioco l'automazione, ovvero un algoritmo progettato dal team con l'utilizzo del linguaggio Java che in modo completamente automatico andrà a generare una scheda totalmente in linea con il diario creato dall'utente. Nel caso in cui i trainer reputano che nella scheda creata in modo automatico ci sia un esercizio che va modificato hanno la possibilità di farlo. Per quanto riguarda i corsi nel caso in cui se ne voglia aggiungere uno, eliminarlo, oppure si voglia modificare un esercizio di un corso già presente, queste azioni verranno svolte dall'admin del sistema.

## 27. Prima fase INCEPTION.

### 27.1.1 ANALISI DEI REQUISITI.

Descriviamo i nuovi requisiti del sistema che si aggiungono a quelli descritti in precedenza.

### 27.1.2 Descrizione stakeholder.

Oltre agli stakeholder già descritti, qui andremo a descrivere quelle che saranno le nuove funzionalità che si aggiungono a quelle già descritte:

- **Utente:** egli avrà il compito di creare il diario personalizzato, che coinvolge nello scegliere le parti del corpo da allenare, la modalità di allenamento (dimagrimento o aumento), i giorni in cui egli si vuole allenare, i momenti della giornata in cui vuole allenarsi (mattina, pomeriggio, sera) e infine l'intensità di allenamento (leggero, medio e pesante) associata ad ogni giorno e momento.
- **Trainer:** visiterà il diario creato dall'utente e creerà in modo automatico la scheda. Se ritiene che la scheda generata in modo automatico non è completamente corretta potrà modificarla.
- **Admin:** potrà aggiungere un nuovo corso e i relativi esercizi, modificare gli esercizi di un corso già creato oppure aggiungerne di nuovi e infine eliminare un corso.

### 27.1.3 Specifica dei requisiti.

Oltre ai requisiti già descritti aggiungiamo i seguenti requisiti.

### 27.1.4 Requisiti funzionali.

#### **Utente può:**

- Chiedere una scheda personalizzata;
- Chiedere più volte una scheda personalizzata;
- Inserire i dati del diario;
- Modificare i dati del diario;
- Ottenere e visualizzare la propria scheda;

#### **Trainer può:**

- Visualizzare il diario dell'utente;
- Creare una scheda in modo automatico;
- Modificare gli esercizi di una scheda.

#### **Admin può:**

- Inserire un nuovo corso;
- Inserire gli esercizi per il nuovo corso;
- Modificare gli esercizi per i corsi esistenti;
- Eliminare un corso.

### 27.1.5 Requisiti non funzionali.

Tali requisiti rimangono uguali a quelli presenati nel paragrafo 5.1.5.

## 27.2 Indagine sul sistema.

### 27.2.1 Analisi del dominio.

Il dominio applicativo del nostro software è immutato da quello descritto in precedenza.

Per comprendere al meglio il nuovo dominio applicativo il team ha ampliato il context diagram. Ecco il nuovo diagramma:

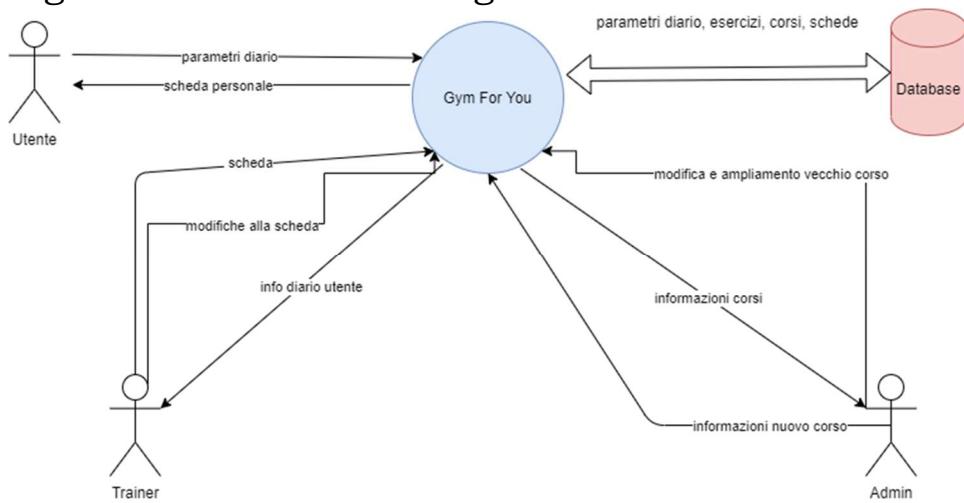


Figura 27 Context diagram.

### 27.2.2 Studio fattibilità.

Per lo studio di fattibilità si rimanda la paragrafo 5.2.2 in quanto le caratteristiche fondamentali del sistema sono rimaste invariate.

## 27.3 Analisi dei rischi.

### 27.3.1 Rischi di progetto.

I rischi che potrebbero capitare nel nostro sistema dipendono molto dagli input inseriti dagli attori coinvolti, il team avendo analizzato questo tipo di rischio a predisposto tutti i controlli necessari per fare in modo che tutti gli input inseriti siano corretti, nel caso contrario verranno generati degli eventi che comunicheranno e rifiuteranno gli input errati. Anche nel caso del diario bisogna stare attenti a non stressare troppo una parte del corpo, perciò potrebbe capitare che nella modifica della scheda personale un trainer vada ad aggiungere troppi esercizi per una parte del corpo. Altri rischi che potrebbero capitare riguardano la stabilità della rete, avendo

scelto un tipo di architettura client-server il funzionamento del nostro sistema dipende in parte dalla stabilità delle connessioni.

### 27.3.2 Rischi di prodotto.

I rischi di prodotto rimangono immutati a quelli presenti nel paragrafo 5.3.2 perciò si rimanda al suddetto paragrafo.

### 27.3.3 Tabella valutazione rischi.

RISCHI	VALUTAZIONE RISCHIO	PROBABILITA' CHE ACCADA	CONSEGUENZE	METODO DI RISOLUZIONE
Impossibilità di inviare il diario.	Poco probabile.	Media.	L'utente non riesce ad inviare il proprio diario e non riceverà la sua scheda.	Utilizzo di una delle migliori tecnologie lato server (Spark) e progettazione di tutti i controlli per quanto riguarda l'inserimento degli input. In merito ai problemi di connessione il team non può garantire nessun metodo risolutivo.
Impossibilità di consultare il diario da parte del trainer.	Poco probabile.	Media.	Il trainer non riesce a leggere il diario il diario degli utenti.	Utilizzo di una delle migliori tecnologie lato server (Spark) e progettazione di tutti i controlli per quanto riguarda l'inserimento degli input. In merito ai problemi di connessione il team non può garantire nessun metodo risolutivo.
Non viene generata la scheda in modo automatico.	Poco probabile.	Molto bassa.	Il trainer non riesce a generare la scheda.	L'algoritmo creato per la generazione della scheda è stato progettato con attenzione maniacale perciò il team garantisce sempre le massime prestazioni e il minimo rischio.
Non modifica di un esercizio da parte del trainer.	Poco probabile.	Molto bassa.	Il trainer non riesce a modificare un esercizio presente all'interno della scheda generata.	Utilizzo di una delle migliori tecnologie lato server (Spark) e progettazione di tutti i controlli per quanto riguarda l'inserimento degli input. In merito ai problemi di connessione il team non può garantire nessun metodo risolutivo.
Scheda personale non ottenibile e/o non visualizzabile.	Alto.	Media.	L'utente non riesce ad allenarsi.	Utilizzo di una delle migliori tecnologie lato server (Spark). In merito ai problemi di connessione che potrebbero non far vedere la scheda, il team non può garantire nessun metodo risolutivo.
Impossibilità di gestire i corsi.	Poco probabile.	Media.	L'admin non riesce a gestire i corsi.	Utilizzo di una delle migliori tecnologie lato server (Spark) e progettazione di tutti i controlli per quanto riguarda l'inserimento degli input. In merito ai problemi di connessione il team non può garantire nessun metodo risolutivo.
Impossibilità di modificare gli esercizi di un corso.	Poco probabile.	Media.	L'admin non riesce a modificare gli esercizi di un corso.	Utilizzo di una delle migliori tecnologie lato server (Spark) e progettazione di tutti i controlli per quanto riguarda l'inserimento degli input. In merito ai problemi di connessione il team non può garantire nessun metodo risolutivo.
Superamento del numero di esercizi per una parte del corpo	Poco probabile.	Bassa.	Il trainer aggiunge troppi esercizi per una parte del corpo.	Il team ha creato un metodo che nel caso in cui si debba modificare un esercizio per una determinata parte del corpo può essere modificato

## SECONDO PROTOTIPO.

### 28. Seconda fase ELABORATION.

#### 28.1.1 Prima attività requisiti.

Oltre ai requisiti presentati nei precedenti prototipi aggiungiamo i seguenti e come già successo essi vengono divisi in 4 macrocategorie.

- 1) Invio dati e visualizza scheda.
- 2) Creazione scheda.
- 3) Modifica scheda.
- 4) Gestione corsi.

#### 28.1.2 Requisiti funzionali.

##### **Requisito A: Invio dati e visualizza scheda.**

- ID A.A.A:** Scelta parte del corpo.
- ID A.A.B:** Scelta modalità di allenamento.
- ID A.A.C:** Scelta giorni.
- ID A.A.D:** Scelta momenti della giornata.
- ID A.A.E:** Scelta intensità allenamento.
- ID A.A.F:** Scelta trainer.
- ID A.A.G:** Scelta scheda personale o corso.
- ID A.A.H:** Modifica input.
- ID A.A.I :** Visualizzazione scheda.

##### **Requisito B: Creazione scheda.**

- ID B.A.A:** Visualizzazione parametri diario.
- ID B.A.B:** Creazione scheda.

##### **Requisito C: Modifica scheda.**

- ID C.A.A:** Visualizzazione scheda.
- ID C.A.B:** Modifica di uno o più esercizi.

##### **Requisito D: Gestione corsi.**

**ID D.A.A:** Inserimento nuovo corso.

**ID D.A.B:** Inserimento esercizi nuovo corso.

**ID D.A.C:** Modifica esercizi vecchio corso.

**ID D.A.D:** Ampliamento esercizi vecchio corso.

**ID D.A.E:** Eliminazione corso ed esercizi.

**ID D.A.F:** Visualizzazione corsi.

### 28.1.3 Requisiti non funzionali

- 1) Utilizzo di una database per la memorizzazione delle domande e delle risposte.
- 2) Trattamento dei dati nel maggior rispetto della privacy e rispetto delle leggi vigenti a riguardo.
- 3) Compatibilità con diverse piattaforme.

### 28.1.4 Documento SRS.

NOME REQUISITO	ID	PRIORITA'	DESCRIZIONE	REQUISITO PADRE	REQUISITO FIGLIO
Invio dati e visualizza scheda.	A	Indispensabile.	Permette all'utente di inviare i dati del proprio diario e di visualizzare la scheda.		A.A.A;A.A.B A.A.C;A.A.D; A.A.E;A.A.F; A.A.G;A.A.H A.A.I
Scelta parte del corpo.	A.A.A	Indispensabile	Permette di scegliere la parte del corpo da allenare. Le parti del corpo già allenate non possono essere scelte.	A	
Scelta modalità di allenamento.	A.A.B	Indispensabile	Permette di scegliere se allenarsi per dimagrire oppure per aumentare la massa muscolare.	A	
Scelta giorni.	A.A.C	Indispensabile	Permette di scegliere i giorni in cui allenarsi.	A	
Scelta momenti giornata.	A.A.D	Indispensabile	Permette di scegliere se allemarsi di mattina pomeriggio o sera.	A	
Scelta intensità.	A.A.E	Indispensabile	Permette di scegliere se allenarsi in modo leggero, medio o pesante.	A	
Scelta trainer.	A.A.F	Indispensabile	Permette di scegliere il trainer che visionerà la scheda.	A	
Scelta scheda o corso.	A.A.G	Indispensabile	Permette di scegliere esercizi generici oppure provenienti da un corso specifico.	A	

Modifica dati.	A.A.H	Indispensabile	Permette di tornare indietro e modificare gli input inseriti.	A	
Visualizzazione scheda.	A.A.I	Indispensabile	Permette all'utente di visualizzare la scheda.	A	
Creazione scheda.	B.	Indispensabile.	Grazie all'algoritmo creato dal team viene creata una scheda personale per l'utente.		B.A.A; B.A.B
Visualizzazione dati diario.	B.A.A	Sarebbe meglio avere.	Permette al trainer di visualizzare il diario creato dall'utente.	B	
Creazione scheda.	B.A.B	Indispensabile.	Crea la scheda in base all'esigenze dell'utente.	B	
Modifica scheda.	C	Sarebbe meglio avere.	Permette di modificare gli esercizi generati in modo automatico.		C.A.A; C.A.B
Visualizzazione scheda.	C.A.A	Indispensabile.	Permette di visualizzare la scheda creata dall'algoritmo.	C	
Modifica di uno o più esercizi.	C.A.B	Sarebbe meglio avere.	Permette al trainer di sostituire un esercizio con uno diverso.	C	
Gestione corsi.	D	Indispensabile.	Permette la gestione dei corsi che fanno o faranno parte di <i>Gym for You</i> .		D.A.A;D.A.B D.A.C;D.A.D D.A.E;D.A.F
Inserimento nuovo corso.	D.A.A	Sarebbe meglio avere.	Permette di inserire un nuovo corso.	D	
Inserimento esercizi nuovo corso.	D.A.B	Sarebbe meglio avere.	Permette d'inserire esercizi per un corso.	D	
Modifica esercizi vecchio corso.	D.A.C	Sarebbe meglio avere.	Permette di modificare gli esercizi di un corso già esistente.	D	
Ampliamento esercizi vecchio corso.	D.A.D	Sarebbe meglio avere.	Permette di ampliare gli esercizi di un corso già esistente.	D	
Eliminazione corso ed esercizi.	D.A.E	Sarebbe meglio avere.	Permette di eliminare un corso.	D	
Visualizzazione corsi	D.A.F	Sarebbe meglio avere	Permette la visualizzazione di tutti i corsi	D	

## 28.1.5 Diagramma dei casi d'uso.

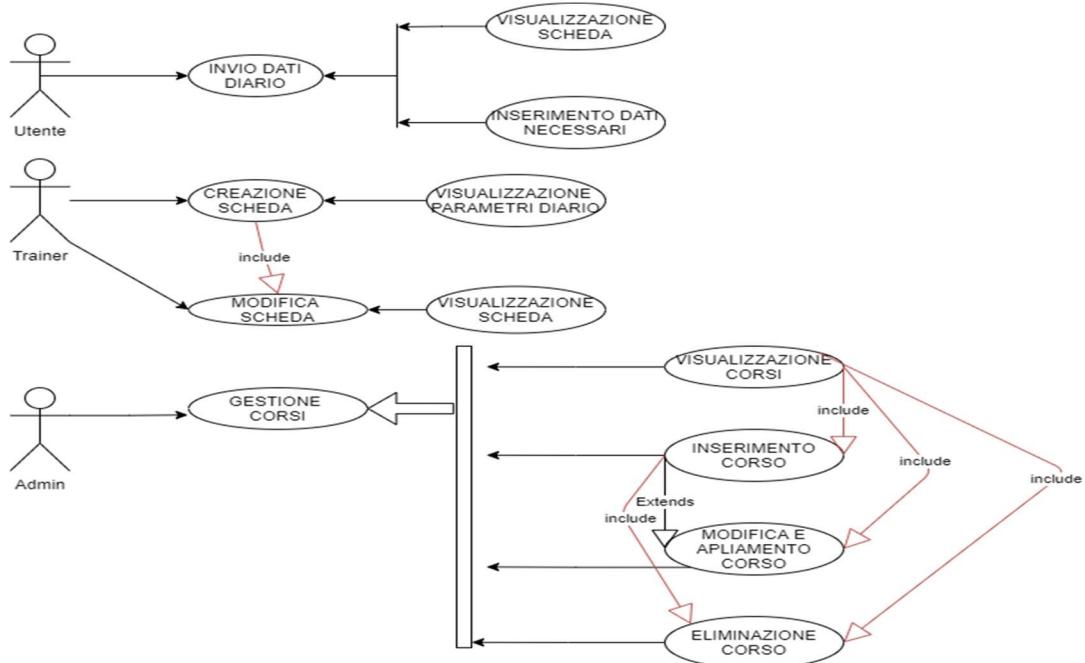


Figura 28 Use case diagram.

## 28.1.6 Descrizione attori.

<b>ATTORE</b>	Utente.
<b>ID</b>	1
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Egli è colui il quale potrà creare il proprio diario.

<b>ATTORE</b>	Trainer.
<b>ID</b>	1
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Ha il compito di avviare la creazione della scheda di allenamento. Se reputa l'allenamento corretto lo deve confermare altrimenti può modificare gli esercizi.

<b>ATTORE</b>	Admin.
<b>ID</b>	1
<b>GENITORE</b>	Nessuno.
<b>RUOLO</b>	Gestisce in maniera completa i corsi.

## 28.1.7 Descrizione casi d'uso.

Caso d'uso	Invio dati e visualizza scheda.
ID	A.
Descrizione	Permette agli utenti di creare il proprio diario. Permette di ottenere la scheda e di usarla per l'allenamento.
Attori	Utente.
Pre-condizioni	Avere effettuato il log-in. Aver inserito gli input in modo corretto.
Flusso principale	Crea il diario, riceve la scheda, si allena.
Scenario secondario.	Errore durante l'inserimento degli input richiesti.
Flusso alternativo.	Non ottiene la scheda, non si può allenare.
Post-condizioni	Accesso al pannello e invio dati scheda.

Caso d'uso	Creazione scheda.
ID	B.
Descrizione	Avvia l'algoritmo che crea la scheda.
Attori	Trainer.
Pre-condizioni	Avere effettuato il log-in da parte del trainer.
Flusso principale	Accede al suo pannello. Crea la scheda.
Scenario secondario.	Errore durante il log-in. Impossibile creare la scheda.
Flusso alternativo.	Non accede al pannello. Non può creare la scheda.
Post-condizioni	Accesso al pannello e creazione scheda.

Caso d'uso	Modifica scheda.
ID	C.
Descrizione	Permette al trainer di modificare uno o più esercizi.
Attori	Trainer.
Pre-condizioni	Avere effettuato il log-in. Aver creato e visualizzato la scheda.
Flusso principale	Accede al suo pannello. Crea e visualizza la scheda.
Scenario secondario.	Errore durante il log-in. Impossibilità di visualizzare e modificare la scheda.
Flusso alternativo.	Non accede al pannello. Non viene modificata la scheda.

Post-condizioni	Accesso al pannello e modifica degli esercizi della scheda.
-----------------	---

Caso d'uso	Gestione corsi.
ID	D.
Descrizione	Permette all'admin inserire, eliminare e modificare uno o più corsi.
Attori	Admin.
Pre-condizioni	Avere effettuato il log-in.
Flusso principale	Accede al suo pannello. E svolge le azioni sui corsi.
Scenario secondario.	Errore durante il log-in. Input errati.
Flusso alternativo.	Non accede al pannello. Non può gestire i corsi.
Post-condizioni	Gestione corretta dei corsi.

## 28.1.8 Diagramma delle attività.

Nei prossimi paragrafi presentiamo i diagrammi di attività relativi al nuovo prototipo. Per evitare la ridondanza di seguito sono riportati i diagrammi relativi alle nuove attività e funzionalità del nostro sistema.

### Diagramma delle attività utente.

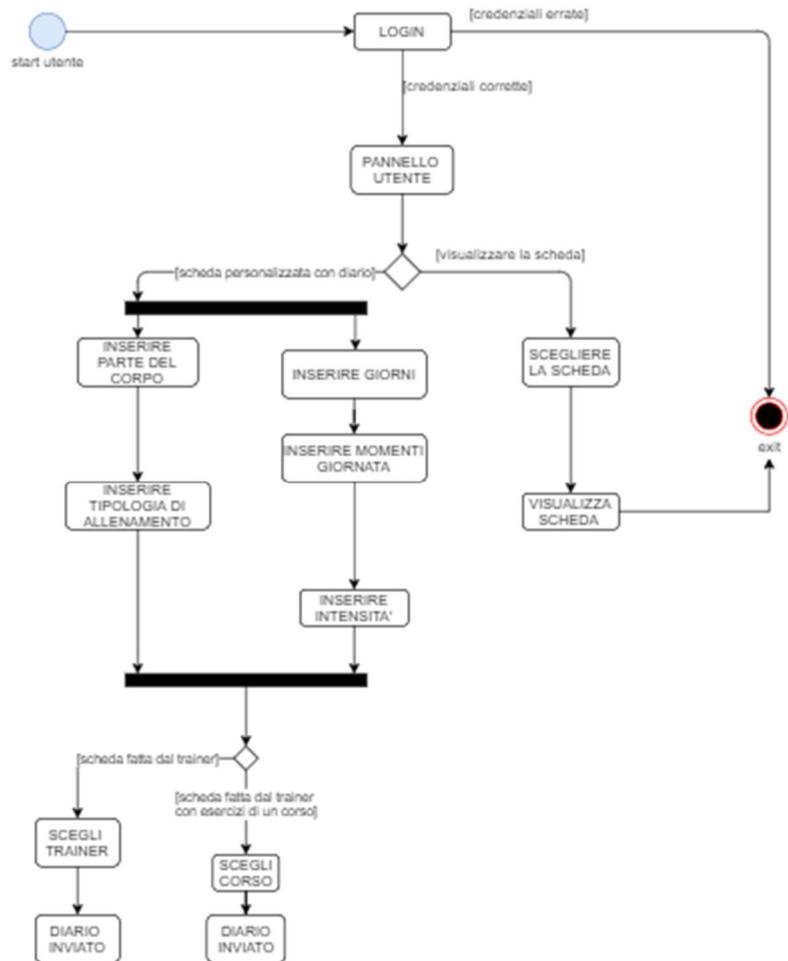


Figura 28.1 Activity diagram Utente

## Diagramma delle attività trainer.

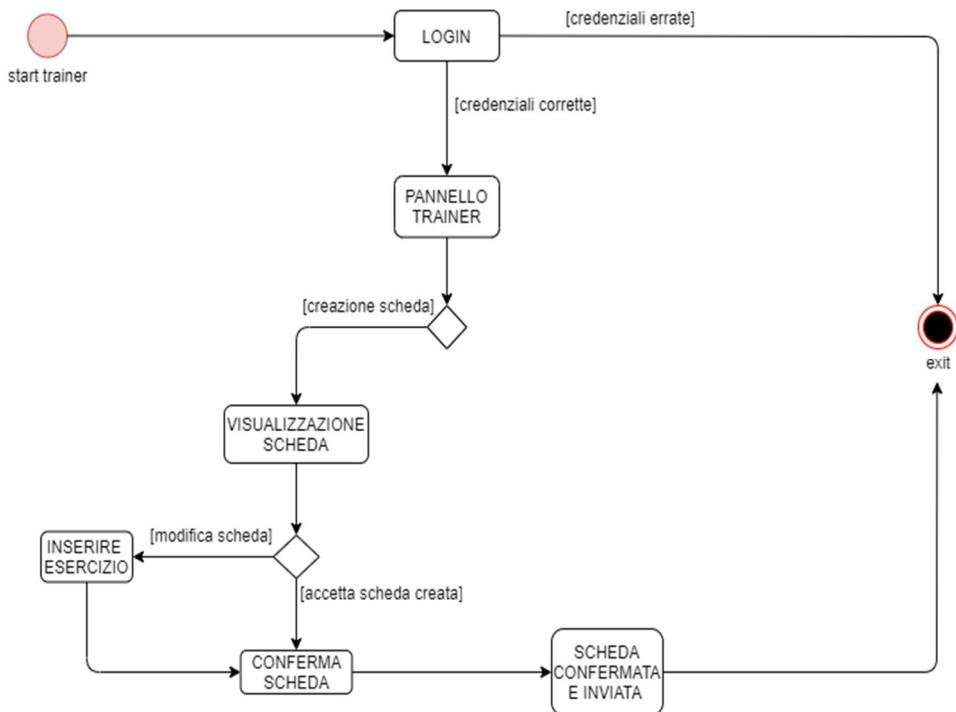


Figura 28.2 Activity diagram trainer.

## Diagramma delle attività admin.

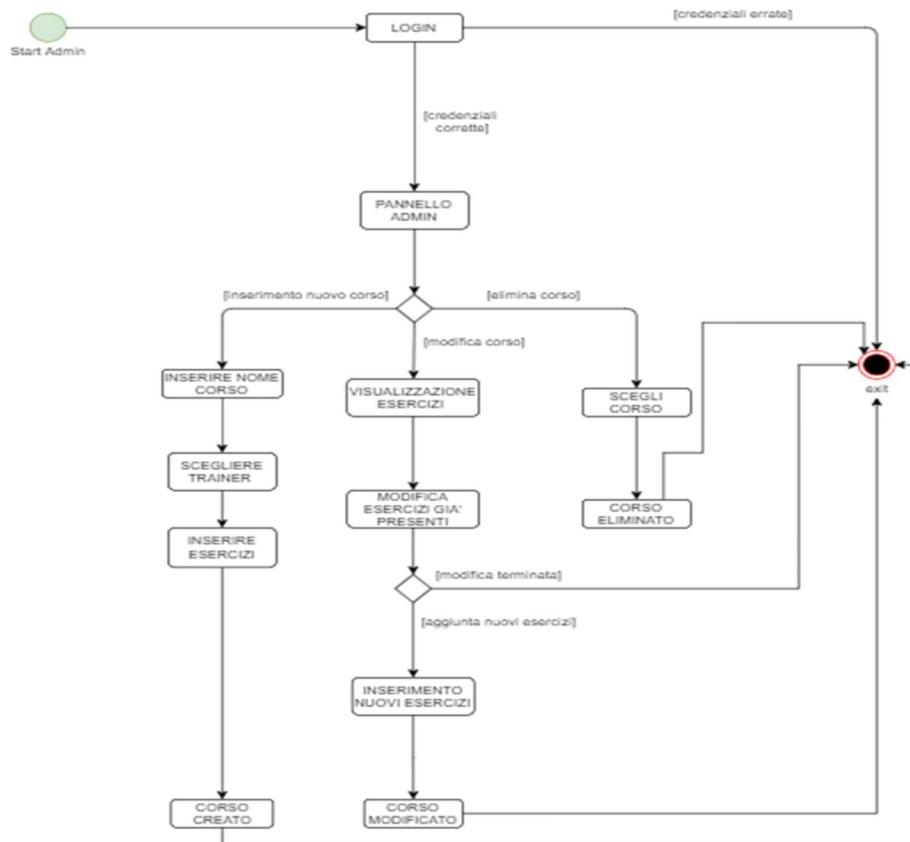


Figura 28.3 Activity diagram admin.

# TERZO PROTOTIPO.

## 29. Seconda attività: ANALISI.

Analisi architetturale.

### 29.1 Diagramma dei package.

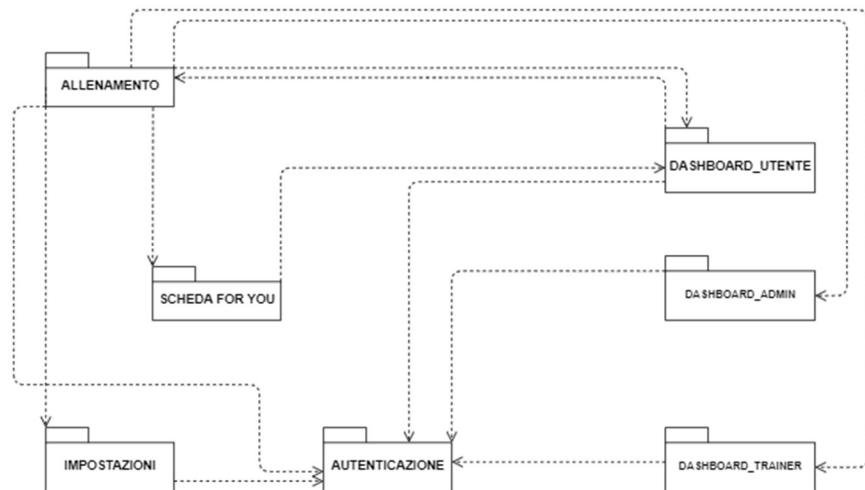


Figura 29 Package diagram.

Individuazione delle classi di analisi.

Oltre alle classi che sono già state descritte in seguito descriviamo le nuove classi che sono state create per questo terzo prototipo. Anche in questo caso per ottenere il diagramma delle classi abbiamo utilizzato il plug-in di Netbeans, EasyUML.

### 29.2 Diagrammi delle classi.

Analisi delle classi: Dashboard trainer.

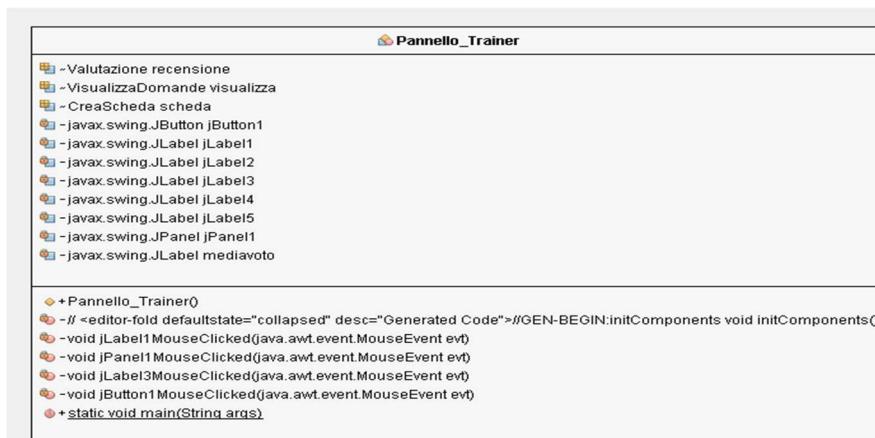


Figura 29.1 Class diagram dashboard trainer.

## Analisi delle classi: Package Allenamento.

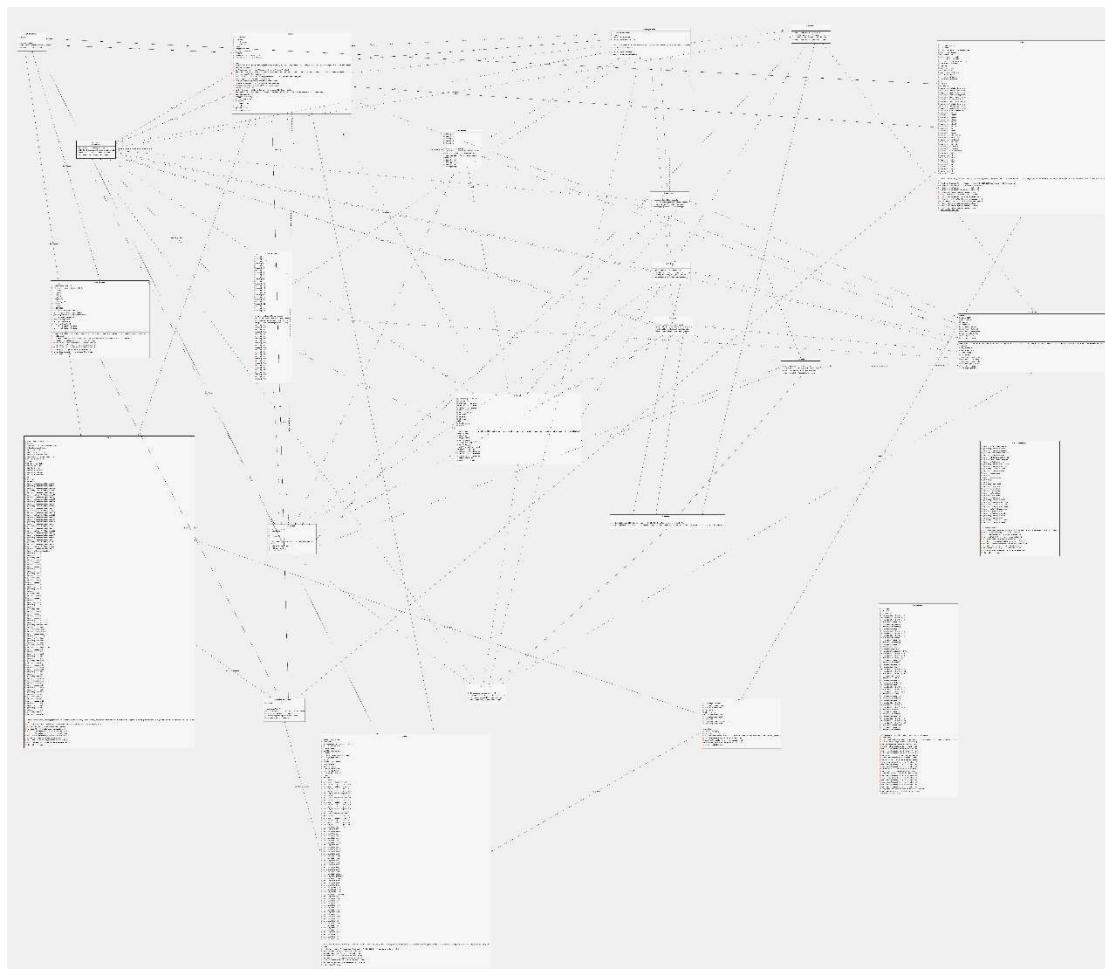


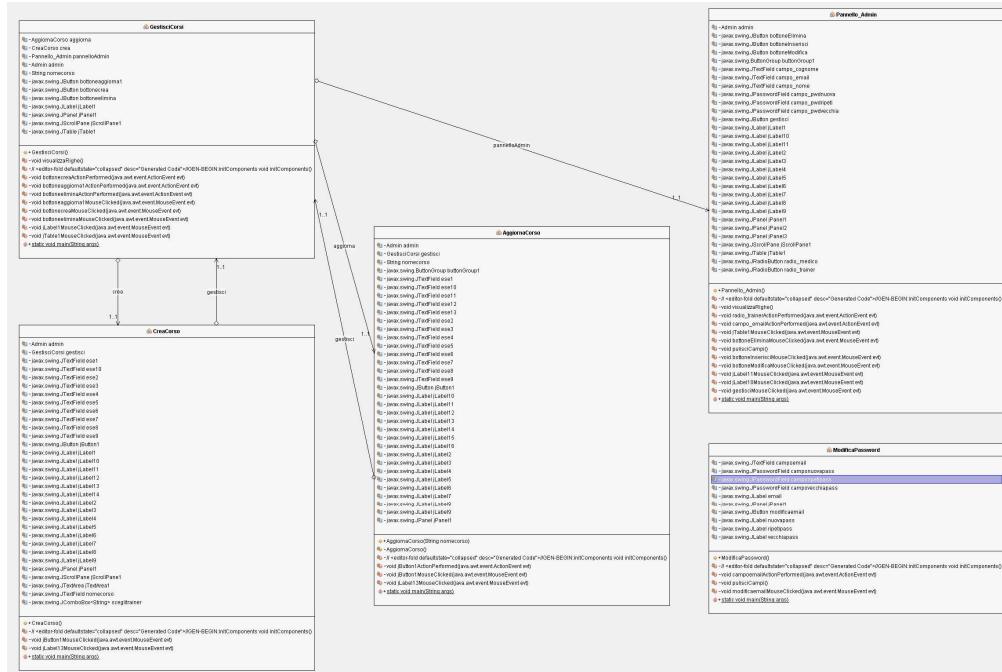
Figura 29.2 Class diagram package scelta

## Analisi delle classi: Dashboard utente.



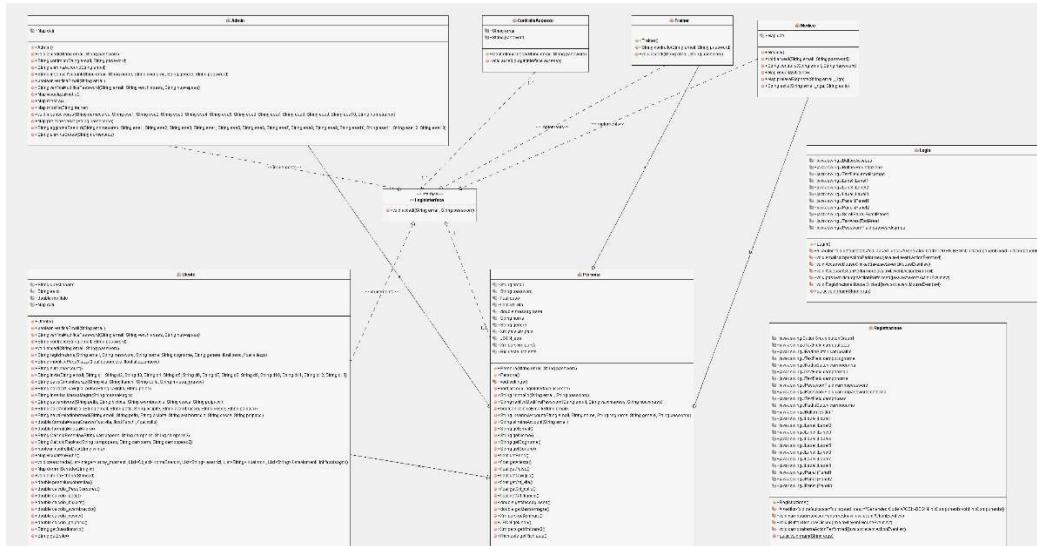
Figura 29.3 Class diagram dashboard utente.

## Analisi delle classi: Dashboard admin.



*Figura 29.4 Class diagram dashboard admin*

## Analisi delle classi:Autenticazione.



*Figura 29.5 Class diagram Autenticazione.*

## Analisi dei casi d'uso.

### 30. Diagrammi di sequenza.

Anche per questo terzo prototipo abbiamo utilizzato l'architettura client-server. Quindi il funzionamento dei servizi proposti è uguale a quello descritto nel paragrafo 8.

#### Diagramma di sequenza Utente.

Il diagramma in figura 30 mostra la sequenza di azioni che riguardano l'utente. Per prima cosa l'utente deve aver effettuato il login, dopodiché dovrà entrare nella sezione dedicata alla compilazione del diario, tale sezione è accessibile solamente se l'utente possiede già le sue massime misurazioni muscolari. L'utente inserirà gli input del diario essi verranno inviati al client che avrà il compito di controllare se sono corretti, nel caso in cui saranno corretti verranno inviati al server e salvati all'interno del db, nel caso contrario quindi input errati il client comunicherà all'utente l'errore tramite un pop-up. Per quanto riguarda la visualizzazione della scheda, gli esercizi vengono prelevati dall'apposita tabella del database, inviati al server e da questi al client che utilizzando gli appositi frame farà vedere gli esercizi da svolgere.

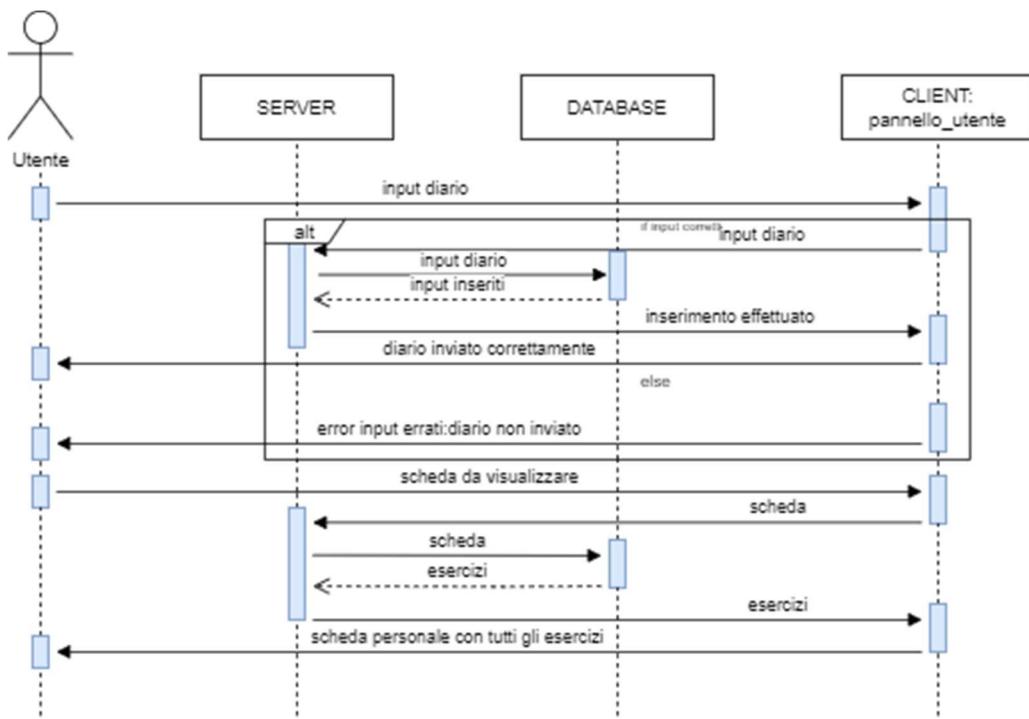


Figura 30 Sequence diagram utente.

#### Diagramma di sequenza trainer.

Per quanto riguarda il trainer come si vede in figura 30.1, egli deve poter visualizzare il diario creato dall'utente, per fare ciò viene fatta una visita al

nostro db che restituirà al server tutti i dati dei diari contenenti l'email del trainer in questione poiché l'utente può decidere chi sarà il trainer che creerà e modificherà la scheda. Per la modifica, gli esercizi verranno prelevati dal database, inviati al server, passati al client e infine mostrati all'interno dei *jComboBox*. Gli esercizi che verranno mostrati riguardano solo una parte del corpo, quindi un esercizio può essere sostituito solamente con uno che allena la stessa parte del corpo. In questo modo manteniamo la priorità di allenamento della parte del corpo che ne ha più bisogno.

Per fare la modifica il trainer sceglierà uno o più esercizi che dovranno essere cambiati, dopodiché gli esercizi scelti verranno comunicati dal client al server e infine al database che modificherà la scheda con i nuovi esercizi scelti dal trainer. Nell'operazione di modifica non è stato adibito alcun controllo in quanto gli esercizi vengono prelevati dal database, e il controllo avviene nel momento in cui questi esercizi vengono inseriti.

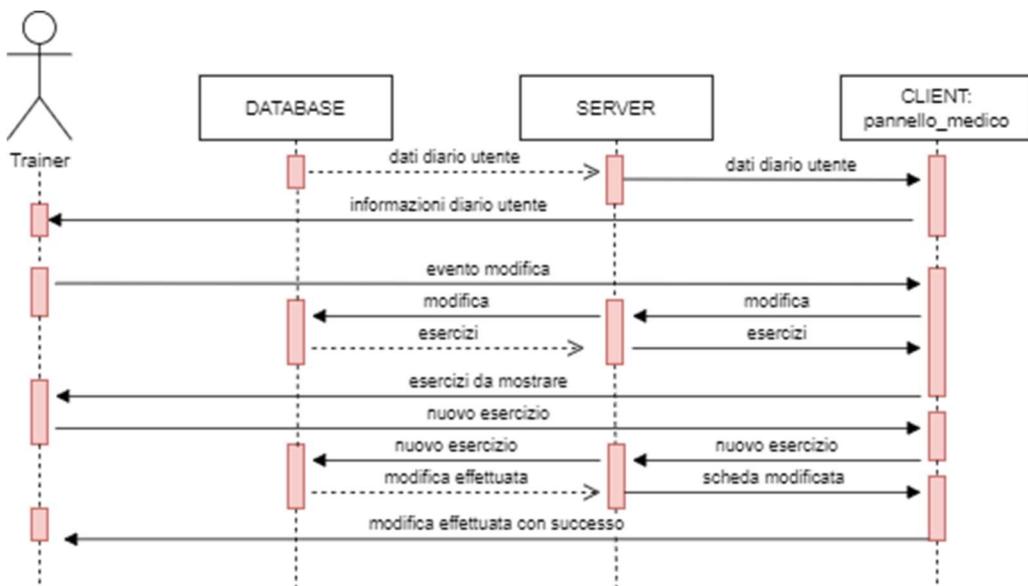


Figura 30.1 Sequence diagram trainer.

### Diagramma di sequenza admin.

L'admin ha la possibilità di gestire i corsi. Per prima cosa può inserire un nuovo corso, come si vede in figura 30.2, il client riceve gli input e ha il compito di controllarli, se sono corretti vengono inviati al server e salvati nel database. Lo stesso procedimento utilizzato per l'inserimento viene attuato sia per la modifica degli esercizi di un corso che per l'ampliamento degli esercizi di un corso, che altro non è che un inserimento. Per quanto riguarda l'eliminazione l'admin sceglierà il corso da eliminare e lo comunicherà al client, dopodiché sarà il server a far partire l'evento di eliminazione sul database. Come si può vedere dal diagramma sarà il client ad avere il compito di verificare che gli input siano corretti.

Nel diagramma per evitare ripetizioni, nella modifica, ampliamneto e eliminazione dei corsi è stata omessa la sequenza relativa alla visualizzazione in quanto identitica per tutti i casi.

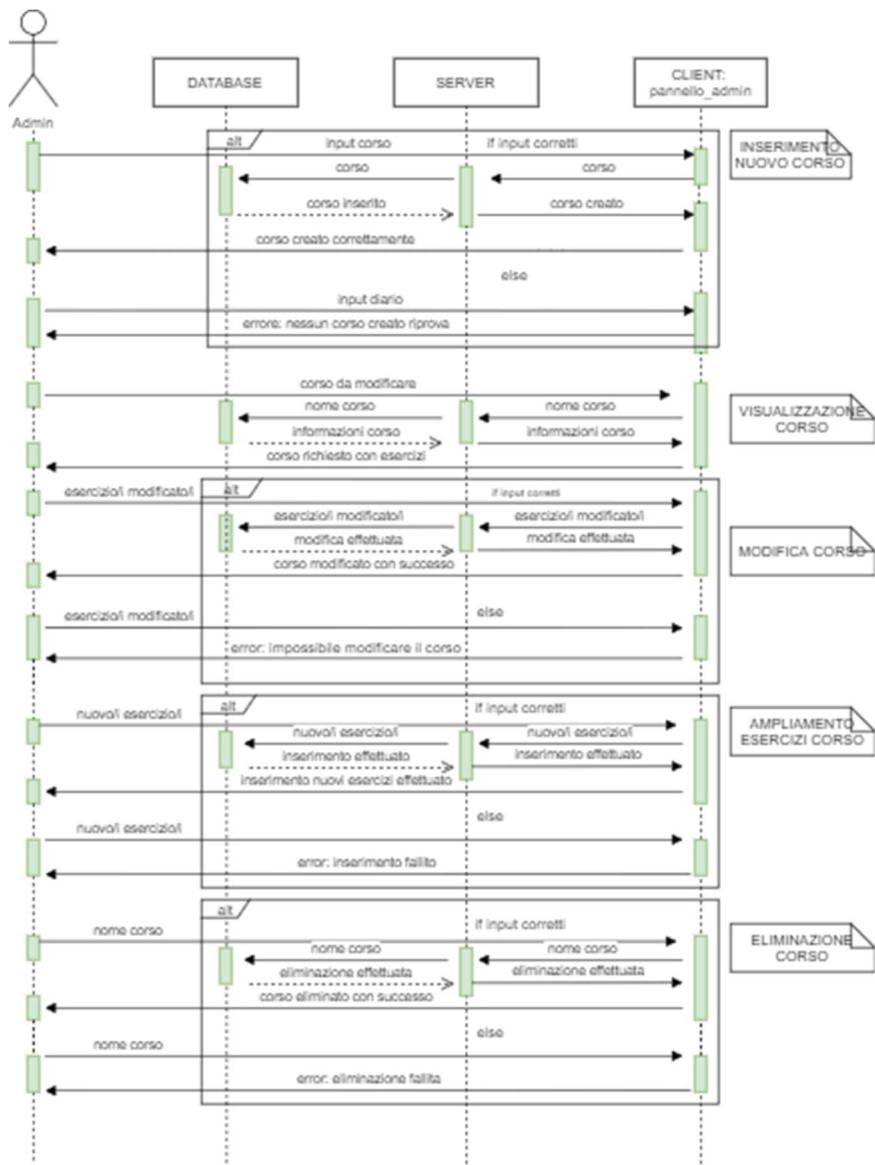


Figura 30.2 Sequence diagram admin.

## 31. Terza attività: PROGETTAZIONE.

### 31.1 Architettura di sistema.

L'architettura di sistema rimane client-server.

### 31.2 Database (db).

Il database viene utilizzato per immagazzinare al suo interno i dati del diario, gli esercizi da svolgere, gli esercizi suddivisi per parti del corpo ed infine i corsi ognuno con i propri esercizi.

Come sappiamo l'archittetura del nostro sistema è del tipo client-server. Per questo motivo il database ha un ruolo abbastanza importante per far in

modo che le informazioni vengano condivise tra tutti gli attori che utilizzano il sistema.

gymforyou diario	gymforyou esercizi	gymforyou esercizicorso
id : int(5)	id : int(5)	id : int(5)
utente : varchar(255)	email : varchar(255)	nome : varchar(255)
tipo : varchar(255)	tipo : varchar(255)	trainer : varchar(255)
parte_corpo : varchar(255)	Lunedì : mediumtext	1 : varchar(255)
giorni : varchar(255)	Martedì : mediumtext	2 : varchar(255)
mattina : varchar(255)	Mercoledì : mediumtext	3 : varchar(255)
pomeriggio : varchar(255)	Giovedì : mediumtext	4 : varchar(255)
sera : varchar(255)	Venerdì : mediumtext	5 : varchar(255)
calorie : varchar(255)	Sabato : mediumtext	6 : varchar(255)
trainer : varchar(255)	Domenica : mediumtext	7 : varchar(255)
fabbisogno : varchar(50)		8 : varchar(255)

gymforyou esercizischeda
id : int(5)
parte : varchar(255)
nome : varchar(255)
met : double
rep : int(100)

Figura 30 Tabelle database.

Le operazioni che sarà possibile fare sul database sono:

- 1) Inserimento dati diario;
- 2) Visualizzazione diario;
- 3) Salvataggio diario e scheda personale;
- 4) Modifica esercizi;
- 5) Gestione dei corsi.

Tutte le operazioni sopra indicate sono presenti all'interno della classe database, contenuta nel progetto **GymForYou\_server**.

### 31.3 Diagrammi degli stati.

**Diagramma degli stati: Invio dati e visualizza scheda.**

L'utente deve effettuare il login, in seguito entrerà nell'apposita sezione di compilazione del diario all'interno della quale dovrà inserire i valori relativi al diario di allenamento, naturalmente i valori inseriti potranno essere modificati prima di essere inviati. Per la visualizzazione della scheda l'utente dovrà andare nella sezione dedicata scegliere la scheda da visualizzare nel caso ne abbia richieste più di una e infine troverà davanti a sé gli allenamenti da svolgere.

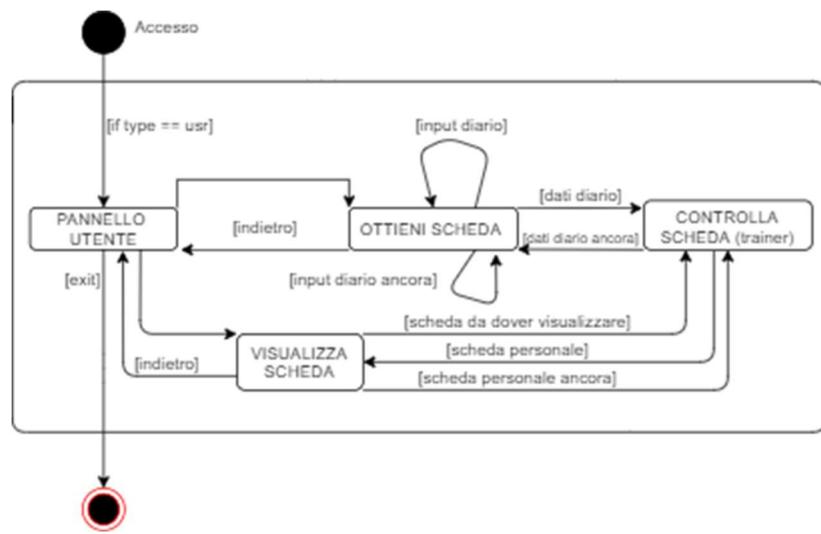


Figura 31.1 State diagram invio dati diario e visualizzazione scheda.

### Diagramma degli stati: Creazione scheda.

Per la creazione della scheda il meccanismo è tutto completamente automattizzato, l'unica azione che devono essere svolte dal trainer sono l'accesso alla piattaforma e la scelta della scheda da creare. Se il trainer ritiene che la scheda generata in modo automatico vada bene deve solamente confermarla così da essere salvata nel database e poi visualizzata dall'utente. Nel momento in cui una scheda viene creata viene automaticamente eliminata, ciò accade per evitare che una scheda venga creata più volte.

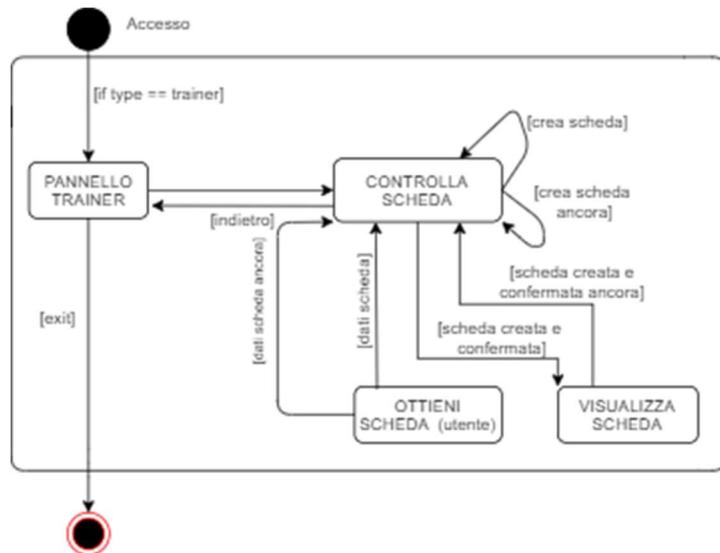


Figura 31.2 State diagram creazione scheda.

### Diagramma degli stati: Modifica scheda.

Innanzitutto il trainer deve aver effettuato il log-in, dopodiché recandosi nella sezione dedicata alla creazione della scheda dovrà generarla, se reputa che uno o più esercizi siano da modificare lo può fare. La modifica consiste nel sostituire un esercizio con uno che allena la stessa parte del corpo.

Dopo la modifica la scheda va confermata così da poter essere visualizzata dall'utente.

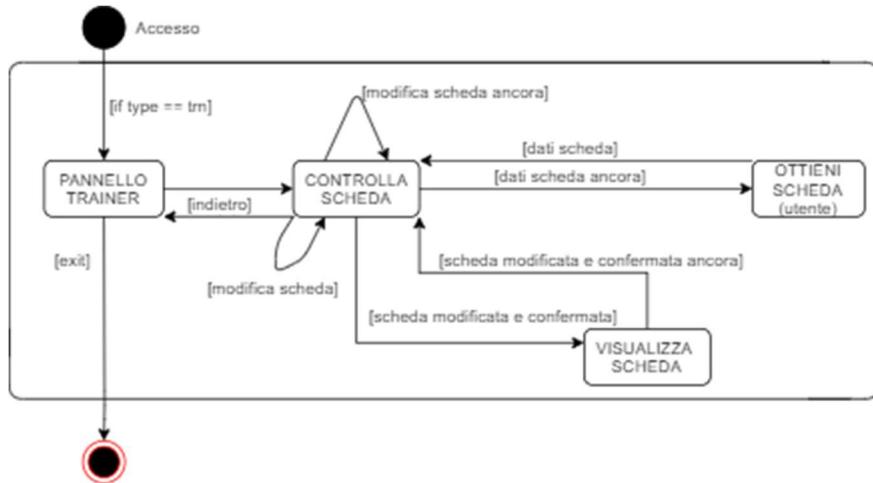


Figura 31.3 State diagram modifica scheda.

### Diagramma degli stati: Gestione corsi.

L'admin deve effettuare l'accesso. Quando sceglierà di entrare nella sezione di gestione corsi le operazioni che potrà svolgere sono l'inserimento di un nuovo corso con i suoi esercizi, la modifica e/o l'ampliamento di un corso già creato e infine l'eliminazione di un corso esistente.

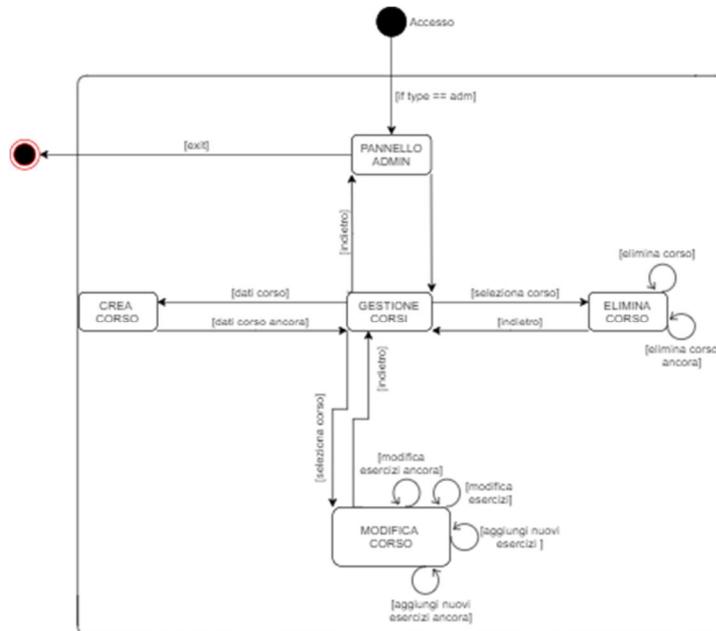


Figura 31.4 State diagram gestione corsi.

## 32. Descrizione delle GUI.

### Pannello utente.

Nel pannello utente (figura 32) sono stati aggiunti due nuovi button uno “crea diario” che è l’altro “visualizza schede” a cui sono stati associati i rispettivi frame.



Figura 32 Nuovo pannello utente.

### Crea diario.

In figura 32.1 e figura 32.2 e figura 32.3 possiamo vedere come sono composti i due diversi frame per la creazione del diario. Per realizzarli il team si è servito di diversi strumenti tra cui i *jButton*, *jTextField* e *jComboBox* che permettono all’utente in modo semplice e veloce di compilare il proprio diario. Nel primo frame si dovrà scegliere il tipo di allenamento e la suddivisione giornaliera. Nel secondo si dovrà scegliere l’intensità di allenamento. Infine bisognerà indicare se si vuole fare un allenamento generico oppure se si vogliono svolgere esercizi relativi ad un corso.



Figura 32.1 Scegli allenamento



Figura 32.2 Scegli intensità



Figura 32.3 Scegli tipologia

### Pannello trainer.

Al pannello trainer (figura 32.4) è stato aggiunto un jButton che lo reindirizzerà verso il frame per la visualizzazione e poi creazione della scheda.



Figura 32.4 Nuovo pannello trainer.

## Crea scheda.

Il frame per l'assegnazione della scheda (figura 32.5) sarà composto da una *jTable* in cui sono presenti tutte le informazioni inserite dall'utente. Il trainer per generare una scheda in modo automatizzato dovrà solamente selezionare la riga della tabella e in seguito cliccare sul *jButton* “crea scheda”.



Figura 32.5 Crea scheda

## Assegna scheda 2.

Il team di sviluppo ha deciso di implementare 3 diversi frame, in base alla suddivisione degli allenamenti. Se l'utente ha scelto di suddividere gli allenamenti in un solo momento della giornata avremmo il frame in figura 32.6, se ha scelto due momenti quello in figura 32.7 e infine se ha scelto tre momenti quello in figura 32.8.

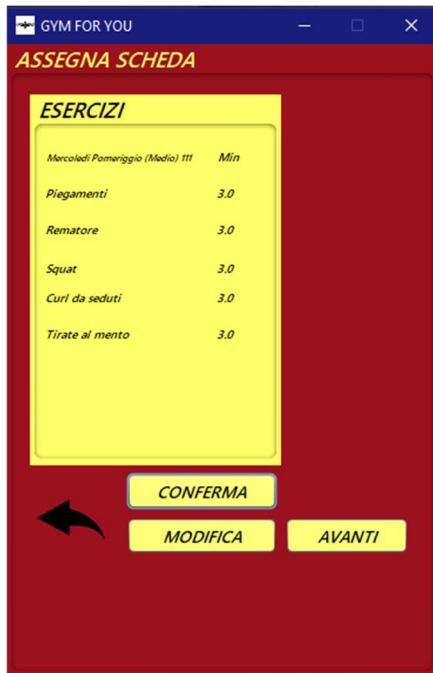


Figura 32.6 Scheda un momento



Figura 32.7 Scheda due momenti



Figura 32.8 Scheda tre momenti

Nei frame in questione il trainer all'interno delle *jLabel* predisposte potrà visualizzare tutte le informazioni relative agli esercizi assegnati in maniera automatizzata.

In basso al frame troviamo 3 *jButton* diversi:

- cliccando sul bottone “**avanti**” il trainer passerà al frame successivo dove se scelti, verranno presentati la suddivisione degli allenamenti per i giorni successivi.
- il *jButton* “**modifica**” permetterà di ritoccare gli esercizi quando il trainer lo ritiene necessario. L'evento associato al *jButton* in questione farà comparire dei *jComboBox* in cui saranno presenti tutti gli esercizi da poter inserire. Le figure 32.9, 32.10 e 32.11 mostrano come appariranno i frame dopo aver premuto il bottone.



Figura 32.9 Modifica un momento

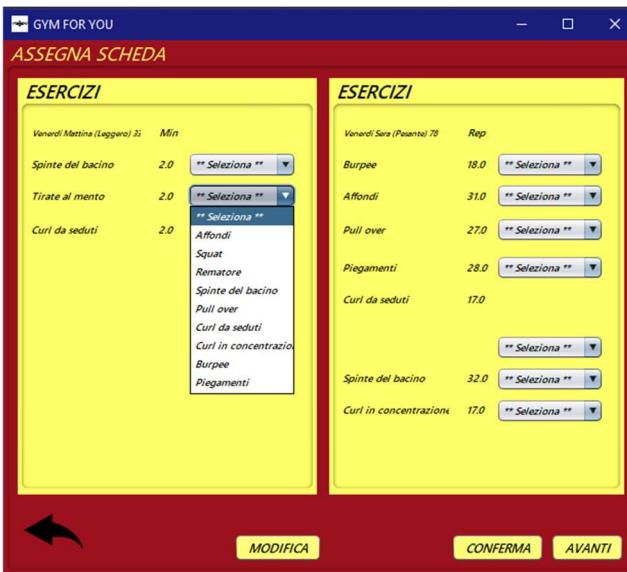


Figura 32.10 Modifica due momenti



Figura 32.11 Modifica tre momenti

- **“conferma”** permetterà di confermare la scheda, l’evento associato al jButton in questione è di scrittura della scheda sul database.

### Visualizza scheda Utente.

Prima di visualizzare le schede di allenamento l’utente deve scegliere quale scheda visualizzare nel caso in cui ne abbia richieste più di una (figura 32.12).

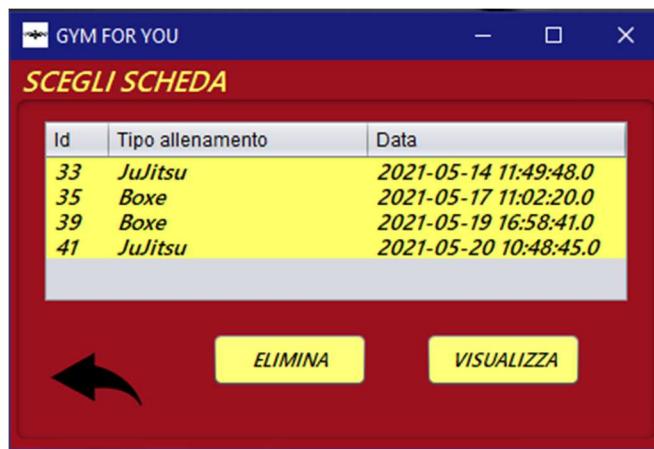


Figura 32.12 Visualizza schede.

Per far visualizzare gli esercizi fisici da svolgere si è scelto di sfruttare la stessa suddivisione dei frame utilizzata nell’assegnazione della scheda. Abbiamo creato i frame nel caso in cui si è scelto un solo momento (figura 32.13), quello in cui se sono scelti due (figura 32.14) e infine quello composto

da tre momenti (figura 32.15). Ogni frame ha al suo interno delle *jLabel* con gli esercizi da svolgere e le ripetizioni o minuti per ogni esercizio. In basso troviamo un *jButton* grazie al quale cliccandolo l'utente può spostarsi nei diversi giorni di allenamento e visualizzare gli esercizi da svolgere.



Figura 32.13 Visualizza Scheda  
Allenamento1



Figura 32.14 Visualizza Allenamento2

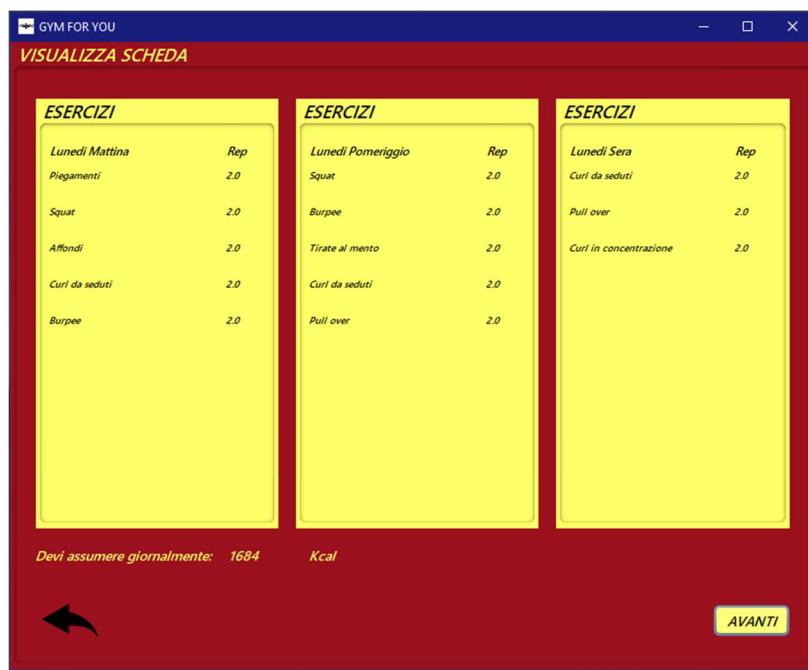


Figura 32.15 Visualizza Allenamento3

## Pannello Admin.

Al pannello admin (figura 32.16) è stato aggiunto un *jButton* che porterà l'admin nel frame per la gestione dei corsi.

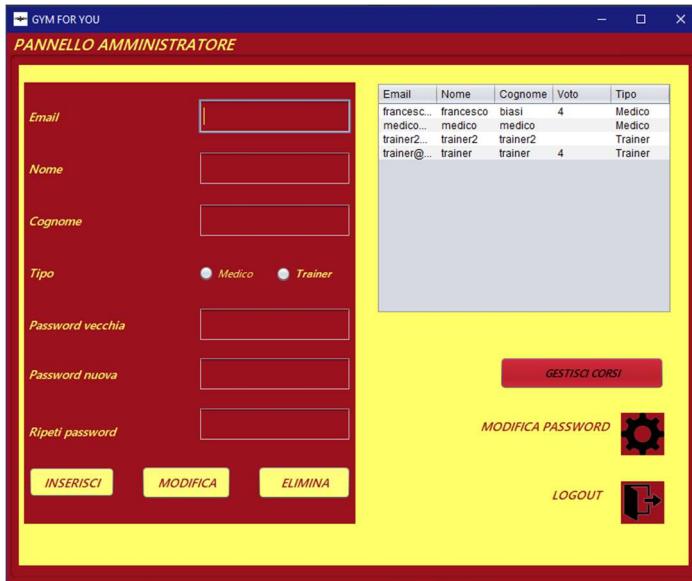


Figura 32.16 Nuovo pannello admin

## Gestione corsi.

L'homepage della gestione dei corsi come si può vedere in 32.17 è composta da una *Jtable* al cui interno troviamo le informazioni dei corsi già creati. L'admin per eseguire un'azione su un corso dovrà scegliere una riga della tabella e poi premere il *jButton* corrispondente all'azione che vuole svolgere.



Figura 32.17 Gestisci corsi.

## Crea corso.

All'interno di questo frame (figura 32.18) troviamo un *jComboBox* in cui saranno presenti i trainer presenti nel nostro sistema. In seguito abbiamo

10 *jTextField* che devono essere utilizzati per scrivere i dieci esercizi che compongono il corso che si sta creando.

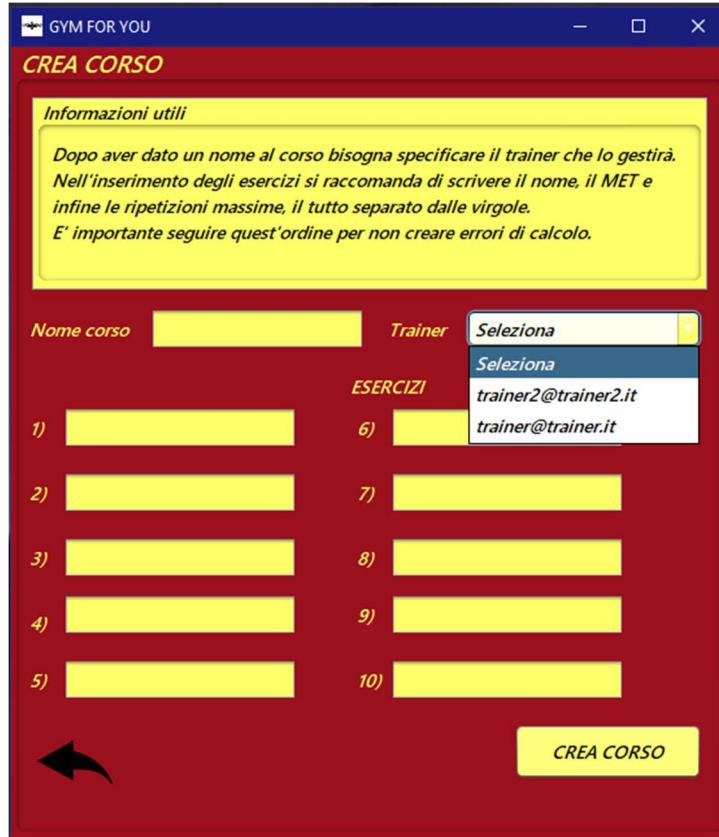


Figura 32.18 Crea corso.

### Modifica e amplia corso.

In questo frame (figura 32.19) abbiamo sempre 10 *jTextField* che saranno popolati con gli esercizi del corso selezionato essi potranno essere modificati oppure sostituiti con altri esercizi. Nel caso in cui l'admin vuole ampliare il corso scelto dovrà dare una risposta positiva al *RadioButton*, nel momento in cui verrà fatto appariranno altri 3 *jTextField* che potranno essere sfruttati per l'aggiunta di nuovi esercizi.



Figura 32.19 Modifica e amplia corso.

## 32.1 Descrizione algoritmo.

Come ampiamente descritto in precedenza questo terzo prototipo si basa sulla creazione di una scheda personale decisa, dall'utente. Dato che la creazione della scheda avviene in modo totalmente automatizzato, come richiesto dal cliente, il team ha dovuto implementare un algoritmo che facesse tutte le operazioni in maniera automatica. In seguito andremo a illustrare le metodologie applicate per creare l'algoritmo.

### Gestione input.

Per prima cosa il nostro algoritmo dovrà gestire tutti gli input provenienti dalla compilazione del diario.

Gli input sono:

1. Parti del corpo da allenare;
2. Metodologia di allenamento (dimagrimento o aumento massa muscolare);
3. Giorni della settimana in cui allenarsi;
4. Momenti della giornata in cui allenarsi;
5. Intensità di allenamento per ogni momento;

Riguardo l'input numero uno c'è da fare una piccola precisazione ovvero, se dal calcolo delle massime misurazioni muscolari (che deve essere obbligatoriamente fatto se si vuole avere una scheda personale), risulta che una o più parti del corpo sono già allenate esse non potranno essere scelte per la scheda personale, infatti non compariranno nemmeno tre le scelte.

### Compito dell'algoritmo.

Il compito dell'algoritmo sarà di:

- elaborare tutti gli input ricevuti;
- prelevare dai file xml gli esercizi;
- calcolare le calorie bruciate per ogni esercizio;
- generare la scheda.

La scheda generata dovrà rispettare le regole imposte dal team di sviluppo e la suddivisione degli allenamenti creata dall'utente. I dettagli di come ciò è stato realizzato verranno illustrati nei sequenti paragrafi.

### Regole algoritmo.

Un algoritmo che si rispetti soprattutto quando deve gestire molti input deve seguire delle regole che vengono detatte da chi lo sta creando. Il team ha imposto le seguenti regole:

- 1) In base ai momenti scelti, alle parti del corpo da allenare e all'intensità di allenamento gli esercizi verrano così suddivisi:

	Leggero	Medio	Pesante
	Esercizi	Esercizi	Esercizi
1 PARTE DEL CORPO	2	3	5
2 PARTE DEL CORPO	2	3	5
3 PARTE DEL CORPO	3	5	7
4 PARTE DEL CORPO	4	7	9
5 PARTE DEL CORPO	5	8	10

- 2) In base all'intensità di allenamento scelta, le kcal da bruciare saranno suddivise nel seguente modo:

MOMENTI	INTENSITA' ALLENAMENTO		
	LEGGERO	MEDIO	PESANTE
1 MOMENTO	100%		
		100%	
			100%
2 MOMENTI	100%		
		100%	
			100%
	40%	60%	
	30%		70%
		40%	60%
TRE MOMENTI	100%		
		100%	
			100%
	15%	35%	50%
	33%-33%-34%		
	20%-20%	60%	
	15%-15%		70%
	20%	40%-40%	
	10%		45%-45%
		33%-33%-34%	
		25%-25%	50%
			33%-33%-34%

Le percentuali esprimono il modo in cui verrà suddiviso il monte delle kcal da bruciare. Naturalmente vale il viceversa quando abbiamo due o tre momenti, quindi per esempio se un utente sceglie un'intensità media e pesante, oppure pesante e media la suddivisione sarà uguale.

- 3) Un'altra regola che il team ha imposto è che se un utente ha scelto un'intensità di allenamento con un numero di esercizi totali minore di 6 allora gli esercizi dovranno essere svolti seguendo una suddivisione in minuti. Nel caso in cui gli esercizi totali da eseguire sono maggiori di 6, l'esecuzione degli esercizi dovrà essere fatta seguendo le ripetizioni. Si è deciso di agire in questo modo in quanto assegnare le ripetizioni per un numero esiguo di esercizi risulta non corretto poiché le ripetizioni assegnate erano più del dovuto.
- 4) Per evitare errori di calcolo il team ha imposto ai trainer solo la modifica dell'esercizio e non del numero di ripetizioni o di minuti, inoltre l'esercizio da modificare può essere sostituito **solo** con uno che allena la stessa parte del corpo. Questa scelta è stata fatta in quanto il calcolo delle calorie da bruciare deve rimanere sempre automatizzato.

Detto questo il nostro algoritmo è stato implementato all'interno della classe **“Scelta”**.

### Metodo per il calcolo delle calorie da bruciare.

Per prima cosa il team grazie all'aiuto dei testi presenti nella bibliografia, ha dovuto implementare il calcolo delle calorie da bruciare.

Se un utente ha scelto di **dimagrire** per prima cosa dovrà indicare nell'apposito *jTextFielded* quanti grammi vuole perdere a settimana. Dopodiché il calcolo che verrà fatta è il seguente:

$$x = \frac{\text{peso da perdere}}{450}$$

Il numero presente al denominatore è una costante trovata dallo studioso.

Per ottenere il numero di kcal da bruciare bisogna fare il seguente calcolo:

$$\text{kcal da bruciare} = 500 * x$$

Dove  $x$  è il risultato dell'equazione precedente, anche in questo caso il moltiplicando è una costante trovata dallo studioso.

Le *kcal da bruciare* adesso verranno suddivise dal nostro algoritmo in base alle scelte di intensità di allenamento fatte dal nostro utente.

## Esempio:

Un utente deve bruciare 300kcal è ha indicato un allenamento che si svolgerà in un unico giorno ma in due momenti diversi ovvero mattina e sera con un intensità pesante per la mattina e media per la sera.

L'algoritmo preleverà da un'array il numero di momenti e l'intensità indicata. In base alle regole dettate dal team tramite il costrutto condizionale **if-else if** esso saprà che nel caso in cui i momenti sono due e le intensità sono medie e pesante i calcoli da fare sono:

$$\text{intensità media} = 300\text{kcal} * 0,4$$

$$\text{intensità pesante} = 300\text{kcal} * 0,6$$

Grazie alle regole date, l'algoritmo farà bruciare 180kcal la mattina e 120kcal la sera.

Grazie a queste regole lo stress fisico che dovrà sostenere l'utente è in linea con la scelta dell'allenamento che egli ha fatto.

Nel caso in cui un utente vuole **aumentare** la massa muscolare dovrà comunicare il suo lifestyle (sedentario, poco attivo, ecc), e da quanti anni fa attività fisica. Inseriti questi input verranno fatti i seguenti calcoli:

- Per prima cosa bisogna conoscere il suo fabbisogno calorico giornaliero (TEE) che si calcola nel seguente modo:

$$BMR = 370 + (21.6 * \text{massa magra})$$

$$TEE = BMR * \text{lifestyle}$$

La massa magra viene prelevata dal file *xml* dell'utente invece *lifestyle* indica delle costanti associate ad ogni tipo di stile di vita. Il BMR indica il fabbisogno medio. Il TEE indica il fabbisogno calorico per mantenere il peso corrente in base allo stile di vita dell'utente, ed è quello che serve all'algoritmo.

- Per sapere quanti cm di massa muscolare può assumere in una settimana il nostro algoritmo farà:

$$x = \frac{0.3 * \text{circ polso}^2 * 0,5^{(\text{anni allenamento}-1)}}{52}$$

- Per conoscere settimanalmente quante kcal deve assumere per aumentare la massa muscolare, partiamo dal presupposto che in una settimana un umano in media consuma 3500kcal (dato preso da uno studio) e poi facciamo i seguenti calcoli:

$$kcal = \frac{3500 * x}{7}$$

In questo modo sappiamo quante calorie il nostro utente dovrà aggiungere alla proprio TEE e quante ne dovrà consumare per aumentare in modo controllato la propria massa muscolare.

Il team tiene a precisare che l'aumento della massa muscolare dipende molto da fattori come la struttura ossea, il metabolismo ecc. pertanto non assicura completamente l'aumento della massa.

### **Metodo per l'assegnazione degli esercizi.**

Prima di esporre il funzionamento dell'algoritmo bisogna fare un piccola premessa su due valori molto importanti che vengono usati per fare i calcoli, i valori sono il MET e le ripetizioni medie. Il MET esprime il dispendio di ossigeno durante un'attività ed è un valore costante associato ad un esercizio. Le ripetizioni medie indicano mediamente quante ripetizioni vengono fatte in un minuto per un determinato esercizio.

Il team ha preso questi valori da studi fatti nell'ambito delle scienze motorie.

Tutti gli esercizi presenti all'interno del nostro sistema devono obbligatoriamente contenere questi due valori.

Per assegnare gli esercizi in modo equo l'algoritmo progettato dal team segue questa logica:

- Calcola quante kcal fa perdere un esercizio in un minuto:

$$kcal1min = 0,0175 * peso * MET$$

Nel caso in cui gli esercizi da fare sono maggiori di 6 deve suddividere l'esecuzione degli esercizi usando le ripetizioni e lo fa con i seguenti calcoli:

$$x = \frac{kcal1min}{repmedie}$$

Dopo bisogna prelevare le kcal giornaliere da bruciare e fare:

$$rep = \frac{kcal}{x}$$

Facendo ciò riusciamo a capire quante ripetizioni di un determinato esercizio assegnare per bruciare le kcal indicate.

Nel caso in cui gli esercizi da fare sono minori di 6, l'algoritmo deve suddividere l'esecuzione degli esercizi usando i minuti e lo fa utilizzando i calcoli fatti in precedenza però aggiungendo il seguente calcolo:

$$\frac{rep}{rep\ media}$$

Tramutando il risultato in minuti si sa quanti minuti di esercizio fare per bruciare le kcal indicate.

### Funzionamento dell'algoritmo in caso di modifica.

Come sappiamo i nostri trainer hanno la possibilità di modificare qualsiasi esercizio essi vogliano seguendo le regole per la modifica.

L'evento per la modifica è associato appunto al bottone “*modifca*”, una volta premuto compariranno dei combo box che conterranno tutti gli esercizi presenti nel database. Il team per rispettare la regola 4 ha implementato un metodo che va a prelevare all'interno del nostro database solo gli esercizi relativi alla parte del corpo di cui fa parte l'esercizio che deve essere sostituito. Quando l'esercizio viene modificato il trainer dovrà premere sul bottone “*conferma*” una volta premuto verranno rifatti i calcoli descritti sopra tenendo conto però dei nuovi valori (MET e ripetizioni medie) relativi ai nuovi esercizi.

### 33. Quarta attività: IMPLEMENTAZIONE.

#### 33.1 Diagramma dei componenti: client server (aggiornato).

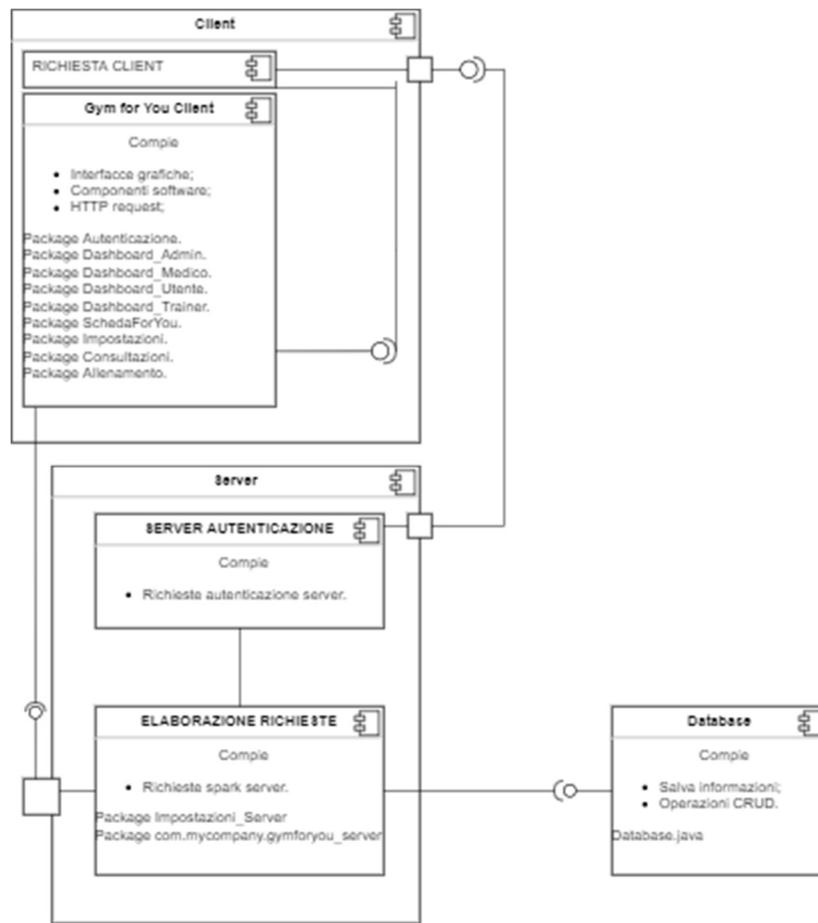


Figura 33 Component diagram client-server aggiornato.

### 33.2 Diagramma dei componenti: Gym for you client (aggiornato).

In figura 33.1 possiamo vedere come si aggiorna il diagramma dei componenti per questo terzo prototipo.

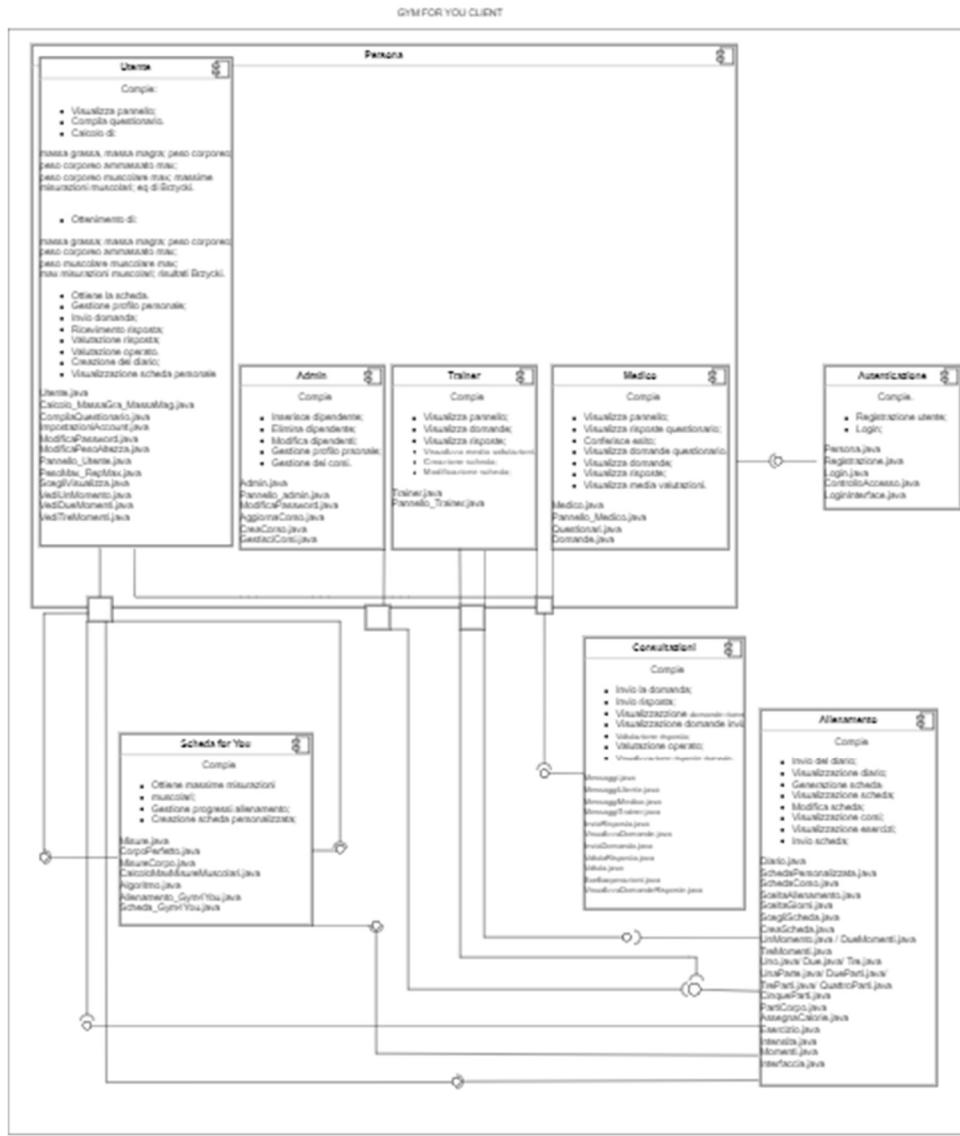


Figura 33.1 Component diagram Gym for You client aggiornato.

## 33.2 Caratteristiche OOP.

Nei seguenti paragrafi andremo a descrivere in che modo sono state utilizzate le caratteristiche della programmazione orientata agli oggetti all'interno del nostro sistema.

### Incapsulamento e information hiding.

Nel nostro sistema il team ha utilizzato l'incapsulamento per diversi scopi. Nel primo prototipo tale caratteristica è servita per prelevare (**get**) il peso, l'altezza ed altre caratteristiche fisiche legate alla classe “**Utente**”. Nella classe Utente (figura 33.2) troviamo i diversi attributi, per accedere a tali attributi lo facciamo attraverso i metodi presenti in figura 33.3 metodi ed attributi che però sono contenuti e nascosti in tutt'altra classe, quindi la classe utente per poter accedere a tali attributi lo potrà fare solamente tramite il metodo *get()*. In questo modo gli attributi rimangono protetti da entità esterne.

```
public class Utente extends Persona implements LoginInterface{  
    private String questionario;  
    private String esito;  
    private double risultato = 0;  
    private Map dati;  
  
    public Utente(){  
        this.questionario = getXmlpars2().getElement("Questionario");  
        this.esito = getXmlpars2().getElement("Esito_Dottore");  
    }  
}
```

Figura 33.2 Classe Utente.

```
public String getEmail() {  
    return email;  
}  
  
public String getNome() {  
    return nome;  
}  
  
public String getCognome() {  
    return cognome;  
}  
  
public String getGenere() {  
    return genere;  
}  
  
public float getPeso() {  
    return peso;  
}  
  
public float getAltezza() {  
    return altezza;  
}  
  
public float getPolso() {  
    return polso;  
}  
  
public float getCaviglia() {  
    return caviglia;  
}  
  
public float getCrif_vita() {  
    return crf_vita;  
}
```

Figura 33.3 Metodi *get()* classe Persona.

Un'altra parte del nostro sistema in cui il team si è servito di questa caratteristica OOP è stato quando nel primo prototipo ha dovuto tenere traccia degli andamenti delle parti del corpo. Abbiamo creato la classe “**Misure**” (figura 33.4) e grazie ai metodi presenti abbiamo potuto prelevare le varie misure inserite dai nostri utenti. Le suddette misure serviranno all'algoritmo per capire qual è la parte del corpo che va allenata di più rispetto alle altre, ciò significa che verranno assegnati un numero maggiore di esercizi per quella parte del corpo. Un altro motivo per cui l'incapsulamento è stato utilizzato, riguarda l'avanzamento delle *jProgressBar* infatti dopo aver ottenuto gli attributi tramite i metodi *get*,

vengono fatti dei calcoli inerenti per ottenere una percentuale di completamento dell'allenamento.

```
package SchedaForYou;
import Impostazioni.Xmlpars;
public class Misure {
    private float petto,petto_inter, petto_f;
    private float bicipite,bicipite_inter,bicipite_f;
    private float avambracci, avambracci_inter, avambracci_f;
    private float cosce,cosce_inter,cosce_f;
    private float polpacci,polpacci_inter,polpacci_f;
    private Xmlpars parser3 = new Xmlpars("misure.save");
    public Misure() {
        this.petto = Float.parseFloat(parser3.getElement("Petto_Iniziale"));
        this.petto_inter = Float.parseFloat(parser3.getElement("Petto_Intermedio"));
        this.petto_f = Float.parseFloat(parser3.getElement("Petto_Finale"));
        this.bicipite = Float.parseFloat(parser3.getElement("Bicipite_Iniziale"));
        this.bicipite_inter = Float.parseFloat(parser3.getElement("Bicipite_Intermedio"));
        this.bicipite_f = Float.parseFloat(parser3.getElement("Bicipite_Finale"));
        this.avambracci = Float.parseFloat(parser3.getElement("Avambraccio_Iniziale"));
        this.avambracci_inter = Float.parseFloat(parser3.getElement("Avambraccio_Intermedio"));
        this.avambracci_f = Float.parseFloat(parser3.getElement("Avambraccio_Finale"));
        this.cosce = Float.parseFloat(parser3.getElement("Cosce_Iniziale"));
        this.cosce_inter = Float.parseFloat(parser3.getElement("Cosce_Intermedio"));
        this.cosce_f = Float.parseFloat(parser3.getElement("Cosce_Finale"));
        this.polpacci = Float.parseFloat(parser3.getElement("Polpacci_Iniziale"));
        this.polpacci_inter = Float.parseFloat(parser3.getElement("Polpacci_Intermedio"));
        this.polpacci_f = Float.parseFloat(parser3.getElement("Polpacci_Finale"));
    }
}
```

Figura 33.4 Classe Misure.

```
public float getPetto() {
    return petto;
}

public float getPetto_inter() {
    return petto_inter;
}

public float getPetto_f() {
    return petto_f;
}

public float.getBicipite() {
    return bicipite;
}

public float.getBicipite_inter() {
    return bicipite_inter;
}

public float.getBicipite_f() {
    return bicipite_f;
}

public float.getAvambracci() {
    return avambracci;
}

public float.getAvambracci_inter() {
    return avambracci_inter;
}

public float.getAvambracci_f() {
    return avambracci_f;
}

public float.getCosce() {
    return cosce;
}

public float.getCosce_inter() {
    return cosce_inter;
}
```

Figura 33.5 Metodi classe Misure.

Oltre agli esempi visti in precedenza possiamo aggiungere che all'interno del nostro sistema tutte gli attributi delle classi sono protetti mediante il modificatore di accesso **private** e automaticamente ogniqualvolta i metodi hanno bisogno di tali attributi possono accederli solamente utilizzando i metodi *get()*, grazie a ciò riusciamo a proteggere l'accesso a tali attributi da fonti esterne. Oltre agli attributi, in alcuni casi anche i metodi di alcune classi godono di information hiding e encapsulamento, infatti essi sono stati protetti mediante l'utilizzo del modificatore di accesso *private*, in questo modo tali metodi risultano accessibili ed utilizzabili solamente nelle classi in cui sono inseriti e quindi non all'esterno. In seguito andiamo a mostrare delle immagini in cui viene rappresentato l'utilizzo di queste caratteristiche OOP. Ci teniamo a precisare che per ragioni di spazio evitiamo di inserire tutti i casi perciò se si vuole comprendere maggiormente ciò che si è fatto il team consiglia l'analisi del codice sorgente.

```

private final int size;
private final int eserciziDB;
private final int eserciziDBL;
private final String tipo;
private final List<Object> esercizi;
private final List<Object> listaEsercizi;
private final List<Object> listaMomenti;
private final List<Object> listaPartiCorpo;
private final List<Double> listaMetRep;
private final List<Integer> shuffle;
private final List<Integer> shuffleL;

public Esercizio(List<Object> listaEsercizi, List<Object> listaMomenti, List<Object> listaPartiCorpo, List<Object> listaMetRep) {
    this.size = size;
    this.eserciziDB = eserciziDB;
    this.eserciziDBL = eserciziDBL;
    this.tipo = tipo;
    this.esercizi = esercizi;
    this.listaEsercizi = listaEsercizi;
    this.listaMomenti = listaMomenti;
    this.listaPartiCorpo = listaPartiCorpo;
    this.listaMetRep = listaMetRep;
    this.shuffle = shuffle;
    this.shuffleL = shuffleL;
}

public int getSize() {
    return size;
}

public int getEserciziDB() {
    return eserciziDB;
}

public int getEserciziDBL() {
    return eserciziDBL;
}

public String getTipo() {
    return tipo;
}

public List<Object> getEsercizi() {
    return esercizi;
}

```

Figura 33.6

```

public class Persona {
    private final List<String> primiNome;
    private final List<Object> esercizi;
    private final List<Object> eserciziDB;
    private final List<Object> eserciziDBL;
    private final List<Object> eserciziL;
    private final List<Object> eserciziLDB;
    private final List<Object> eserciziLDBL;
    private final List<Object> eserciziOrdinate;
    private final int[] unoDici = new int[]{1, 2, 3, 4};
    private final int[] tre = new int[]{1, 3, 5, 7};
    private final int[] quattro = new int[]{1, 2, 4, 6};
    private final int[] seiSette = new int[]{1, 3, 5, 7, 9};
    private final String[] settanta = new String[]{"Leggero", "Medio", "Ressente"};
    private final Map<Date, Integer> dati = new HashMap<>();
    put(1, getEta());
    put(2, getEta());
    put(3, getEta());
    put(4, getEta());
    put(5, getEta());
    put(6, getEta());
}

public Momenti(List<String> primiNome, List<Object> esercizi, List<Object> eserciziDB,
    List<Object> eserciziDBL, List<Object> eserciziL, List<Object> eserciziLDB,
    List<Object> eserciziOrdinate) {
    this.primiNome = primiNome;
    this.esercizi = esercizi;
    this.eserciziDB = eserciziDB;
    this.eserciziDBL = eserciziDBL;
    this.eserciziL = eserciziL;
    this.eserciziLDB = eserciziLDB;
    this.eserciziOrdinate = eserciziOrdinate;
    this.vociPerOrdinare = sizeEserciziOrdinate;
}

public int[] getDici() {
    return unoDici;
}

public int[] getTre() {
    return tre;
}

```

Figura 33.7

## Ereditarietà e polimorfismo.

Tali caratteristiche sono state ampiamente usate dal team durante lo sviluppo del sistema.

Il team ha creato una superclasse “**Persona**” (figura 33.6), all’interno della quale sono presenti tutti gli attributi e i metodi. Tramite queste caratteristiche di OOP, abbiamo fatto in modo che gli attributi e i metodi presenti all’interno della classe padre sono stati ereditati dalle classi figlie. Grazie a queste caratteristiche il team ha ottenuto il vantaggio di non dover editare codice ridondante, ma ha creato dei metodi comuni che possono essere riutilizzati. Per quanto riguarda il polimorfismo il team nel caso della classe in esame ha deciso di optare per il tipo di poliformismo dinamico quindi i metodi sono stati ereditati tramite l’**override**. Il team ha preferito questo tipo di polimorfismo in quanto si adegua alle caratteristiche progettuali e di sviluppo del sistema creato.

```

package Autenticazione;

import Impostazioni.JSON;
import Impostazioni.Richieste;
import Impostazioni.Xmlpars;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;

public class Persona {

    private String email;
    private String password;
    private float peso, altezza, polso, caviglia;
    private float crf_vita, crf_collo, crf_fianchi;
    private double massagrasse, massamagra;
    private final String nome, cognome;
    private final String genere;
    private final Xmlpars xmppars = new Xmlpars("Impostazioni.xml");
    private final JSON json = new JSON();
    private final Xmlpars xmppars2 = new Xmlpars("filexml.xml");
    private final Richieste richieste = new Richieste(xmppars.getElement("PROTOCOL"), xmppars.getElement("SERV_ALCHES"));

    public Persona(String email, String password) {
        settings();
        this.email = email;
        this.password = password;
        this.nome = xmppars2.getElement("Nome");
        this.cognome = xmppars2.getElement("Cognome");
        this.genere = xmppars2.getElement("Genere");
    }

    public Persona() {
        settings();
        this.email = xmppars2.getElement("Email");
        this.nome = xmppars2.getElement("Nome");
        this.cognome = xmppars2.getElement("Cognome");
        this.genere = xmppars2.getElement("Genere");
        if (xmppars.nameRoot().equals("Utente")) {
            this.peso = Float.parseFloat(xmppars.getXmlElement("Peso"));
            this.altezza = Float.parseFloat(xmppars.getXmlElement("Altezza"));
            this.polso = Float.parseFloat(xmppars.getXmlElement("Polso"));
            this.caviglia = Float.parseFloat(xmppars.getXmlElement("Caviglia"));
            this.crf_vita = Float.parseFloat(xmppars.getXmlElement("Circonference_Vita"));
            this.crf_collo = Float.parseFloat(xmppars.getXmlElement("Circonference_Collo"));
            this.crf_fianchi = Float.parseFloat(xmppars.getXmlElement("Circonference_Fianchi"));
        }
    }
}

```

Figura 33.8 Classe Persona.

Tra le classi figlie troviamo: admin, trainer e utente che possono essere visionate nella loro interezza all'interno del package autenticazione. Qui in figura 33.7 mostriamo solo un piccolo estratto della sottoclassse “**Admin**”.

```

package Autenticazione;

import Dashboard_Admin.Pannello_Admin;
import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;

public class Admin extends Persona implements LoginInterface{

    private Map dati;
    public Admin() {
    }

    @Override
    public void accedi(String email, String password) {
        //SCRIVO FILEXML.SAVE CON I VALORI PRELEVATI DAL DB
        dati = new HashMap();
        dati.put("email", email);
        getXmlpars2().ScriviXML("Admin", dati);
        //
        Pannello_Admin pannello_admin = new Pannello_Admin();
        pannello_admin.setVisible(true);
    }

    @Override
    public String controllo(String email, String password) {
        return super.controllo(email, password); //To change body of generated methods, choose Tools | Templates.
    }
}

```

Figura 33.9 Classe Admin.

L'ereditarietà e i suoi vantaggi sono stati poi utilizzati dal team anche all'interno del package Allenamento. Infatti il team ha creato la superclasse “**Diario**”(figura 33.8)

```

package Allenamento;

import Autenticazione.Trainer;
import Impostazioni.JSON;
import Impostazioni.Richieste;
import Impostazioni.XmlParser;
import SchedaTuttiUtenti;
import java.io.InputStream;
import java.util.ArrayList;
import java.util.Collection;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class Dario {

    private final XmlParser xpmpars = new XmlParser("Impostazioni.xml");
    private final JSON json = new JSON();
    private final Richieste richieste = new Richieste(xmlparser.getElement("PHOTOURL"), xmlparser.getElement("HTTP_ADDRESS"), xmlparser.getElement("HTTP_PORTA"));
    private final XmlParser pxmpars = new XmlParser("passwords.xml");
    private Map dati;
    private final String[] partitiDelCorpo = new String[] {"Fatto", "Bicipiti", "Avambracci", "Cresce", "Polsacci"};
    private final String[] initialiPartiCorpo = new String[] {"A", "B", "C", "D"};
    private final String[] day = {"Lunedì", "Martedì", "Mercoledì", "Giovedì", "Venerdì", "Sabato", "Domenica"};

    public Dario() {
        if (!xmlparser.exists()) {
            //SE IL FILE IMPOSTAZIONI.XML NON ESISTE VIENE CREATO CON I VALORI DI DEFAULT
            xmlparser.ImpostaDefault();
        }
    }
}

```

Figura 33.10 Classe Dario.

che con i suoi attributi e metodi è stata ereditata nelle sottoclassi presenti sempre all'interno del suddetto package tra cui: SchedaPersonalizzata(figura 33.7) e SchedaCorso(figura 33.8).

```

package Allenamento;

import java.io.InputStream;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class SchedaPersonalizzata extends Dario {

    private Map dati;

    public Map prelevaTrainer() {
        dati = new HashMap();
        InputStream richiesta = getRichieste().GetRichiesta("/scelta/prelevaTrainer", dati, null);
        dati.clear();
        dati = getJson().LeggiJson(richiesta);

        return dati;
    }

    public Map modificaSchedaPersonalizzata(List<Object> esercizi) {
        dati = new HashMap();
        for (int i = 0; i < esercizi.size(); i++) {
            dati.put("+" + i, trovaEsercizi((String)esercizi.get(i)));
        }
        return dati;
    }
}

```

Figura 33.11 Classe SchedaPersonalizzata.

```

package Allenamento;

import java.io.InputStream;
import java.util.HashMap;
import java.util.Map;

public class SchedaCorso extends Dario {

    private Map dati;

    public Map prelevaCorsi() {
        dati = new HashMap();
        InputStream richiesta = getRichieste().GetRichiesta("/scelta/prelevaCorsi", dati, null);
        dati.clear();
        dati = getJson().LeggiJson(richiesta);

        return dati;
    }

    public String ottieniEmail(String nomecorso) {
        //CREA UN HASHMAP CONTENENTE I VALORI
        dati = new HashMap();
        dati.put("nomecorso", nomecorso);
        //EFFETTUO LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
        InputStream richiesta = getRichieste().GetRichiesta("/scelta/ottieniEmail", dati, null);
        //FUSOLICE L'HASHMAP
        dati.clear();
        //INSEGNICE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'HASHMAP
        dati = getJson().LeggiJson(richiesta);

        String email = (String)dati.get("email");

        return email;
    }
}

```

Figura 33.12 Classe SchedaCorso..

Oltre alle classi descritte in precedenza il team ha creato altre classi che godono delle caratteristiche dell'ereditarietà. Il team all'interno del package **Allenamento** ha creato una superclasse nominata **PartiCorpo** tale classe è estesa da ben cinque sottoclassi figlie.

Il team ha deciso di agire in questo modo per diverse ragioni. Innanzitutto uno dei motivi per cui si usa il paradigma OOP riguarda la riusabilità del codice, infatti grazie all'utilizzo dell'ereditarietà abbiamo potuto implementare dei metodi nelle nostre superclassi che sono stati poi ereditati da tutte le sottoclassi evitando la riscrittura di codice. Un altro motivo per cui si è deciso di agire in questo modo riguarda la manutenibilità del codice, infatti grazie all'ereditarietà i metodi sono più facili da debuggarre infatti più volte durante la fase di testing nel momento in cui sono apparsi

dei bug sono stati risolti in maniera molto veloce andando ad agire direttamente nei metodi. Infine un altro motivo per cui l'ereditarietà è stata utile è l'ampliamento del sistema. Infatti nel caso in cui si vogliano aggiornare le persone oppure le parti del corpo che riguardano il nostro sistema basterà creare una nuova sottoclass che estende la superclasse, quindi il nostro sistema risulta altamente incline ai cambiamenti, ed inoltre essendo facile apportare cambiamenti il ciclo di vita del nostro sistema ne giova sicuramente.

### Interfaccie e Polimorfismo.

Una delle regole che va rispettata se si vuole creare un sistema che segue le caratteristiche della OOP è quello di evitare ove possibile il cosiddetto “code smells”. Esso è generato dalla presenza di molti cicli switch che rendono il codice difficile da modificare, manutenere e a basse prestazioni. Nel nostro sistema per evitare tutto ciò abbiamo sfruttato le caratteristiche del polimorfismo. Ci siamo serviti di interfaccie(descritte nel prossimo paragrafo), di superclassi e sottoclassi.

La signature dei metodi è presente all'interno delle interfaccie, invece il corpo all'interno delle classi che implementano l'interfaccia. Ogni classe al suo interno ha i metodi che la riguardano. Questi metodi poi vengono richiamati all'interno di un'altra classe attraverso l'utilizzo delle HashMap, infatti sfruttando questo costrutto di Java è possibile richiamare una classe ed i rispettivi metodi solo nel momento in cui essa serve.

Sfruttando queste caratteristiche offerte dal polimorfismo oltre ad evitare il costrutto switch il nostro sistema è molto più facile da manutenere, nel caso in cui si voglia ampliare il sistema non si dovrà agire sul costrutto switch aggiungendo un caso ma si potrà tranquillamente creare una nuova classe e fargli implementare l'interfaccia. Altro vantaggio è dato dalla flessibilità infatti se si vuole aggiungere un nuovo tipo, bisogna aggiornare tutti i condizionali, ma con le sottoclassi basta solo creare una nuova sottoclass e fornire i metodi appropriati.

Un vantaggio di questo approccio è che nuove azioni possono essere supportate scrivendo (e non modificando) il codice esistente. La sottoclass che si creerà supporterà efficacemente l'aggiunta di nuove funzionalità aggiungendo un nuovo concetto, senza modificare l'implementazione esistente. È chiaro che, se la nuova funzionalità richiede l'aggiunta di un nuovo metodo polimorfico alla gerarchia, saranno necessarie modifiche.

Mostriamo in seguito degli snippet che dimostrano le cose descritte sopra:

### Utilizzo del polimorfismo nel package Autenticazione:

```

package Autenticazione;

public interface LoginInterface {
    public void accedi(String email, String password);
}

}

```

Figura 33.13 Interfaccia utile per il login.

```

package Autenticazione;

public class ControlloAccesso {

    private String email;
    private String password;

    public ControlloAccesso(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public void accedi(LoginInterface accesso){
        accesso.accedi(this.email, this.password);
    }

}

```

Figura 33.14 Classe ControlloAccesso.

```

public class Utente extends Persona implements LoginInterface{

    private String questionario;
    private String esito;
    private double risultato = 0;
    private Map dati;

    public Utente(){
        this.questionario = getXmlpars2().getElement("Questionario");
        this.esito = getXmlpars2().getElement("Esito_Dottore");
    }

    @Override
    public void accedi(String email, String password) {

        Xml xml = new Xml();
        xml.scriviXmlDati(email);
        xml.salvaXmlMisure(email);
        //MOSTRA FINESTRA DASHBOARD UTENTE
        Pannello_Utente dash = new Pannello_Utente();
        dash.setVisible(true);
    }
}

```

Figura 33.15 Classe che implemta l'interfaccia.

```

private void AccessoMouseClicked(java.awt.event.MouseEvent evt) {
    // BOTONE ACCEDI
    Persona verifica = new Persona(emailcampo.getText(), passwordcampo.getText());
    //ControlloAccesso c = new ControlloAccesso(emailcampo.getText(), passwordcampo.getText());

    String msg = verifica.controllo(emailcampo.getText(), passwordcampo.getText());
    try {
        if (msg.equals("Credenziali errate.")) {
            passwordcampo.setText("");
            JOptionPane.showMessageDialog(null, "Credenziali Errate", "Login", JOptionPane.WARNING_MESSAGE);
        } else{
            verifica.accedi((LoginInterface) Class.forName("Autenticazione." + msg).newInstance();
            dispose();
        }
    } catch (ClassNotFoundException | InstantiationException | IllegalAccessException ex) {
        Logger.getLogger(Login.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Figura 33.16 Richiamo del metodo.

## Utilizzo del polimorfismo nel package Allenamento:

```
package Allenamento;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

public interface Interfaccia {

    public Map assegnaEsercizi(Esercizio product);

    public List<Object> assegnaNumEsercizi (Momenti momenti);

    public List<ArrayList> calcoloCalorie(Intensita intensita);

    public Map assegnaCalorie(AssegnaCalorie assegna);

}
```

Figura 33.17 Interfaccia

```
public class Momenti {
    private final List<String> primomomento;
    private final List<Object> esercizi;
    private final List<Object> esercizi1mom;
    private final List<Object> esercizi2mom;
    private final List<Object> esercizi3mom;
    private final int numeromomento;
    private final int sizePartiOrdinate;
    private final int[] unoDue = new int[]{ 1,2,5 };
    private final int[] tre = new int[]{ 3,5,7 };
    private final int[] quattro = new int[]{ 4,7,9 };
    private final int[] cinque = new int[]{ 5,9,10 };
    private final String[] intensita = new String[]{"Leggero", "Medio", "Pesante"};
    private final Map dati = new HashMap(){
        put(1, getHodou());
        put(2, getHodou());
        put(3, getTre());
        put(4, getQuattro());
        put(5, getCinque());
    };

    public Momenti(List<String> primomomento, List<Object> esercizi, List<Object> esercizi1mom, List<Object> esercizi2mom, List<Object> esercizi3mom,
        this.primomomento = primomomento;
        this.esercizi = esercizi;
        this.esercizi1mom = esercizi1mom;
        this.esercizi2mom = esercizi2mom;
        this.esercizi3mom = esercizi3mom;
        this.numeromomento = numeromomento;
        this.sizePartiOrdinate = sizePartiOrdinate;
    }
}
```

Figura 33.18 Classe Momenti.

```
public Map esercizi (int numeroMomento, List<String> primomomento, List<Object> partiOrdinate, String tipo){

    List<Object> esercizi = new ArrayList<>();
    List<Object> esercizi1mom = new ArrayList<>();
    List<Object> esercizi2mom = new ArrayList<>();
    List<Object> esercizi3mom = new ArrayList<>();
    int sizePartiOrdinate = partiOrdinate.size()/2;

    Momenti momenti = new Momenti(primomomento, esercizi, esercizi1mom, esercizi2mom, esercizi3mom, numeroMomento, sizePartiOrdinate);
    esercizi = numeroEse(numeroMomento, momenti);
    esercizi.addAll(partiOrdinate);
}
```

Figura 33.19 Metodo per richiamare la classe Momenti.

```
private HashMap <Integer, Interfaccia> nMomemnti = new HashMap() {
    put(1, new UnMomento());
    put(2, new DueMomenti());
    put(3, new TreMomenti());
};

public List<Object> numeroEse(int numeroMomento, Momenti momenti ){
    return nMomemnti.get(numeroMomento).assegnaNumEsercizi(momenti);
}
```

Figura 33.20 HashMap per gestire i casi.

## Astrazione.

Il team ha utilizzato tale caratteristica OOP. Le classi astratte sono molto utili nella modellazione. In un certo senso, esse definiscono un “vocabolario” di alto livello che arricchisce il linguaggio della modellazione. Una classe astratta permette di definire alcune operazioni comuni, lasciando alle sue sottoclassi l’implementazioni di operazioni specifiche. Tra le classi astratte all’interno del nostro sistema troviamo la superclasse astratta **“Messaggi”**(figura 33.11).

```
package Consultazioni;

import Impostazioni.UGOSSI;
import Impostazioni.Richieste;
import Impostazioni.Xmpars;
import java.util.Map;

public abstract class Messaggi {
    private final Xmpars xmppars = new Xmpars("Impostazioni.xml");
    private final Richieste richieste = new Richieste(xmppars.getElement("RICHIESTE"), xmppars.getElement("SERVER_ACCESS"), xmppars.getElement("SERVER_PORTA"));
    private final JSON json = new JSON();

    public Messaggi() {
        //INIZIALIZZA LA VARIABILE XMPPARS CON IL FILE IMPOSTAZIONI.XML
        if (!xmppars.exists()) xmppars.ImpDefaut();
    }

    public abstract Map visualizzaDomande(String tabella, String email, String coach);

    public abstract String inviaRisposta(String id, String risposta);

    public abstract String mediaVoti(String coach);

    public abstract float prelevaMediaVoto(String tabella);
}
```

Figura 33.21 Classe astratta Messaggi.

Tale classe presenta due sottoclassi che ereditano i metodi della loro superclasse. Nella figura 33.11 vediamo un piccolo estratto della sottoclasse **MessaggiMedico** e nella figura 33.12 della sottoclasse **MessaggiTrainer**.

```
public class MessaggiMedico extends Messaggi{
    private Medico medico;
    private Map dati;
    @Override
    public Map visualizzaDomande(String tabella, String email, String coach) {
        dati = new HashMap();
        dati.put("tabella", tabella);
        dati.put("email", email);
        dati.put("coach", coach);
        //ESEGUITE LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
        InputStream richiesta = getRichieste().GetRichieste("/consultazione/visualizzaDomande", dati, null);
        //ESULSA L'INPUTMAP
        dati.clear();
        //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'INPUTMAP
        dati = getJson().LeggiJson(richiesta);
        return dati;
    }
    @Override
    public String inviaRisposta(String id, String risposta) {
        String msg = "Risposta inviata!";
        if (risposta.length() < 250) {
            dati = new HashMap();
            dati.put("id", id);
            dati.put("risposta", risposta);
            InputStream richiesta = getRichieste().GetRichieste("/consultazione/inviaRisposta", dati, null);
            dati.clear();
            dati = getJson().LeggiJson(richiesta);
            String result = (String) dati.get("result");
            if (result.equals("Risposta inviata")) {
                msg = "Risposta inviata!";
            } else {
                msg = "Risposta non inviata! La domanda contiene più di 250 caratteri!";
            }
        }
        return msg;
    }
}
```

Figura 33.22 Sottoclasse MessaggiMedico.

```
public class MessaggiTrainer extends Messaggi{
    private Trainer trainer;
    private Map dati;
    @Override
    public Map visualizzaDomande(String tabella, String email, String coach) {
        dati = new HashMap();
        dati.put("tabella", tabella);
        dati.put("email", email);
        dati.put("coach", coach);
        //ESEGUITE LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
        InputStream richiesta = getRichieste().GetRichieste("/consultazione/visualizzaDomande", dati, null);
        //ESULSA L'INPUTMAP
        dati.clear();
        //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'INPUTMAP
        dati = getJson().LeggiJson(richiesta);
        return dati;
    }
    @Override
    public String inviaRisposta(String id, String risposta) {
        String msg = "Risposta inviata!";
        if (risposta.length() < 250) {
            dati = new HashMap();
            dati.put("id", id);
            dati.put("risposta", risposta);
            InputStream richiesta = getRichieste().GetRichieste("/consultazione/inviaRisposta", dati, null);
            dati.clear();
            dati = getJson().LeggiJson(richiesta);
            String result = (String) dati.get("result");
            if (result.equals("Risposta inviata")) {
                msg = "Risposta inviata!";
            } else {
                msg = "Risposta non inviata! La domanda contiene più di 250 caratteri!";
            }
        }
        return msg;
    }
}
```

Figura 33.23 Sottoclasse MessaggiTrainer.

In questo caso è stato nuovamente utilizzato l’override dei metodi. Come si può vedere dalle figure la superclasse astratta rispetta le regole dell’astrazione in quanto al suo interno troviamo solamente la signature dei

metodi, invece nelle sottoclassi troviamo l'implementazione. Grazie alla classe astratta il team ha risparmiato nella scrittura del codice, ha evitato codice duplicato e soprattutto ha potuto **riusare** il codice.

## Generalizzazione.

Il team si è anche servito di questa caratteristica OOP. La generalizzazione permette di creare delle relazioni tra oggetti di classi diverse.

La classe **Persona** presenta degli oggetti (nome, cognome ecc.), tali oggetti servono all'interno dei metodi presenti nella classe **Utente** che è una sottoclasse di Persona. Il team sfruttando la generalizzazione ha potuto utilizzare gli oggetti presenti in Persona nei metodi presenti in Utente evitando di ridichiarare gli oggetti una seconda volta.

Il team si è servito di questa caratteristica per far sì che un oggetto della sottoclasse possa essere usato ovunque sia permesso l'uso di un oggetto della superclasse. La generalizzazione permette di non ridefinire proprietà già definite nel nostro caso nome cognome ed email.

Gli attributi e i metodi già introdotti nella superclasse possono essere riusati nella sottoclasse. Nel caso dei metodi vogliamo specificare come è stata usata la generalizzazione:

- Il metodo *verificaEmail* (figura 33.24) della superclasse Persona è stato generalizzato nelle sottoclassi Utente e Admin. Metodo usato per verificare che non ci siano password uguali.
- Il metodo *verificaModificaPassword* (figura 33.25) della superclasse Persona è stato generalizzato nelle sottoclassi Utente e Admin. Metodo usato per verificare che la nuova password rispetti i requisiti in caso in cui si voglia modificare la vecchia password.
- Il metodo *eliminaAccount* della superclasse Persona è stato generalizzato nelle sottoclassi Utente e Admin. Metodo che permette l'eliminazione di un account.

```

public boolean verificaEmail(String email){

    String msg;
    //CREA UN HASHMAP CONTENENTE I VALORI
    Map dati = new HashMap();
    dati.put("email", email);
    dati.put("tabella", "credenziali");
    //
    //EFFECTUA LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
    InputStream richiesta = getRichieste().GetRichiesta("/controllo/emaildoppia", dati, null);
    //FUSLISCE L'HASHMAP
    dati.clear();
    //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'HASHMAP
    dati = getJson().LeggiJson(richiesta);
    //INSERISCO NELLA VARIABILE RESULT IL VALORE DELLA CHIAVE RESULT DELL'HASHMAP DATI
    String result = (String) dati.get("result");
    //STRING MSG E' UGUALE AL VALORE DELLA STRINGA RESULT
    msg = result;

    Boolean risultato;

    //RITORNA TRUE NEL CASO IN CUI L'EMAIL E' GIA' UTILIZZATA DA UN ALTRO UTENTE
    if (result.equals("Email Presente")) {
        risultato = true;
    }else{
        risultato = false;
    }

    return risultato;
}

```

Figura 33.24 Metodo verificaEmail.

```

public String verificaModificaPassword(String email, String vecchiapass, String nuovapass){

    String msg = "Password modificata con successo!";
    if (!email.isEmpty() && !vecchiapass.isEmpty() && !nuovapass.isEmpty()) {
        //CREA UN HASHMAP CONTENENTE LE SEGUENTI CHIAVI-VALORI
        Map dati = new HashMap();
        dati.put("email", email);
        dati.put("vecchiapass", vecchiapass);
        dati.put("nuovapass", nuovapass);
        dati.put("tabella", "credenziali");
        //

        //INSERISCO NELLA VARIABILE RESULT IL VALORE DELLA CHIAVE RESULT DELL'HASHMAP DATI
        InputStream richiesta = getRichieste().GetRichiesta("/impostazioni/verificaModificaPassword", dati, null);
        //FUSLISCE L'HASHMAP
        dati.clear();
        //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'HASHMAP
        dati = getJson().LeggiJson(richiesta);
        //INSERISCO NELLA VARIABILE RESULT IL VALORE DELLA CHIAVE RESULT DELL'HASHMAP DATI
        String result = (String) dati.get("result");
        msg = result;

        if(result.equals("Password non coincidono")){
            msg = "Password non coincidono";
        }else if(result.equals("Password modificata con successo")){
            msg = "Modifica password esecuita";
        }else{
            msg = "Campi password vuoti!";
        }
    }
    return msg;
}

```

Figura 33.25 Metodo VerificaModificaPassword.

La generalizzazione è stata anche usata nella superclasse astratta **Messaggi** infatti abbiamo:

- Il metodo *visualizzaDomande* nella superclasse Messaggi è stato generalizzato nelle sottoclassi Medico e Trainer. Esso permette di visualizzare le domande all'interno della jTable dedicata.
- Il metodo *visualizzaRisposte* nella superclasse Messaggi è stato generalizzato nelle sottoclassi Utente, Medico e Trainer. Esso permette di visualizzare le risposte all'interno della jTable dedicata.
- Il metodo *inviaRisposta* nella superclasse Messaggi è stato generalizzato nelle sottoclassi Medico e Trainer. Metodo che permette l'invio della risposta agli attori indicati.
- Il metodo *invia* nella superclasse Messaggi è stato generalizzato nelle sottoclassi Utente. Permette all'attore indicato di inviare domande al medico o ai trainer.
- Il metodo *mediaVoti* nella superclasse Messaggi è stato generalizzato nelle sottoclassi Medico e Trainer. Permette di calcolare e mostrare la media voti delle risposte date, agli attori interessati.

Grazie alla generalizzazione il team è stato facilitato nella specifica incrementale, nell'uso delle proprietà comuni tra classi ed una migliore localizzazione delle modifiche.

### Binding dinamico.

Questa caratteristica dipende dall'utilizzo del polimorfismo statico quindi **overload**. Nel nostro sistema come si può vedere nella figura 33.15 è stato usato all'interno della classe Admin.

```

public Map mostra(){
    //CREA UN HASHMAP CONTENENTE I VALORI
    dati = new HashMap();
    //EFFETTUO LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
    InputStream richiesta = richieste.GetRichiesta("/gestione/visualizzaRighe", dati, null);
    //FUSLISCE L'HASHMAP
    dati.clear();
    //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'HASHMAP
    dati = json.LeggiJson(richiesta);

    return dati;
}

public Map mostra (String trainer){
    //CREA UN HASHMAP CONTENENTE I VALORI
    dati = new HashMap();
    //EFFETTUO LA RICHIESTA AL SERVER UTILIZZANDO IL PATH SOTTOSTANTE
    InputStream richiesta = richieste.GetRichiesta("/gestione/prelevaTrainer", dati, null);
    //FUSLISCE L'HASHMAP
    dati.clear();
    //INSERISCE LA RISPOSTA DEL SERVER ALL'INTERNO DELL'HASHMAP
    dati = json.LeggiJson(richiesta);
    return dati;
}

```

Figura 33.15

Nel momento in cui il team ha richiamato tale metodo all'interno della classe che lo richiedeva l'ha potuto scegliere tra i suggerimenti dati dall'IDE tutto grazie al binding dinamico. Uno dei principali vantaggi appunto del Dynamic Binding è la flessibilità, grazie ad essa un singolo metodo può gestire diversi tipi di un oggetto in fase di esecuzione.

### 34. Quinta attività: testing.

Anche in questo terzo prototipo per effettuare il testing ci siamo serviti della metodologia unit-testing.

#### Invio dati diario.

- **Inserimento dati:** questo terzo prototipo dipende molto dal modo in cui vengono inseriti i dati, infatti se l'operazione non viene fatta nel modo giusto viene meno il funzionamento dell'intero sistema di generazione della scheda. Il team sapendo che l'inserimento dei dati è un'operazione molto delicata ha optato per ridurre al minimo gli input da tastiera, l'unico punto critico è l'immissione dei grammi che si vogliono perdere in caso di dimagrimento, in quel caso il team tramite degli opportuni metodi ha garantito che tutto ciò che non è un numero intero venga prontamente rifiutato. Nel caso in cui uno o più valori indispensabili relativi al diario siano stati omessi il team ha prontamente predisposto dei metodi che impediscono la creazione della scheda e comunicano all'utente tramite dei pop-up il problema.

- **Visualizzazione scheda:** il team si è assicurato che non ci siano problemi grafici nella visualizzazione della scheda e ciò non avviene.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Creazione scheda.

- **Visualizzazione dati diario:** il team si è assicurato che la tabella presentata al trainer non abbia problemi grafici di visualizzazione. La tabella risulta priva di bug, facile da leggere e da utilizzare.
- **Creazione scheda:** essendo un'operazione automatica il team ha solamente dovuto verificare che gli input che servono per la generazione della scheda siano corretti, come detto sopra avendo applicato le giuste precauzioni e regole l'algoritmo di automazione riceverà gli input proprio come se li aspetta.

Il team dopo aver fatto il testing ha ritenuto che i moduli presi in esame rispettano i requisiti.

### Modifica scheda.

- **Visualizzazione scheda:** il team ha progettato i frame per la visione della scheda in modo molto accurato evitando problemi di visualizzazione.
- **Modifica di uno o più esercizi:** nella modifica degli esercizi poteva sorgere un problema, ovvero che uno dei nostri trainer andava sbadatamente ad aggiungere troppi esercizi per una parte del corpo andando a violare i principi del nostro sistema. Per evitare una situazione del genere il team ha fatto in modo che all'interno dei combo-box dedicati siano visibili e selezionabili solo gli esercizi che riguardano quella parte del corpo, facendo ciò un esercizio può essere sostituito solo con uno che allena la stessa parte del corpo.

### Gestione corsi.

- **Inserimento nuovo corso:** durante l'inserimento di un nuovo corso potrebbe capitare l'inserimento di valori errati e/o non consentiti, il team per evitare ciò a predisposto dei metodi di controllo che rifiutano prontamente qualsiasi input non approvato.
- **Inserimento esercizi nuovo corso:** anche in questo caso sono presenti dei metodi che controllano l'inserimento corretto degli input richiesti.

- **Modifica esercizi vecchio corso:** la modifica consiste nell'inserimento di nuovi valori, perciò tutti i controlli sull'inserimento vengono fatti anche nella modifica.
- **Ampliamento esercizi vecchio corso:** quest'operazione consiste nell'inserimento di nuovi esercizi appunto perciò il team ha utilizzato gli stessi metodi di controllo usati per l'inserimento.
- **Visualizzazione corso:** il team ha testato che la rappresentazione grafica sia corretta e lo è, quindi l'admin non avrà problemi a visualizzare un corso e i suoi rispettivi esercizi.
- **Eliminazione corso ed esercizi:** in questo caso l'admin deve solamente scegliere il corso da eliminare, per evitare errori nell'eliminazione, l'admin deve prima aver confermato l'eliminazione.

## 35. Verifica.

Test 1: input diario.

Test	Input	Output
Inserimento valori per indicare i grammi da perdere.	Numeri interi positivi.	Invio dati accettato.
	Numeri negativi.	Invio dati rifiutato.
	Numeri decimali.	Invio dati rifiutato.

Test 2: creazione, modifica e ampliamento corso.

Test	Input	Output
Inserimento valori corso.	Stringhe, numeri.	Invio dati accettato.
	Spazi, simboli speciali, qualsiasi tipo di punteggiatura eccetto la virgola.	Invio dati rifiutato.

## 36. Ultima attività: DEPLOYMENT.

Diagramma di deployment.

Il diagramma di deployment rimane identico a quello presentato a pagina 83.

## 37. Diagramma delle fasi-workflow RUP.

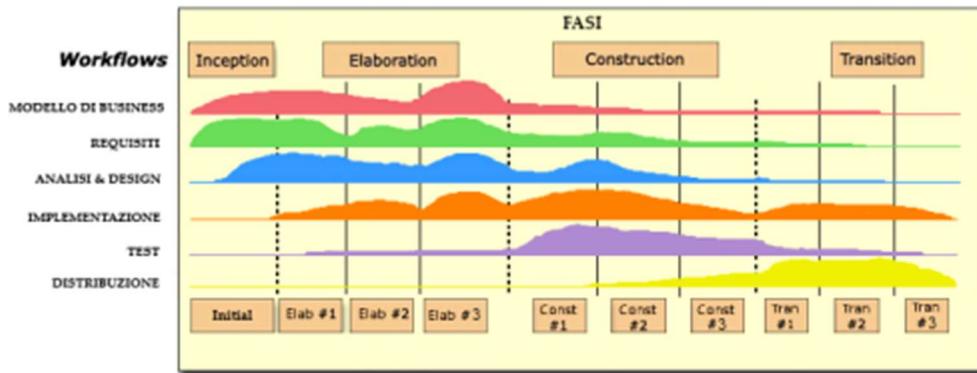


Figura 37 Workflow-phases diagram

### Installazione e Acceptance testing.

Dopo aver sviluppato la terza ed ultima iterazione il team ha provveduto ad installare il sistema nell'ambiente di sviluppo del cliente. Durante l'installazione e prova del sistema non si sono verificati nessun tipo di problemi o guasti, il sistema è funzionante, privo di errori, s'interfaccia bene con il sistema operativo del cliente (Windows) e riesce ad interoperare con gli altri sistemi già installati nel computer del nostro cliente.

Dopodiché il sistema è stato sottoposto all'acceptance testing ovvero il cliente ha iniziato ad usare il sistema per verificare che tutti i requisiti, funzionali e non siano stati rispettati. Dopo una attenta verifica il cliente ha confermato che tutte le funzionalità sono state implementate e non presentano guasti, inoltre tutti i vincoli imposti sono stati anch'essi rispettati. Perciò come da contratto dopo circa tre mesi di sviluppo il sistema è stato consegnato al cliente e il team ha ricevuto il pagamento per il lavoro svolto.

### Evoluzione e mantenimento.

Come stipulato da contratto il lavoro del team termina alla consegna della terza ed ultima iterazione. Il team perciò non dovrà apportare più nessuna modifica al sistema.

Come scritto nel contratto il team dovrà garantire per i primi tre mesi dalla consegna del sistema il mantenimento dello stesso, al termine del periodo prestabilito al sistema non verranno più fatti interventi di manutenzione.

## Bibliografia.

Leszek A. Maciaszeck – Sviluppo di Sistemi Informativi con UML (Analisi dei requisiti e progetto di sistema) – Assison-Wesley

Jim Arlow, Ila Neustadt – UML and Unified Process (Practical Object-Oriented Analisys and Design) – Booch Jacobson Rumbaugh (Edizione italiana a cura della Mc Graw Hill).

Lucidi del corso di Ingegneria del Software tenuto dal professor Salvatore Distefano.

Vari siti Internet.

Per i calcoli abbiamo consultato il seguente studio:

Casey Butt Ph.D - Your muscular potential: how to predict your Maximum muscular bodyweight and measurements.