

Università degli Studi di Napoli Federico II

**Corso di Laurea in Ingegneria Informatica**

**Corso di Ingegneria del Software**

**Prof. S. Russo - A.A. 2020 - 21**

***Progetto***

KeepFit

Gruppo: libre

Studenti:

Vittorio Libretti M63/1123 v.libretti@studenti.unina.it

Versione 2 del *17/01/2020*

**Indice**

1. Requisiti 4

1.1 Specifiche informali d’utente 4

1.2 Requisiti software 5

1.2.1 Requisiti funzionali 5

1.2.2 Requisiti sui dati 9

1.2.3 Requisiti non funzionali 10

1.2.4 Analisi nome/verbo 10

1.2.5 Tabella requisiti 11

2. Analisi e specifica dei requisiti 12

2.1 Modellazione dei casi d’uso 12

2.1.1 Aggiunta nuovo abbonato 12

2.1.2 Visualizzazione dettaglio abbonato 13

2.1.3 Visualizzazione elenco abbonati 13

2.1.4 Modifica informazioni abbonato 14

2.1.5 Rimozione abbonato 15

2.1.6 Promozione abbonato a premium 16

2.1.7 Visualizza elenco abbonati premium 17

2.1.8 Cancellazione account 18

2.1.9 Rinnovo abbonamento palestra 19

2.1.10 Visualizzazione dettaglio esercizio 19

2.1.11 Visualizzazione elenco esercizi 20

2.1.12 Visualizzazione elenco protocolli 21

2.1.13 Visualizzazione dettaglio protocollo 21

2.1.14 Diagramma dei casi d’uso 22

2.2 Diagramma delle classi 23

2.3 Diagrammi di sequenza 24

2.3.1 Visualizza elenco abbonati 24

2.3.2 Crea abbonato 24

3. Stima dei costi 26

3.1 Valutazione degli Unadjusted Function Point (UFP) 26

3.1.1 Ambito di conteggio 26

3.1.2 Identificazione funzioni dati e transazionali 26

3.2 Valutazione dei FP 27

3.1.3 Valutazione LLOC 27

3.1.4 Considerazioni 27

4. Piano di test funzionale 28

5. Progettazione 32

5.1 Diagramma delle classi 32

5.2 Diagrammi di sequenza 32

5.2.1 Visualizza elenco abbonati 32

5.2.2 Crea abbonato 33

6. Implementazione 35

6.1 Elenco package 35

6.2 Classi 35

6.3 Elenco artefatti 35

6.4 Javadoc funzionalità implementate 36

6.4.1 AdminBoundary 36

6.4.2 AdminBoundaryImpl 36

6.4.3 AdminController 37

6.4.4 AdminControllerImpl 37

6.4.5 AbbonatoDAO 38

6.4.6 AbbonatoDAOImpl 38

6.5 Diagramma di deployment 39

7. Testing 39

7.1 Test funzionale 39

7.2 Test strutturale 43

7.2.1 Complessità ciclomatica 43

7.2.2 Test di unità 44

7.3 Test di unità con JUnit 45

# Requisiti

## Specifiche informali d’utente

Si vuole realizzare un’applicazione per gestire gli abbonati di una palestra e per offrire loro un elenco di esercizi da consultare durante la loro sessione di allenamento.

Un utente amministratore, accedendo all’applicazione con username e password, può visualizzare un elenco di abbonati iscritti alla palestra. L’admin può aggiungere nuovi abbonati o modificare o cancellare abbonati precedentemente registrati. Di ogni abbonato si vuole visualizzare nome, cognome, data di nascita, stato dell’abbonamento (“attivo”, “scaduto”). Inoltre, selezionando un abbonato dalla lista, l’amministratore può visualizzare una scheda di dettaglio della persona: nome, cognome, codice fiscale, data di nascita, residenza e/o domicilio, data di iscrizione, telefono fisso e/o mobile, data di inizio e fine abbonamento.

Quando l’abbonamento della palestra scade, esso viene segnato come scaduto; l’amministratore provvederà a riattivarlo.

L’amministratore può promuovere gli abbonati a premium, in seguito al pagamento di un supplemento mensile. L’abbonato premium ha diritto ad avere un account personale nell’applicazione, al quale può eccedere tramite username e password. L’amministratore fornirà all’abbonato le credenziali di accesso. L’abbonato premium può visualizzare protocolli predefiniti di allenamento.

L’amministratore può visualizzare un elenco di account visualizzando username, nome e cognome dell’abbonato, data di nascita, stato dell’abbonamento premium (“in corso”, “scaduto”). L’account viene cancellato automaticamente quando scade l’abbonamento premium o manualmente dall’admin, e può essere ricreato dall’admin con il rinnovo.

Un protocollo di allenamento è un elenco di esercizi suddivisi per sessioni di allenamento. Ogni protocollo appartiene ad una specifica categoria determinata dall’obiettivo che si vuole raggiungere (“restare in forma”, “dimagrire”, “aumenta massa muscolare”, ecc).

Gli esercizi sono raggruppati per gruppo muscolare; ogni esercizio è caratterizzato da un nome, da una descrizione dei movimenti da eseguire, da un contenuto multimediale (immagine o video) che mostra visivamente l’esecuzione dell’esercizio.

Mediante delle postazioni dislocate in vari punti della palestra, l’abbonato ha la possibilità di utilizzare l’applicazione; le postazioni consistono in monitor touchscreen agganciati a colonnine o supporto a muro.

## Requisiti software

### Requisiti funzionali

###### Creazione nuovo abbonato

Introduzione

Viene consentito all’amministratore della palestra di registrare un abbonato.

Input

I dati obbligatori da inserire sono il nome, cognome, data di nascita, codice fiscale, data di inizio e fine dell’abbonamento, data di iscrizione; opzionalmente, l’indirizzo di domicilio e/o residenza, numero di telefono fisso e/o mobile.

Elaborazione

1. L’amministratore deve compilare la form di registrazione dell’abbonato;
2. L’amministratore deve salvare i dati della form;
3. Il sistema deve controllare che i dati inseriti siano ben formati e che il codice fiscale non sia già registrato;
4. Il sistema deve salvare i dati.

Output

I dati personali e dell’abbonamento vengono registrati nel sistema.

###### Visualizzazione elenco abbonati

Introduzione

Viene consentito all’amministratore di vedere l’elenco degli abbonati registrati, eventualmente applicando un filtro sul nome, cognome, stato dell’abbonamento attivo o scaduto.

Input

Nome, cognome, stato abbonamento.

Elaborazione

1. L’amministratore deve cliccare su un bottone per visualizzare la pagina con l’elenco degli abbonati;
2. Se la pagina è gia visualizzata si può utilizzare un filtro di ricerca;
3. Il sistema, in base all’esistenza o meno di uno o più filtri, deve controllare se ci sono abbonati che corrispondono alla ricerca.
4. Se esiste almeno un risultato esso viene mostrato;
5. Se non ci sono risultati il sistema stampa un messaggio di avviso.

Output

Nome, cognome, data di nascita, stato dell’abbonamento per ogni abbonato.

###### Visualizzazione dettaglio dell’abbonato

Introduzione

Viene consentito all’amministratore di visualizzare maggiori informazioni di un abbonato.

Input

Identificativo nel sistema dell’abbonato.

Elaborazione

1. Il sistema deve ricercare i dati di un abbonato mediante l’identificativo.
2. Il sistema deve stampare a video i dati dell’abbonato.

Output

L’amministratore visualizza i dati completi dell’abbonato.

###### Modifica dei dati dell’abbonato

Introduzione

L’amministratore vuole correggere e/o aggiornare i dati dell’abbonato.

Input

L’identificativo nel sistema dell’abbonato.

Elaborazione

1. L’amministratore deve selezionare un abbonato dall’elenco;
2. Il sistema deve conoscere l’identificativo dell’abbonato;
3. Il sistema deve ricercare i dati dell’abbonato;
4. Il sistema deve creare la form per la modifica;
5. I campi della form devono essere riempiti con i dati letti
6. L’amministratore deve poter modificare i campi della form;
7. L’amministratore deve salvare i dati nella form;
8. Il sistema deve controllare se i campi aggiornati presentano errori;
9. Il sistema deve avvisare l’amministratore se ci sono dati scorretti.

Output

L’amministratore riesce a salvare le modifiche.

###### Cancellazione abbonato

Introduzione

Viene consentito all’amministratore di cancellare un abbonato.

Input

L’identificativo nel sistema dell’abbonato.

Elaborazione

1. L’amministratore deve selezionare un bottone;
2. Il sistema deve ricercare l’abbonato con il suo identificativo e cancellarlo.

Output

L’abbonato viene cancellato dal sistema.

###### Creazione account per l’abbonato

Introduzione

Viene consentito all’amministratore di creare un account per l’abbonato.

Input

L’identificativo dell’abbonato.

Elaborazione

1. L’amministratore deve associare ad un abbonato registrato un abbonamento premium;
2. L’amministratore deve poter creare un account solo dopo aver creato l’abbonamento premium;
3. L’account deve essere associato all’abbonato;
4. L’amministratore deve fornire una username ed una password;
5. Il sistema deve controllare se le credenziali inserite non esistano per creare l’account;

Output

L’account associato ad un abbonato.

###### Visualizzazione dell’elenco degli account

Introduzione

Viene consentito all’amministratore di visualizzare l’elenco degli account.

Input

Nome, cognome, username.

Elaborazione

1. L’amministratore deve premere un bottone per visualizzare l’elenco;
2. Il sistema deve ricercare tutti gli account creati secondo il criterio di ricerca;
3. Il sistema deve mostrare gli account.

Output

Nome, cognome, username, data scadenza account per ogni account.

###### Visualizzazione di un account

Introduzione

Viene consentito all’amministratore di vedere tutte i dati dell’account

Input

L’identificativo nel sistema dell’account.

Elaborazione

1. L’amministratore deve premere un bottone per vedere più informazioni dell’account;
2. Il sistema deve cercare l’account in base all’identificativo;
3. Il sistema deve mostrare tutte le informazioni dell’account;

Output

Tutte le informazioni associate all’account.

###### Cancellazione account

Introduzione

Il sistema automaticamente cancella l’account allo scadere dell’abbonamento premium.

Input

L’identificativo dell’account.

Elaborazione

1. Il sistema deve controllare se esiste un abbonamento premium scaduto;
2. Il sistema deve cancellare l’account;
3. Il sistema deve cancellare l’abbonamento premium associato all’account.

Output

L’account e l’abbonamento premium vengono cancellati.

###### Visualizzazione esercizio

Introduzione

Viene consentito ad un abbonato di visualizzare il tutorial di un esercizio.

Input

Identificativo dell’esercizio.

Elaborazione

1. Il sistema ricerca l’esercizio tramite identificativo;
2. Il sistema visualizza a video informazioni dettagliate dell’esercizio

Output

L’abbonato visualizza il tutorial dell’esercizio.

###### Visualizzazione elenco degli esercizi

Introduzione

Viene consentito all’abbonato di visualizzare l’elenco degli esercizi.

Input

Nome, gruppo muscolare dell’esercizio.

Elaborazione

1. L’abbonato deve cliccare su un bottone per visualizzare la pagina con l’elenco degli esercizi;
2. Se la pagina è gia visualizzata si può utilizzare un filtro di ricerca;
3. Il sistema, in base all’esistenza o meno di uno o più filtri, deve controllare se ci sono esercizi che corrispondono alla ricerca.
4. Se esiste almeno un risultato esso viene mostrato;
5. Se non ci sono risultati il sistema stampa un messaggio di avviso.

Output

L’abbonato visualizza gli esercizi in base alla ricerca effettuata.

###### Visualizzazione di un protocollo

Introduzione

Viene consentito ad un abbonato con account di visualizzare un protocollo di allenamento personalizzato o generico.

Input

L’identificativo del protocollo.

Elaborazione

1. Il sistema ricerca il protocollo tramite identificativo;
2. Il sistema visualizza a video informazioni dettagliate del protocollo di allenamento.

Output

L’abbonato con account visualizza il protocollo di allenamento.

###### Visualizzazione dell’elenco dei protocolli di allenamento

Introduzione

Viene consentito ad un abbonato con account di consultare un elenco di protocolli di allenamento.

Input

Categoria.

Elaborazione

1. L’abbonato deve cliccare su un bottone per visualizzare la pagina con l’elenco dei protocolli;
2. Se la pagina è gia visualizzata si può utilizzare un filtro di ricerca;
3. Il sistema, in base all’esistenza del filtro, deve controllare se ci sono protocolli che corrispondono alla ricerca.
4. Se esiste almeno un risultato esso viene mostrato;
5. Se non ci sono risultati il sistema stampa un messaggio di avviso.

Output

L’abbonato con account visualizza i protocolli in base al criterio di ricerca selezionato.

### Requisiti sui dati

###### Abbonato

Il sistema deve conservare i seguenti dati dell’abbonato:

1. Nome;
2. Cognome;
3. Data di nascita;
4. Codice fiscale;
5. Indirizzo di residenza;
6. Indirizzo di domicilio;
7. Numero di telefono fisso;
8. Numero di telefono mobile;
9. Data di iscrizione.

###### Amministratore

Il sistema deve conservare i seguenti dati dell’amministratore:

1. Username;
2. Password.

###### Account abbonato

Il sistema deve conservare i seguenti dati dell’account:

1. Username;
2. Password;

###### Abbonamento

Il sistema deve conservare i seguenti dati dell’abbonamento:

1. Data di inizio;
2. Data di fine;

###### Abbonamento premium

Il sistema deve conservare i seguenti dati dell’abbonamento premium:

1. Data di inizio;
2. Data di fine;

###### Esercizio

Il sistema deve conservare i seguenti dati dell’esercizio:

1. Nome;
2. Descrizione;
3. Gruppo muscolare;
4. Contenuto multimediale (video o immagine);

###### Protocollo di allenamento

Il sistema deve conservare i seguenti dati del protocollo di allenamento:

1. Categoria;
2. Nome della sessione di allenamento;

### Requisiti non funzionali

###### Tecnologie di implementazione

Il sistema software dovrà essere implementato utilizzando il linguaggio Java con piattaforma di sviluppo JDK 1.8.

###### Interfaccia utente

Tutte le interfacce verso l’utente devono essere realizzate in linguaggio html. Le interfacce devono adattarsi a qualsiasi dimensione dello schermo.

###### Usabilità

L’applicazione deve avere delle interfacce semplici da utilizzare.

###### Caratteristiche utente

Il sistema software “Keepfit” è rivolto ad una utenza con una discreta conoscenza informatica. All’utente è richiesta solamente una conoscenza informatica di base (alfabetizzazione informatica).

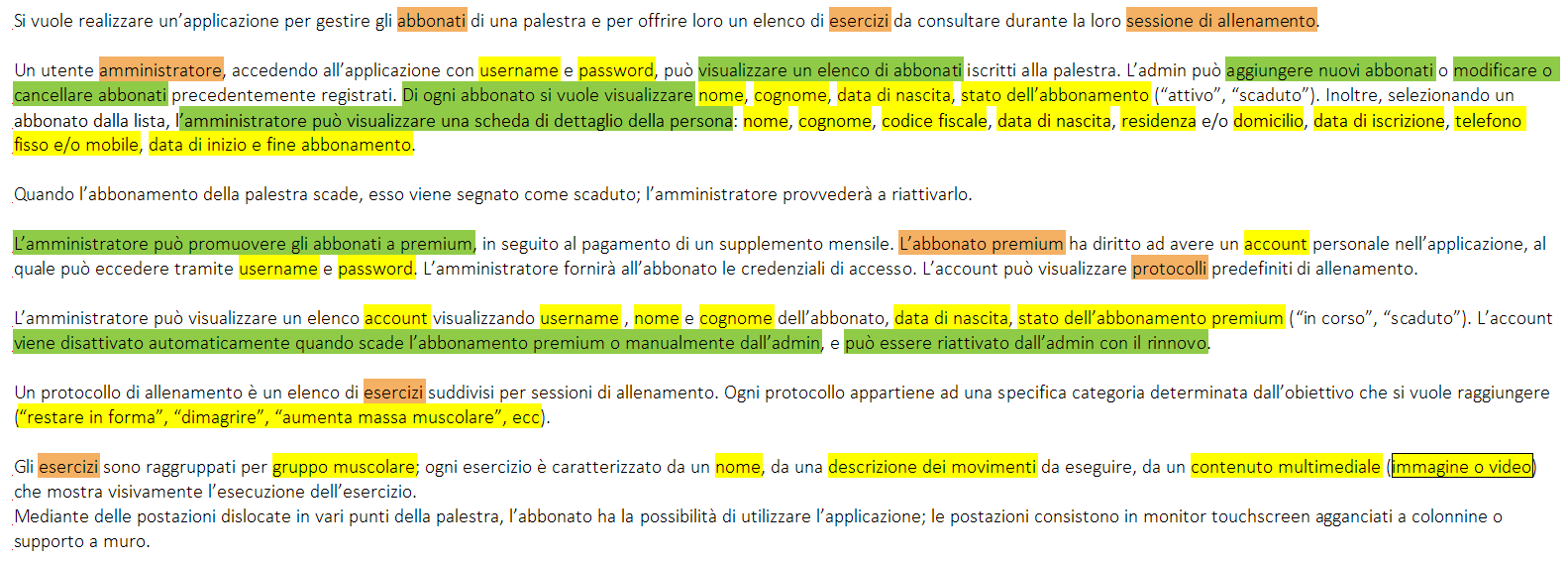
###### Persistenza dei dati

I dati devono essere salvati su database relazionale. Per garantire la massima flessibilità sulla scelta dal database verranno utlizzati lo standard JPA e il framework Hibernate.

###### Logging

Il sistema deve registrare le operazioni critiche per scopo di tracciamento e debug.

### Analisi nome/verbo



### Tabella requisiti

|  |  |
| --- | --- |
| Requisito | Descrizione |
| RF01 | Viene consentito all’amministratore della palestra di registrare un abbonato |
| RF02 | Viene consentito all’amministratore di vedere l’elenco degli abbonati registrati, eventualmente applicando un filtro sul nome, cognome, stato dell’abbonamento attivo o scaduto |
| RF03 | Viene consentito all’amministratore di visualizzare maggiori informazioni di un abbonato |
| RF04 | L’amministratore vuole correggere e/o aggiornare i dati dell’abbonato |
| RF05 | Viene consentito all’amministratore di cancellare un abbonato |
| RF06 | Viene consentito all’amministratore di creare un account per l’abbonato |
| RF07 | Viene consentito all’amministratore di visualizzare l’elenco degli account |
| RF08 | Viene consentito all’amministratore di vedere tutte i dati dell’account |
| RF09 | Il sistema automaticamente cancella l’account allo scadere dell’abbonamento premium |
| RF10 | Viene consentito ad un abbonato di visualizzare il tutorial di un esercizio |
| RF11 | Viene consentito all’abbonato di visualizzare l’elenco degli esercizi |
| RF12 | Viene consentito ad un abbonato con account di visualizzare un protocollo di allenamento personalizzato o generico |
| RF13 | Viene consentito ad un abbonato con account di consultare un elenco di protocolli di allenamento |
| RD01 | Il sistema deve conservare i seguenti dati dell’abbonato:   1. Nome; 2. Cognome; 3. Data di nascita; 4. Codice fiscale; 5. Indirizzo di residenza; 6. Indirizzo di domicilio; 7. Numero di telefono fisso; 8. Numero di telefono mobile; 9. Data di iscrizione. |
| RD02 | Il sistema deve conservare i seguenti dati dell’amministratore:   1. Username; 2. Password. |
| RD03 | Il sistema deve conservare i seguenti dati dell’account:   1. Username; 2. Password; |
| RD04 | Il sistema deve conservare i seguenti dati dell’abbonamento:   1. Data di inizio; 2. Data di fine; |
| RD05 | Il sistema deve conservare i seguenti dati dell’abbonamento premium:   1. Data di inizio; 2. Data di fine; |
| RD06 | Il sistema deve conservare i seguenti dati dell’esercizio:   1. Nome; 2. Descrizione; 3. Gruppo muscolare; 4. Contenuto multimediale (video o immagine); |
| RD07 | Il sistema deve conservare i seguenti dati del protocollo di allenamento:   1. Categoria; 2. Nome della sessione di allenamento; |
| RNF01 | Il sistema software dovrà essere implementato utilizzando il linguaggio Java con piattaforma di sviluppo JDK 1.8 |
| RNF02 | Tutte le interfacce verso l’utente devono essere realizzate in linguaggio html. Le interfacce devono adattarsi a qualsiasi dimensione dello schermo |
| RNF03 | L’applicazione deve avere delle interfacce semplici da utilizzare |
| RNF04 | Il sistema software “Keepfit” è rivolto ad una utenza con una discreta conoscenza informatica. All’utente è richiesta solamente una conoscenza informatica di base (alfabetizzazione informatica) |
| RNF05 | I dati devono essere salvati su database relazionale. Per garantire la massima flessibilità sulla scelta dal database verranno utlizzati lo standard JPA e il framework Hibernate |
| RNF06 | Il sistema deve registrare le operazioni critiche per scopo di tracciamento e debug. |

# Analisi e specifica dei requisiti

## Modellazione dei casi d’uso



### Aggiunta nuovo abbonato

|  |  |
| --- | --- |
| **UC\_KF\_01** | |
| **Descrizione** | | L’utente admin aggiunge un nuovo abbonato nel sistema | |
| **Attore Primario** | | Amministratore | |
| **Attori Secondari** | |  | |
| **Precondizioni** | | E’necessario che:   * L’utente admin sia precedentemente autenticato; | |
| **Postcondizione di successo** | | L’admin aggiunge un nuovo abbonato nel sistema | |
| **Postcondizione di fallimento** | | L’utente non riesce ad aggiungere un nuovo abbonato | |
| **Step** | **Azione** | | |
| 1 | L’admin richiede all’applicazione di registrare un nuovo abbonato | | |
| 2 | L’admin inserisce le informazioni richieste (anagrafica+data inizio e fine abbonamento) | | |
| 3 | L’admin invia i dati inseriti | | |
| 4 | Il sistema controlla la correttezza dei dati inseriti | | |
| 5 | If abbonato non esiste gia Il sistema salva i dati | | |
| 6 | Il sistema conferma la registrazione del nuovo abbonato | | |
| **Alternativa 1** | |  | |
| **Step** | **Azione** | | |
| 1 | L’admin non inserisce tutti i dati obbligatori e/o dati scorretti | | |
| 2 | L’admin invia i dati inseriti | | |
| 3 | Il sistema controlla la correttezza dei dati inseriti | | |
| 4 | Il sistema avvisa l’utente di dati mancanti o sbagliati | | |
| **Alternativa 2** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin abbandona la schermata di creazione di un nuovo abbonato | | |
| **Alternativa 3** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin inserisce i dati di un abbonato gia esistente | | |
| 2 | L’admin invia i dati inseriti | | |
| 3 | Il sistema controlla la correttezza dei dati inseriti | | |
| 4 | Il sistema rileva che esiste gia l’abbonato | | |
| 5 | Il sistema mostra all’admin un messaggio di errore | | |

### Visualizzazione dettaglio abbonato

|  |  |
| --- | --- |
| **UC\_KF\_02** | |
| **Descrizione** | | L’utente admin visualizza un abbonato |
| **Attore Primario** | | Amministratore |
| **Attori Secondari** | |  |
| **Precondizioni** | | E’necessario che:   * L’utente admin sia precedentemente autenticato; * L’utente admin abbia visualizzato l’elenco degli abbonati |
| **Postcondizione di successo** | | L’admin riesce a visualizzare le informazioni di un abbonato |
| **Postcondizione di fallimento** | | L’admin non riesce a visualizzare le informazioni dell’abbonato |
| **Step** | **Azione** | |
| 1 | L’admin seleziona un abbonato dall’elenco | |
| 2 | Il sistema ricerca l’abbonato utilizzando il suo identificativo | |
| 3 | Il sistema mostra il risultato della ricerca | |

### Visualizzazione elenco abbonati

|  |  |
| --- | --- |
| **UC\_KF\_03** | |
| **Descrizione** | | L’utente admin visualizza l’elenco degli abbonati | |
| **Attore Primario** | | Amministratore | |
| **Attori Secondari** | |  | |
| **Precondizioni** | | E’necessario che:   * L’utente admin sia precedentemente autenticato; | |
| **Postcondizione di successo** | | L’admin riesce a visualizzare l’elenco degli abbonati | |
| **Postcondizione di fallimento** | | L’admin non riesce a visualizzare l’elenco degli abbonati | |
| **Step** | **Azione** | | |
| 1 | L’admin accede alla schermata per visualizzare l’elenco degli abbonati | | |
| 2 | Il sistema automaticamente ricerca tutti gli abbonati registrati | | |
| 3 | If risultato ricerca non è vuoto il sistema mostra tutti gli abbonati registrati  Else il sistema avvisa che non esistono abbonati registrati | | |
| **Alternativa 1** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin inserisce lo stato dell’abbonamento e/o il nome e/o il cognome | | |
| 2 | Il sistema ricerca gli abbonati | | |
| 2 | Per ogni abbonato registrato, se i criteri di ricerca corrispondono, il sistema seleziona l’abbonato | | |
| 3 | If elenco degli abbonati selezionati non è vuoto, il sistema stampa l’elenco  Else il sistema avvisa che non è stato trovato alcun risultato | | |

### Modifica informazioni abbonato

|  |  |  |
| --- | --- | --- |
| **UC\_KF\_04** | | |
| **Descrizione** | | | L’utente admin vuole modificare le informazioni di un abbonato |
| **Attore Primario** | | | Amministratore |
| **Attori Secondari** | | |  |
| **Precondizioni** | | | E’necessario che:   * L’utente admin sia precedentemente autenticato; * L’utente admin abbia visualizzato l’elenco degli abbonati o il dettaglio dell’abbonato |
| **Postcondizione di successo** | | | L’admin riesce a salvare le modifiche |
| **Postcondizione di fallimento** | | | L’admin non riesce a salvare le modifiche |
| **Step** | **Azione** | | |
| 1 | L’admin abilita la modifica dell’abbonato | | |
| 2 | L’admin cambia le informazioni anagrafiche e/o la data di inizio e/o la data di fine dell’abbonamento | | |
| 3 | L’admin invia i dati inseriti | | |
| 4 | Il sistema controlla la correttezza dei dati inseriti:   * Dati anagrafici * Data inizio abbonamento < data fine abbonamento * Data fine abbonamento non è una data futura | | |
| 5 | Il sistema aggiorna i dati dell’abbonato e/o dell’abbonamento | | |
| 6 | Il sistema comunica il successo dell’operazione | | |
| **Alternativa 1** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin inserisce dati scorretti | | |
| 2 | Il sistema controlla la correttezza dei dati inseriti:   * Dati anagrafici * Data inizio abbonamento < data fine abbonamento * Data fine abbonamento è una data futura | | |
| 3 | Il sistema rileva un errore nei dati | | |
| 4 | Il sistema mostra un messaggio di errore | | |
| **Alternativa 2** | |  | |
| **Step** | Azione | | |
| 1 | L’admin abbandona le modifiche | | |
| 2 | Il sistema non salva le modifiche sui dati | | |

### Rimozione abbonato

|  |  |
| --- | --- |
| **UC\_KF\_05** | |
| **Descrizione** | | L’utente admin vuole cancellare un abbonato |
| **Attore Primario** | | Amministratore |
| **Attori Secondari** | |  |
| **Precondizioni** | | E’necessario che:   * L’utente admin sia precedentemente autenticato; * L’utente admin abbia visualizzato la lista degli abbonati o il dettaglio dell’abbonato * L’abbonato esiste |
| **Postcondizione di successo** | | L’admin riesce a cancellare l’abbonato dal sistema |
| **Postcondizione di fallimento** | | L’admin non riesce a cancellare l’abbonato dal sistema |
| **Step** | **Azione** | |
| 1 | L’admin richiede la cancellazione dell’abbonato | |
| 2 | Il sistema ricerca l’abbonato tramite il suo identificativo | |
| 3 | Il sistema cancella l’abbonato dal sistema | |
| 4 | Il sistema avvisa che l’abbonato è stato cancellato | |

### Promozione abbonato a premium

|  |  |  |
| --- | --- | --- |
| **UC\_KF\_06** | | |
| **Descrizione** | | | L’utente admin vuole promuovere un nuovo abbonato a premium |
| **Attore Primario** | | | Amministratore |
| **Attori Secondari** | | |  |
| **Precondizioni** | | | E’necessario che:   * L’utente admin sia precedentemente autenticato; * L’utente admin abbia visualizzato la lista degli abbonati o il dettaglio dell’abbonato * L’abbonato non abbia l’abbonamento premium |
| **Postcondizione di successo** | | | L’admin riesce a creare l’utenza per l’abbonato premium |
| **Postcondizione di fallimento** | | | L’admin non riesce a creare l’utenza per l’abbonato premium |
| **Step** | **Azione** | | |
| 1 | L’admin richiede l’associazione di un abbonamento premium al sistema | | |
| 2 | Il sistema mostra una schermata per impostare la data di inizio e fine dell’abbonamento premium | | |
| 3 | L’admin inserisce la data di inizio e fine | | |
| 4 | L’admin invia i dati | | |
| 5 | Il sistema controlla la correttezza dei dati:   * Data inizio abbonamento < data fine abbonamento * Data fine abbonamento è una data futura | | |
| 6 | Il sistema mostra la schermata per impostare le credenziali di accesso per l’abbonato premium | | |
| 7 | L’admin inserisce username e password | | |
| 8 | L’admin invia i dati | | |
| 9 | Il sistema controlla la correttezza di username e password | | |
| 10 | Il sistema controlla se la username e password sono gia utilizzate | | |
| 11 | Il sistema marca l’abbonato come premium | | |
| 12 | Il sistema salva i dati | | |
| 13 | Il sistema avvisa del successo dell’operazione | | |
| **Alternativa 1** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin scrive in modo malformato la data di inizio e/o fine abbonamento e/o la username e/o la password | | |
| 2 | Il sistema rileva l’errore | | |
| 3 | Il sistema invia un messaggio di errore | | |
| **Alternativa 2** | |  | |
| **Step** | Azione | | |
| 1 | L’admin abbandona la creazione dell’abbonato premium in qualsiasi momento | | |
| 2 | Il sistema riporta l’utente alla lista degli abbonati | | |
| **Alternativa 3** | |  | |
| **Step** | Azione | | |
| 1 | L’admin inserisce username e/o password gia utilizzate | | |
| 2 | Il sistema invia un messaggio di errore | | |

### Visualizza elenco abbonati premium

|  |  |
| --- | --- |
| **UC\_KF\_07** | |
| **Descrizione** | | L’utente admin vuole visualizzare l’elenco degli abbonati premium | |
| **Attore Primario** | | Amministratore | |
| **Attori Secondari** | |  | |
| **Precondizioni** | | E’necessario che:   * L’utente admin sia precedentemente autenticato; | |
| **Postcondizione di successo** | | L’admin riesce a visualizzare l’elenco degli abbonati premium | |
| **Postcondizione di fallimento** | | L’admin non riesce a visualizzare l’elenco degli abbonati premium | |
| **Step** | **Azione** | | |
| 1 | L’admin richiede al sistema di visualizzare l’elenco degli abbonati premium | | |
| 2 | Il sistema automaticamente ricerca tutti gli abbonati | | |
| 3 | Per ogni abbonato:   * se ha un abbonamento premium, il sistema lo seleziona | | |
| 4 | Il sistema mostra gli abbonati selezionati | | |
| **Alternativa 1** | | |  |
| **Step** | **Azione** | | |
| 1 | L’admin applica un filtro di ricerca per visualizzare gli abbonati premium con abbonamento valido o scaduto | | |
| 2 | Il sistema recupera gli abbonati in accordo al criterio selezionato | | |
| 3 | If risultato ricerca non è vuoto vengono mostrati gli abbonati premium trovati  Else il sistema avvisa che non esiste alcun risultato | | |

### Cancellazione account

|  |  |
| --- | --- |
| **UC\_KF\_8** | |
| **Descrizione** | | Disattivazione dell’account di un abbonato premium |
| **Attore Primario** | | Amministratore o Tempo |
| **Attori Secondari** | |  |
| **Precondizioni** | | E’necessario che:   * L’abbonamento all’applicazione sia scaduto |
| **Postcondizione di successo** | | L’account dell’abbonato viene disattivato |
| **Postcondizione di fallimento** | | L’account dell’abbonato non viene disattivato |
| **Step** | **Azione** | |
| 1 | L’admin o il tempo richiede al sistema di disattivare l’account | |
| 2 | Il sistema ricerca l’abbonato tramite il suo identificativo | |
| 3 | Il sistema cancella l’abbonamento premium | |
| 4 | Il sistema cancella l’account | |
| 6 | Il sistema comunica che l’operazione è conclusa con successo | |

### Rinnovo abbonamento palestra

|  |  |  |
| --- | --- | --- |
| **UC\_KF\_9** | | |
| **Descrizione** | | | L’admin riattiva l’abbonamento alla palestra |
| **Attore Primario** | | | Amministratore |
| **Attori Secondari** | | |  |
| **Precondizioni** | | | E’necessario che:   * Esiste l’abbonamento |
| **Postcondizione di successo** | | | L’abbonamento alla palestra viene riattivato |
| **Postcondizione di fallimento** | | | L’abbonamento alla palestra non viene riattivato |
| **Step** | **Azione** | | |
| 1 | L’admin seleziona un abbonato | | |
| 2 | L’admin richiede la modifica dell’abbonato | | |
| 2 | L’admin aggiorna la data di inizio e fine abbonamento | | |
| 3 | L’admin invia i dati | | |
| 4 | Il sistema controlla la correttezza delle date:   * Data inizio abbonamento < data fine abbonamento * Data fine abbonamento è una data futura | | |
| 7 | Il sistema ricerca l’abbonamento | | |
| 8 | Il sistema aggiorna le date | | |
| 9 | Il sistema avvisa che l’operazione è conclusa con successo | | |
| **Alternativa 2** | |  | |
| **Step** | Azione | | |
| 1 | Il sistema rileva un errore nelle date inserite | | |
| 2 | Il sistema invia un messaggio di errore | | |
| **Alternativa 3** | |  | |
| **Step** | Azione | | |
| 1 | L’admin abbondona il rinnovo dell’abbonamento | | |
| 2 | Il sistema riporta l’admin all’elenco degli abbonati | | |

### Visualizzazione dettaglio esercizio

|  |  |
| --- | --- |
| **UC\_KF\_10** | |
| **Descrizione** | | L’abbonato vuole visualizzare un esercizio |
| **Attore Primario** | | Abbonato |
| **Attori Secondari** | |  |
| **Precondizioni** | | L’applicazione sia attiva e disponibile |
| **Postcondizione di successo** | | L’abbonato visualizza il dettaglio dell’esercizio |
| **Postcondizione di fallimento** | | L’abbonato non visualizza il dettaglio dell’esercizio |
| **Step** | **Azione** | |
| 1 | L’abbonato richiede di visualizzare un esercizio | |
| 2 | Il sistema ricerca l’esercizio tramite il suo identificativo | |
| 3 | Il sistema mostra il dettaglio dell’esercizio | |

### Visualizzazione elenco esercizi

|  |  |
| --- | --- |
| **UC\_KF\_11** | |
| **Descrizione** | | L’abbonato vuole visualizzare la lista di esercizi |
| **Attore Primario** | | Abbonato |
| **Attori Secondari** | |  |
| **Precondizioni** | | L’applicazione sia attiva e disponibile |
| **Postcondizione di successo** | | L’abbonato visualizza la lista di esercizi |
| **Postcondizione di fallimento** | | L’abbonato non visualizza la lista di esercizi |
| **Step** | **Azione** | |
| 1 | L’abbonato richiede la visualizzazione di tutti gli esercizi | |
| 2 | Il sistema ricerca gli esercizi | |
| 2 | Il sistema mostra l’elenco degli esercizi | |

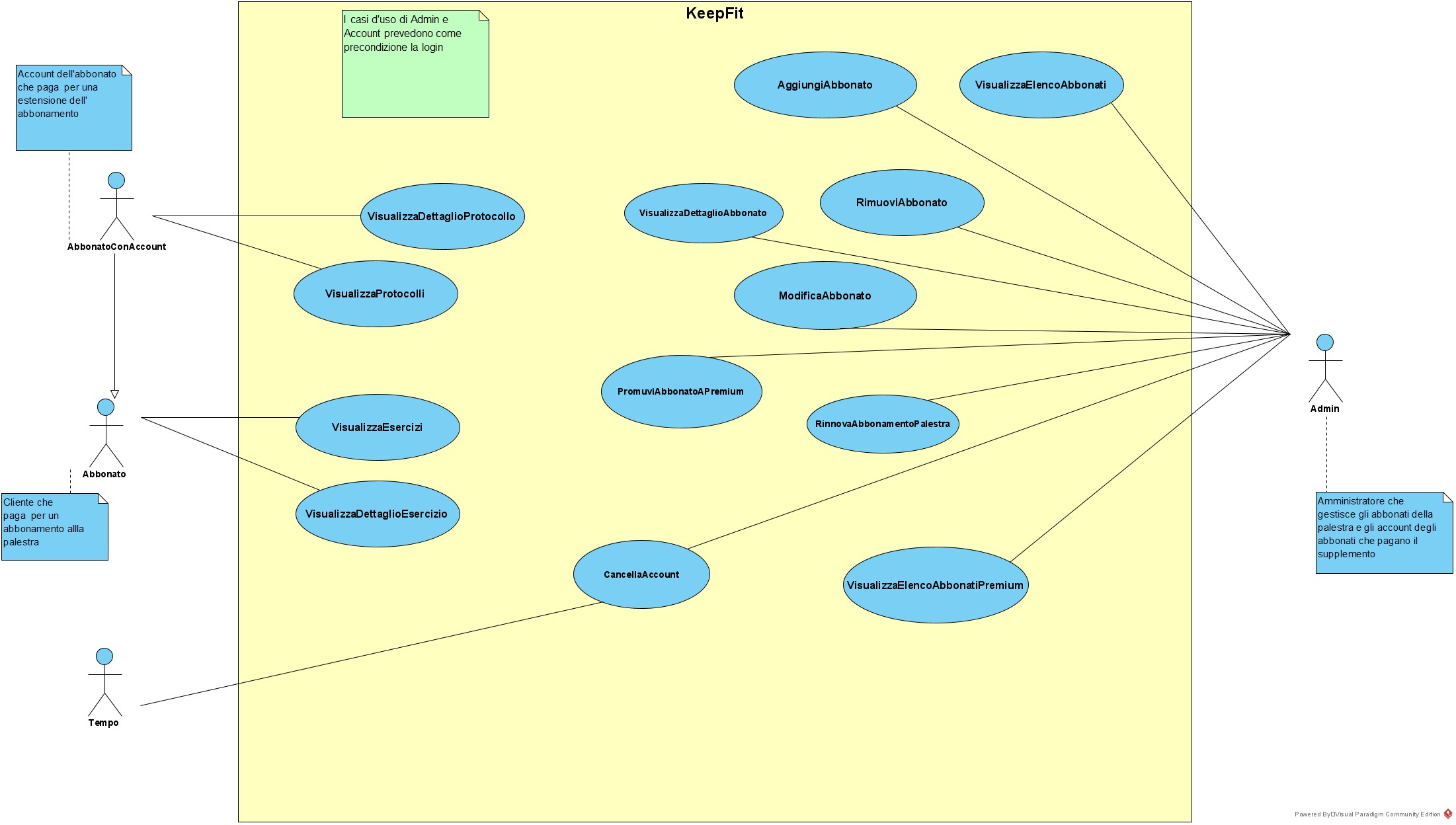
### Visualizzazione elenco protocolli

|  |  |
| --- | --- |
| **UC\_KF\_12** | |
| **Descrizione** | | L’abbonato premium vuole visualizzare i protocolli di allenamento |
| **Attore Primario** | | AbbonatoConAccount |
| **Attori Secondari** | |  |
| **Precondizioni** | | E’necessario che:   * L’abbonato premium sia precedentemente autenticato |
| **Postcondizione di successo** | | L’abbonato premium visualizza la lista dei protocolli |
| **Postcondizione di fallimento** | | L’abbonato premium non visualizza la lista dei protocolli |
| **Step** | **Azione** | |
| 1 | L’abbonato premium richiede di visualizzare i protocolli di allenamento | |
| 2 | Il sistema ricerca tutti i protocolli | |
| 3 | Il sistema mostra l’elenco dei protocolli di allenamento | |

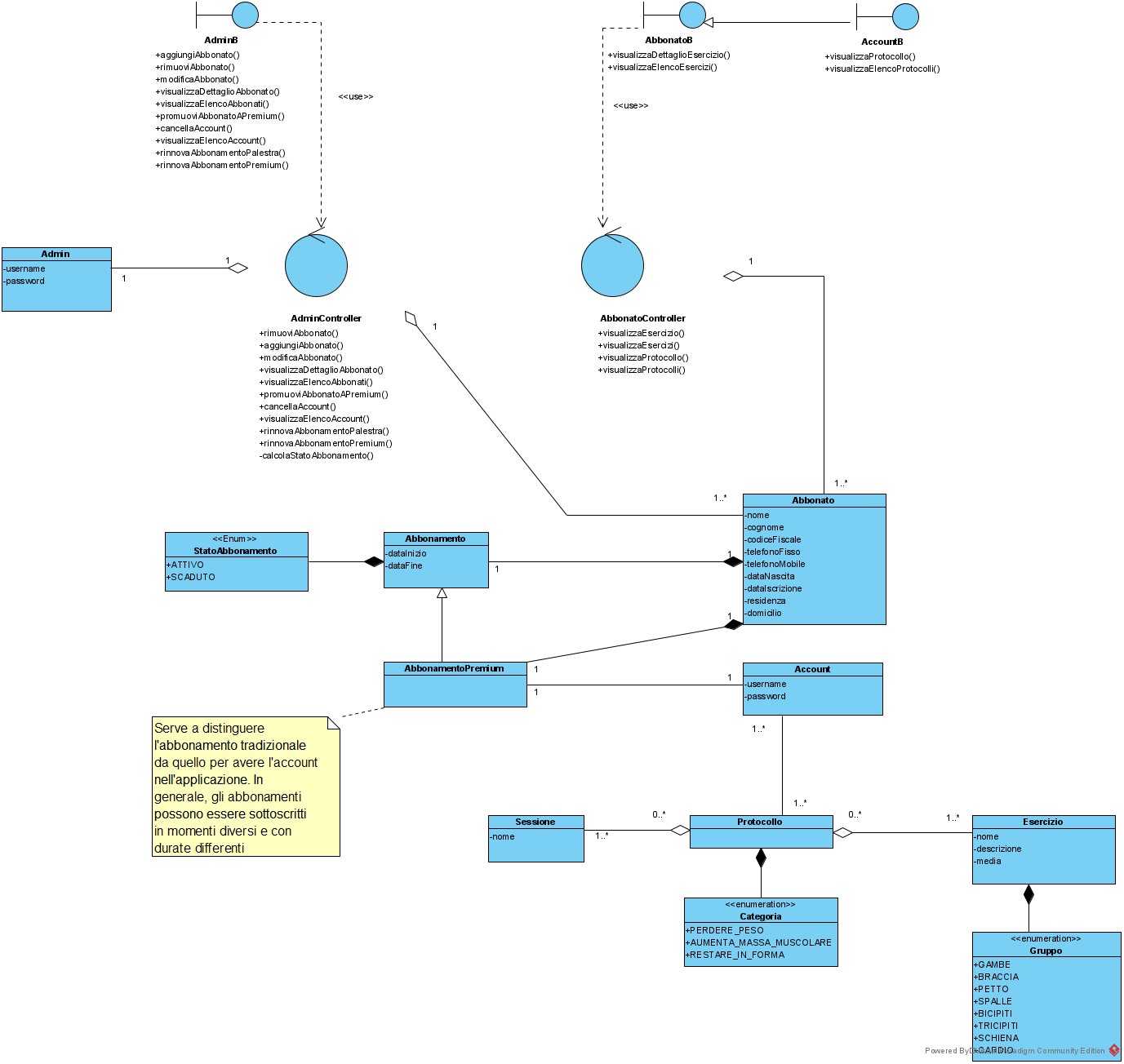
### Visualizzazione dettaglio protocollo

|  |  |
| --- | --- |
| **UC\_KF\_13** | |
| **Descrizione** | | L’abbonato premium vuole visualizzare un protocollo di allenamento |
| **Attore Primario** | | AbbonatoConAccount |
| **Attori Secondari** | |  |
| **Precondizioni** | | E’necessario che:   * L’abbonato premium sia precedentemente autenticato; * L’abbonato premium abbia visualizzato l’elenco dei protocolli |
| **Postcondizione di successo** | | L’abbonato premium visualizza un protocollo |
| **Postcondizione di fallimento** | | L’abbonato premium non visualizza un protocollo |
| **Step** | **Azione** | |
| 1 | L’abbonato premium seleziona un protocollo dalla lista di protocolli | |
| 2 | Il sistema ricerca il protocollo tramite il suo identificativo | |

### Diagramma dei casi d’uso

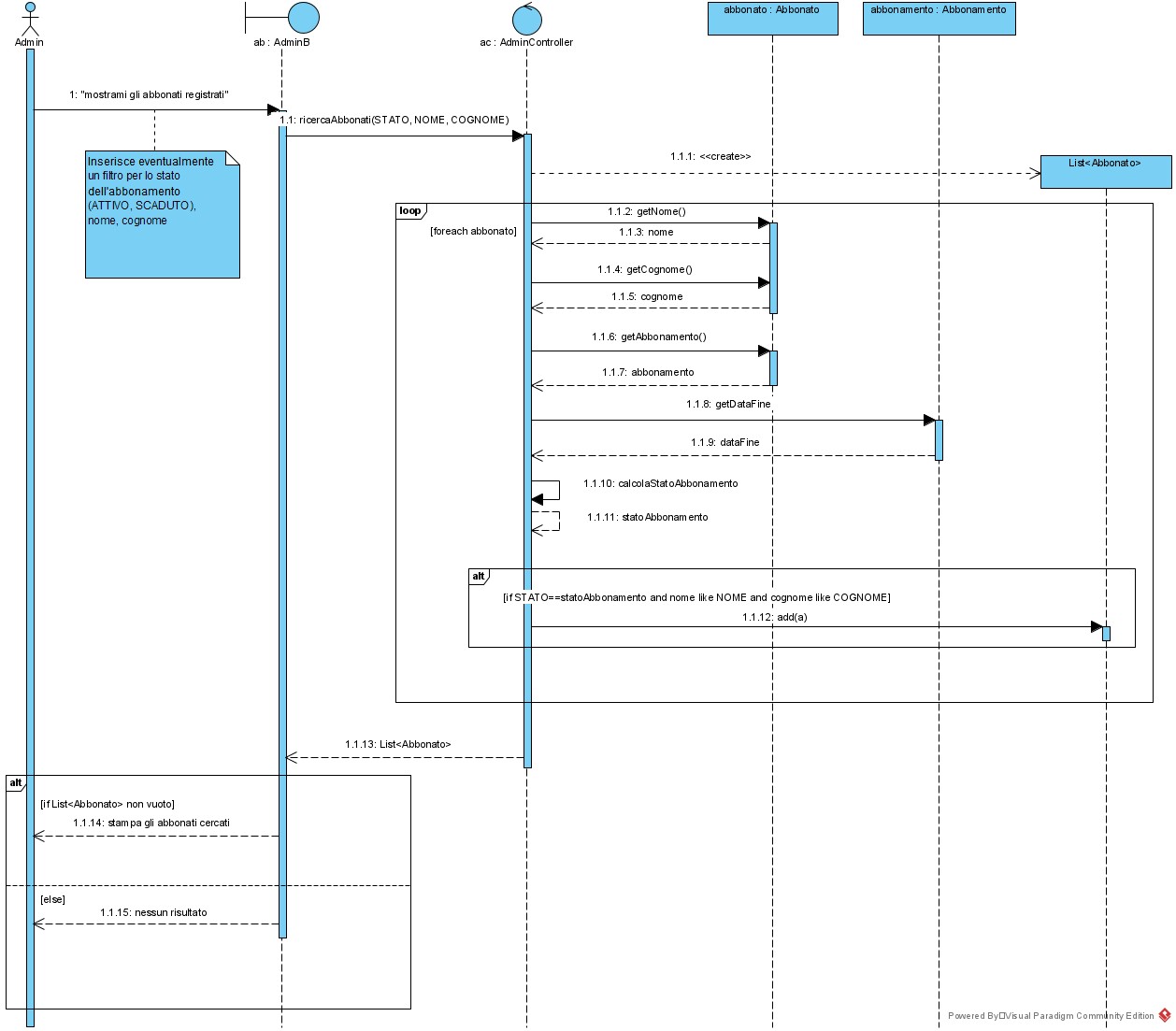


## Diagramma delle classi

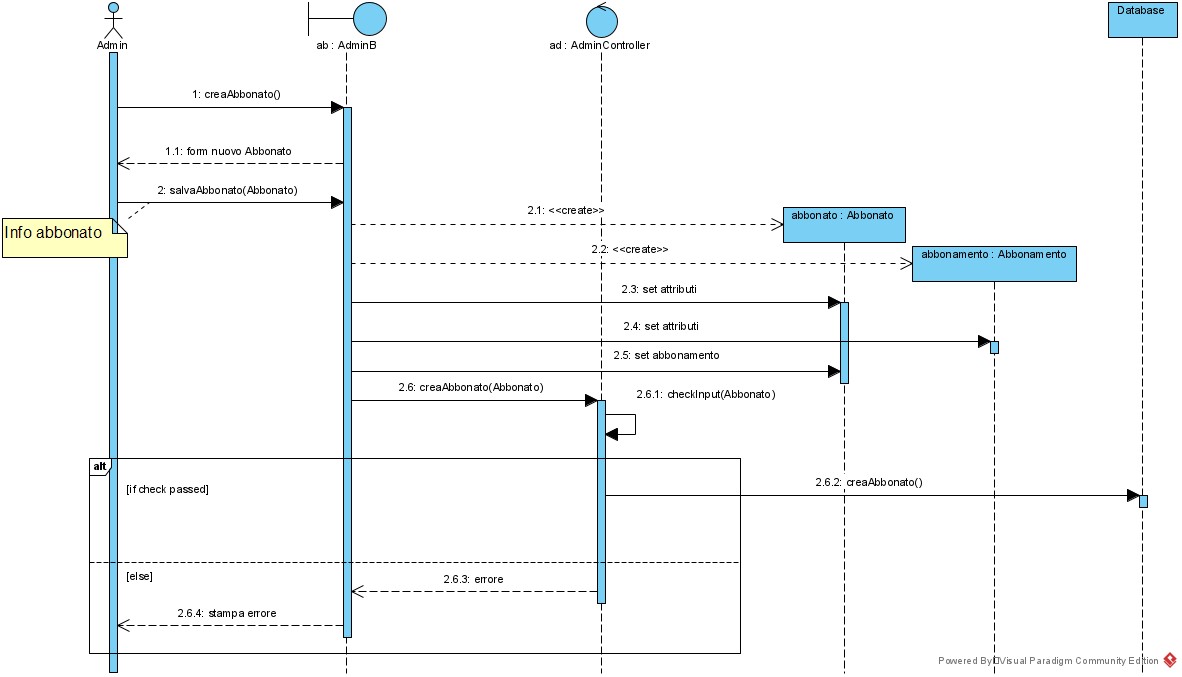


## Diagrammi di sequenza

### Visualizza elenco abbonati



### Crea abbonato



# Stima dei costi

## Valutazione degli Unadjusted Function Point (UFP)

### Ambito di conteggio

Sviluppo di progetto.

### Identificazione funzioni dati e transazionali

Non sono presenti External Interface Files (EIF) poichè il sistema non coopera con altri sistemi esterni, quindi NEIF=0. È presente il database che è un Internal Logical File (ILF), quindi NILF = 1.

La complessità di ILF è 10 perché RET=8 (8 tabelle) e DET=19 (input diversi).

-EQ-

Le funzionalità che eseguono solo query sono 7. Per ognuno di esse FTR=1.

Per i DET consideriamo la funzionalità più complessa:

*Visualizza elenco abbonati*: ha tre parametri di input(nome, cognome, stato abbonamento). Le combinazioni degli input possibili per la ricerca sono 7; aggiungiamo 1 input per applicare il filtro e 1 input per eventuale messaggio di risposta. Quindi, DET=9. Il peso assegnato è 3.

-EI-

Le funzionalità che modificano L’ILF sono 6. Per ognuno di esse FTR=1.

Per i DET consideriamo la funzionalità più complessa:

*Crea abbonato*: bisogna fornire nome, cognome, data di nascita, codice fiscale, residenza, domicilio, data di iscrizione, data inizio e fine abbonamento. Aggiungiamo 1 input per il bottone di invio dati e 1 input per eventuale messaggio di risposta. Quindi, DET=11. Il peso assegnato è 3.

|  |  |  |  |
| --- | --- | --- | --- |
| INDICE | VALORE | PESO | VPI |
| NILF | 1 | 10 | 10 |
| NEIF | 0 | 5 | 0 |
| NEI | 6 | 3 | 18 |
| NEO | 0 | 4 | 0 |
| NEQ | 7 | 3 | 21 |
|  |  | totale | 49 |

## Valutazione dei FP

UFP = 49

|  |  |
| --- | --- |
| CARATTERISTICHE GENERALI | VALORE |
| COMUNICAZIONE DATI | 1 |
| DISTRIBUZIONE ELABORAZIONE | 2 |
| PRESTAZIONI | 3 |
| UTILIZZO INTENSIVO CONFIGURAZIONE | 1 |
| FREQUENZA DELLE TRANSAZIONI | 3 |
| AGGIORNAMENTO INTERATTIVO | 3 |
| EFFICIENZA PER L’UTENTE FINALE | 4 |
| INSERIMENTO DATI INTERATTIVO | 3 |
| COMPLESSITA’ ELABORATIVA | 4 |
| RIUSABILITA’ | 3 |
| FACILITA’ INSTALLAZIONE | 4 |
| FACILITA’ GESTIONE OPERATIVA | 3 |
| MOLTEPLICITA’ DI SITI | 1 |
| FACILITA’ DI MODIFICA | 3 |

AFP = 0.65 + 0.01 \*(2\*1+5\*2+7\*3) = 0,99

FP = UFP x AFP = 49 \* 0.99 = 48,51

### Valutazione LLOC

JAVA LLOC = 48,51\*53 = 2571,03

# Piano di test funzionale

**PIANO DI TEST UTILIZZANDO IL METODO DEL *CATEGORY-PARTITION TESTING* PER LA FUNZIONALITÀ DI “*Visualizza elenco abbonati*”.**

|  |  |  |  |
| --- | --- | --- | --- |
| Categoria Tipo di ricerca | Categoria Numero abbonati su DB | Categoria Stato abbonamento | Categoria Esito ricerca |
| * Per nome [SINGLE] * Per cognome * Per stato abbonamento * Per nome e cognome [SINGLE] * Per nome e stato abbonamento [SINGLE] * Per cognome e stato abbonamento [SINGLE] * Per nome, cognome e stato abbonamento [SINGLE] * Nessun nome, cognome, stato[SINGLE] | * 0 [SINGLE] * 1 [SINGLE] * > 1 | * Solo attivi [SINGLE] * Solo scaduti [SINGLE] * Entrambi | * 0 * 1 * > 1 |

**TEST SUITE**

La funzionalità ha 4 categorie. La prima categoria ha 8 classi di valori, la seconda 3 classi di valori, la terza 3 classi di valori, la quarta 3 classi di valori. Il numero totale di test senza vincoli è:

8x3x3x3=216

Vincoli error: 0

Vincoli single: 10

Vincoli property: 0

Il numero di test con i vincoli single ed error sono:

2x1x1x3=6

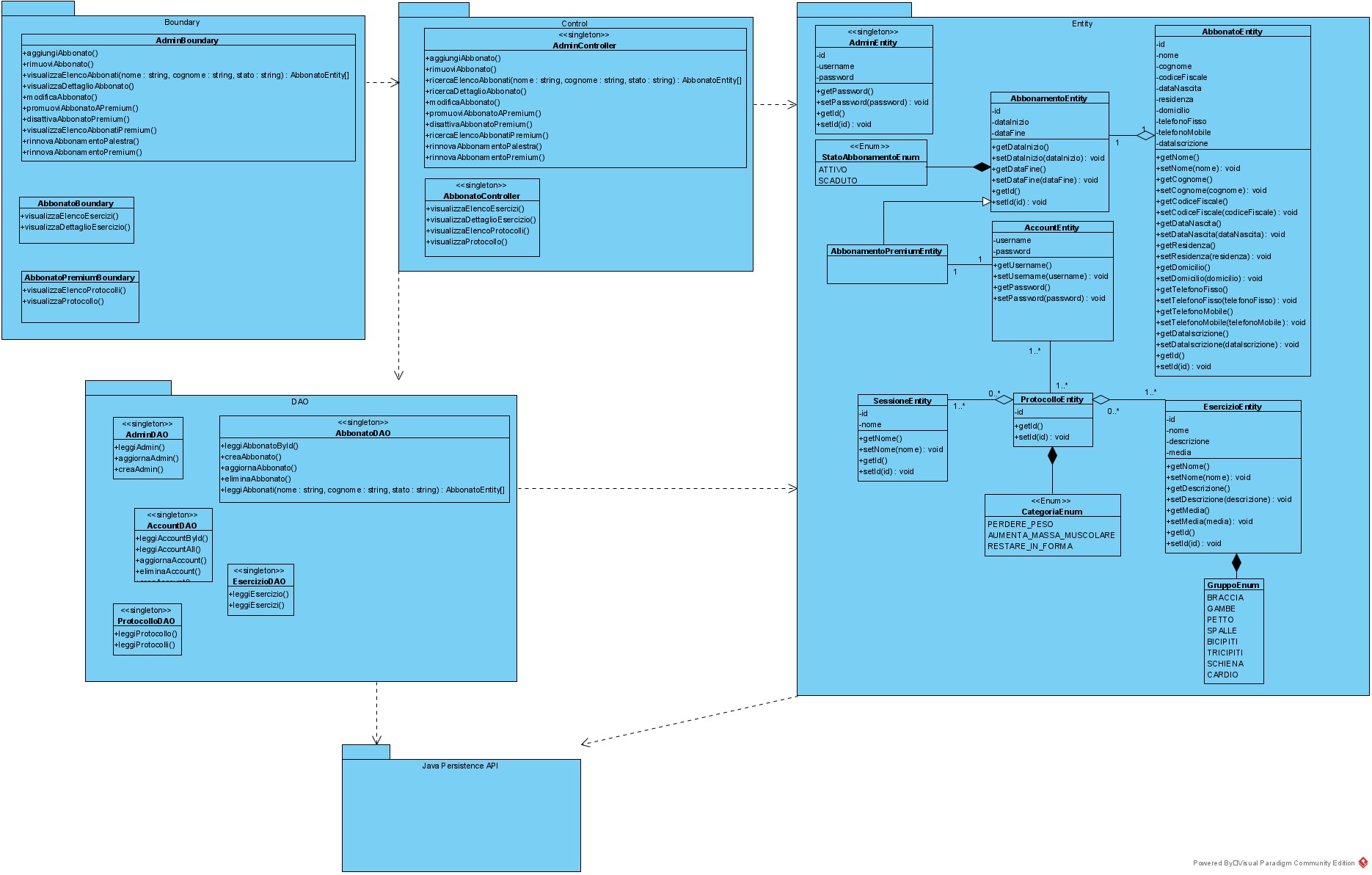
Quindi il numero di test da eseguire è:

6 + 10(single) =16

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** |
| 1 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, ” Esposito”, “”) | 0 risultati | N/A |
| 2 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, ATTIVO]  } | Ricerca(“”, “”, “SCADUTO” ) | 0 risultati | N/A |
| 3 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, “ Rossi”, “”) | 1 risultati | N/A |
| 4 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “”, “ SCADUTO” ) | 1 risultati | N/A |
| 5 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:>1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Rossi , ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “ Rossi”, “” ) | 2 risultati | N/A |
| 6 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:4  Stato abbonamento: ENTRAMBI  Esito ricerca: >1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “”, “ATTIVO” ) | 2 risultati | N/A |
| 7 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ATTIVO  Esito ricerca:>1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, ATTIVO]  } | Ricerca(“”, “”, “ ATTIVO ” ) | 3 risultati | N/A |
| 8 |  | Tipo ricerca: QUALSIASI  Numero abbonati:3  Stato abbonamento: SCADUTO  Esito ricerca:>1 | DB = {  [Mario, Rossi, SCADUTO ],  [Luca, Verde, SCADUTO ],  [Andrea, Giallo, SCADUTO ]  } | Ricerca(“”, “”, “ SCADUTO” ) | 3 risultati | N/A |
| 9 |  | Tipo ricerca: COGNOME  Numero abbonati:1  Stato abbonamento: ATTIVO  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO]  } | Ricerca( “”, “ Rossi”, “”) | 1 risultati | N/A |
| 10 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:0  Stato abbonamento: N/A  Esito ricerca:0 | DB = {  } | Ricerca( “”, “”, “ ATTIVO”) | 0 risultati | N/A |
| 11 |  | Tipo ricerca: TUTTI I CAMPI VUOTI  Numero abbonati:2  Stato abbonamento: ENTRAMBI  Esito ricerca:2 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, SCADUTO],  } | Ricerca( “”, “”, “ ”) | 2 risultati | N/A |
| 12 |  | Tipo ricerca: NOME  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “ Luca”, “”, “”) | 1 risultati | N/A |
| 13 |  | Tipo ricerca: NOME E COGNOME  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “Mario”, “Rossi”, “”) | 1 risultati | N/A |
| 14 |  | Tipo ricerca: NOME E STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “Andrea”, “”, “ATTIVO”) | 0 risultati | N/A |
| 15 |  | Tipo ricerca: COGNOME E STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, “Verde”, “SCADUTO”) | 0 risultati | N/A |
| 16 |  | Tipo ricerca:NOME, COGNOME, STATO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“Mario”, “Esposito”, “ATTIVO” ) | 0 risultati | N/A |

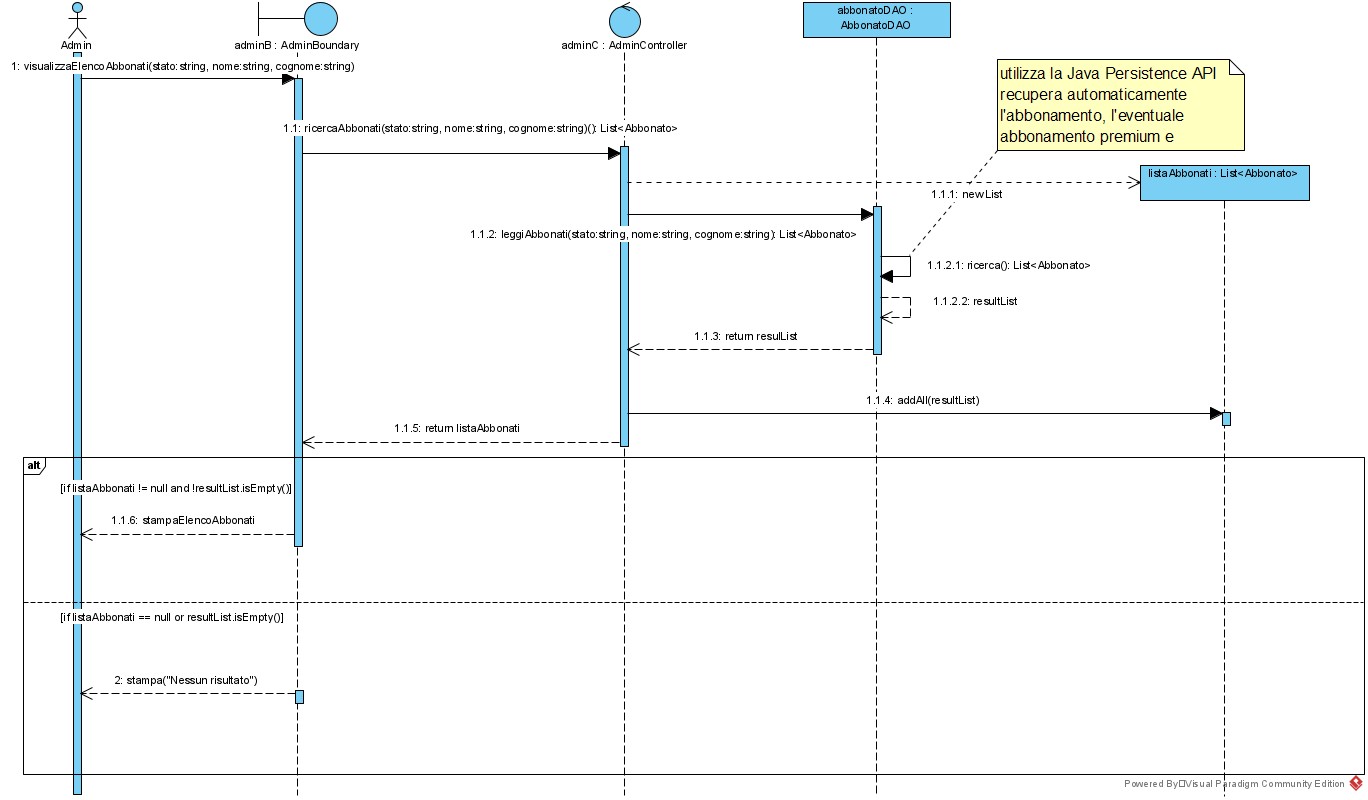
# Progettazione

## Diagramma delle classi

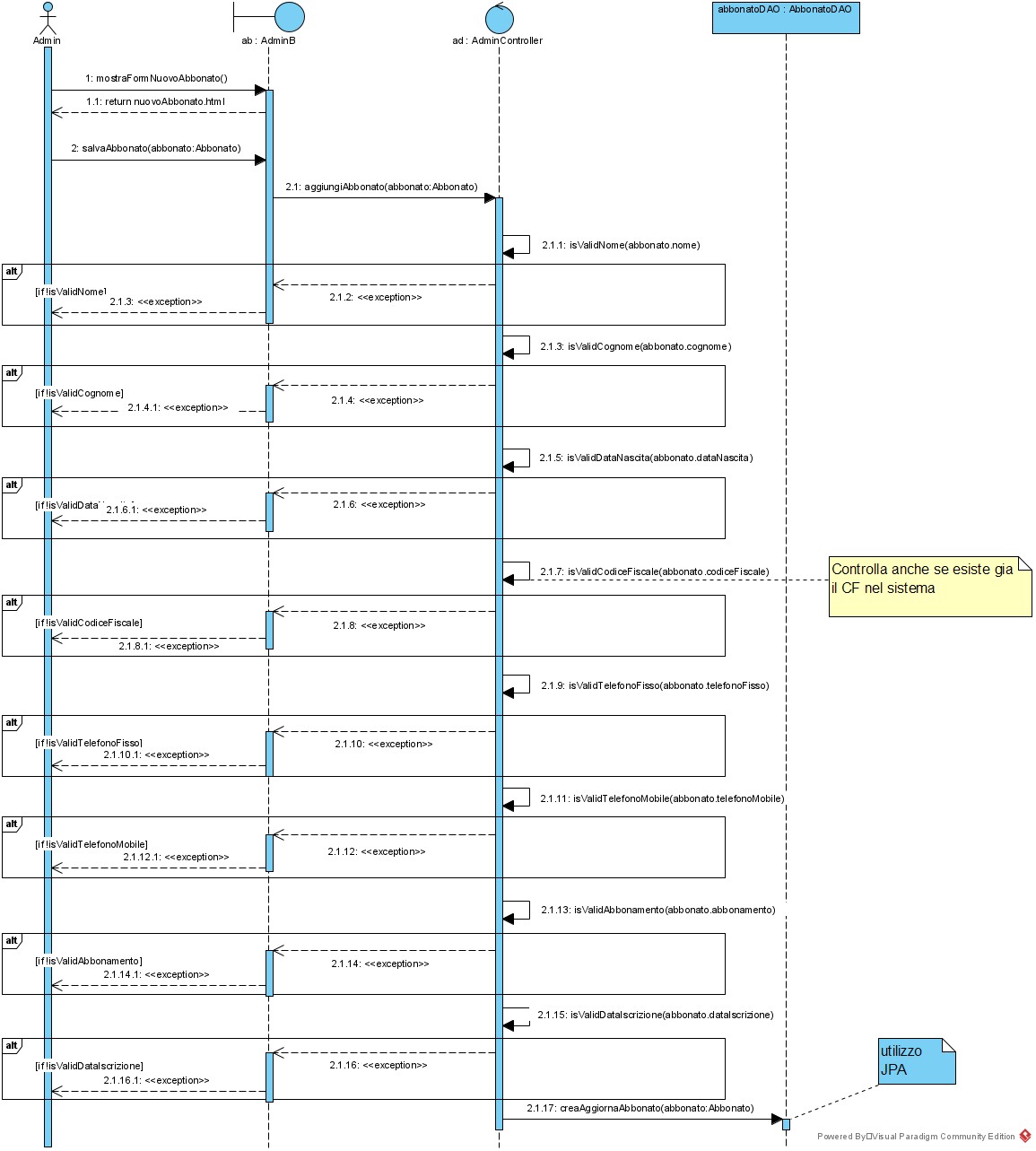


## Diagrammi di sequenza

### Visualizza elenco abbonati



### Crea abbonato



# Implementazione

## Elenco package

* [it.keepfit](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\package-frame.html)
* [it.keepfit.boundary](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\package-frame.html)
* [it.keepfit.boundary.impl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\impl\package-frame.html)
* [it.keepfit.constants](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\constants\package-frame.html)
* [it.keepfit.control](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\package-frame.html)
* [it.keepfit.control.impl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\impl\package-frame.html)
* [it.keepfit.dao](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\package-frame.html)
* [it.keepfit.dao.impl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\impl\package-frame.html)
* [it.keepfit.entity](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\package-frame.html)
* [it.keepfit.exception](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\exception\package-frame.html)

## Classi

* [Abbonamento](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonamento.html)
* [AbbonamentoPremium](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\AbbonamentoPremium.html)
* [Abbonato](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonato.html)
* [*AbbonatoBoundary*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\AbbonatoBoundary.html)
* [AbbonatoBoundaryImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\impl\AbbonatoBoundaryImpl.html)
* [*AbbonatoController*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\AbbonatoController.html)
* [AbbonatoControllerImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\impl\AbbonatoControllerImpl.html)
* [*AbbonatoDAO*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\AbbonatoDAO.html)
* [AbbonatoDAOImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\impl\AbbonatoDAOImpl.html)
* [Account](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Account.html)
* [*AccountBoundary*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\AccountBoundary.html)
* [AccountBoundaryImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\impl\AccountBoundaryImpl.html)
* [*AccountDAO*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\AccountDAO.html)
* [AccountDAOImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\impl\AccountDAOImpl.html)
* [Admin](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Admin.html)
* [*AdminBoundary*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\AdminBoundary.html)
* [AdminBoundaryImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\impl\AdminBoundaryImpl.html)
* [*AdminController*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\AdminController.html)
* [AdminControllerImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\impl\AdminControllerImpl.html)
* [*AdminDAO*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\AdminDAO.html)
* [Categoria](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Categoria.html)
* [Esercizio](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Esercizio.html)
* [*EsercizioDAO*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\EsercizioDAO.html)
* [EsercizioDAOImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\impl\EsercizioDAOImpl.html)
* [GruppoMuscolare](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\GruppoMuscolare.html)
* [KeepfitWebappApplication](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\KeepfitWebappApplication.html)
* [Protocollo](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Protocollo.html)
* [*ProtocolloDAO*](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\ProtocolloDAO.html)
* [ProtocolloDAOImpl](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\impl\ProtocolloDAOImpl.html)
* [Sessione](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Sessione.html)
* [StatoAbbonamento](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\constants\StatoAbbonamento.html)

## Eccezioni

* [AbbonatoException](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\exception\AbbonatoException.html)

## Elenco artefatti

* spring-boot-starter-data-jpa v2.4.0
* spring-boot-starter-thymeleaf v2.4.0
* spring-boot-starter-web v2.4.0
* h2 v1.4.200
* spring-boot-starter-test v2.4.0
* jdk 1.8
* Bootstrap 4

## Javadoc funzionalità implementate

Di seguito la documentazione Javadoc relativa alla funzionalità “visualizza elenco abbonati”.

### AdminBoundary

#### visualizzaElencoAbbonati

@GetMapping(value="/admin/abbonati")

java.lang.String visualizzaElencoAbbonati(@RequestParam(value="nome")

java.lang.String nome,

@RequestParam(value="cognome")

java.lang.String cognome,

@RequestParam(value="stato")

java.lang.String stato,

org.springframework.ui.Model model)

Visualizza elenco abbonati.

**Parameters:**

nome - nome

cognome - cognome

stato - stato abbonamento

model - model

**Returns:**

il nome della pagina html da renderizzare

### AdminBoundaryImpl

**visualizzaElencoAbbonati**

public java.lang.String visualizzaElencoAbbonati(java.lang.String nome,

java.lang.String cognome,

java.lang.String stato,

org.springframework.ui.Model model)

Visualizza elenco abbonati.

Specified by:

[visualizzaElencoAbbonati](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\boundary\AdminBoundary.html#visualizzaElencoAbbonati-java.lang.String-java.lang.String-java.lang.String-org.springframework.ui.Model-) in interface [AdminBoundary](file:///D:\\Users\\libre\\INGEGNERIA%20DEL%20SOFTWARE\\canale_j_z-libre_team\\software\\source\\keepfit\\doc\\it\\keepfit\\boundary\\AdminBoundary.html" \o "interface in it.keepfit.boundary)

Parameters:

nome - nome

cognome - cognome

stato - stato abbonamento

model - il model

Returns:

il nome della pagina html da renderizzare

### AdminController

#### ricercaAbbonati

java.util.List<[Abbonato](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonato.html)> ricercaAbbonati(java.lang.String nome,

java.lang.String cognome,

java.lang.String stato)

Ricerca abbonati.

**Parameters:**

nome - nome

cognome - cognome

stato - stato abbonamento

**Returns:**

the list

### AdminControllerImpl

**ricercaAbbonati**

public java.util.List<[Abbonato](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonato.html)> ricercaAbbonati(java.lang.String nome,

java.lang.String cognome,

java.lang.String stato)

Ricerca abbonati.

Specified by:

[ricercaAbbonati](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\control\AdminController.html#ricercaAbbonati-java.lang.String-java.lang.String-java.lang.String-) in interface [AdminController](file:///D:\\Users\\libre\\INGEGNERIA%20DEL%20SOFTWARE\\canale_j_z-libre_team\\software\\source\\keepfit\\doc\\it\\keepfit\\control\\AdminController.html" \o "interface in it.keepfit.control)

Parameters:

nome - nome

cognome - cognome

stato - stato abbonamento

Returns:

the list

### AbbonatoDAO

#### leggiAbbonati

java.util.List<[Abbonato](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonato.html)> leggiAbbonati(java.lang.String nome,

java.lang.String cognome,

java.lang.String stato)

Leggi abbonati.

**Parameters:**

nome - nome

cognome - cognome

stato - stato abbonamento

**Returns:**

the list

### AbbonatoDAOImpl

**leggiAbbonati**

public java.util.List<[Abbonato](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\entity\Abbonato.html)> leggiAbbonati(java.lang.String nome,

java.lang.String cognome,

java.lang.String stato)

Leggi abbonati.

Specified by:

[leggiAbbonati](file:///D:\Users\libre\INGEGNERIA%20DEL%20SOFTWARE\canale_j_z-libre_team\software\source\keepfit\doc\it\keepfit\dao\AbbonatoDAO.html#leggiAbbonati-java.lang.String-java.lang.String-java.lang.String-) in interface [AbbonatoDAO](file:///D:\\Users\\libre\\INGEGNERIA%20DEL%20SOFTWARE\\canale_j_z-libre_team\\software\\source\\keepfit\\doc\\it\\keepfit\\dao\\AbbonatoDAO.html" \o "interface in it.keepfit.dao)

Parameters:

nome - the nome

cognome - the cognome

stato - the stato

Returns:

the list

### Considerazioni

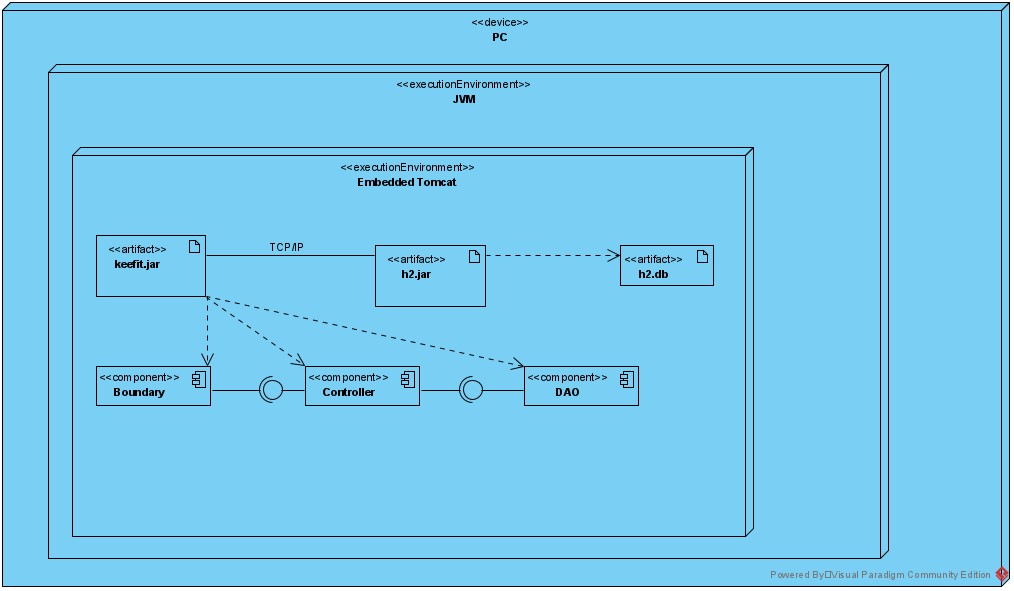
Dato che le funzionalità da implementare sono in totale 13, si è stimato che per ogni funzionalità bisogna scrivere circa 198 righe di codice.

Per la funzionalità “visualizza elenco abbonati” sono stati scritte 213 righe di codice, quasi in linea rispetto alle previsioni.

### Javadoc

link Javadoc completo: <https://github.com/IS-unina/canale_j_z-libre_team/tree/master/software/source/keepfit/doc>

## Diagramma di deployment



# Testing

## Test funzionale

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Descrizione** | **Classi di equivalenza coperte** | **Pre-condizioni** | **Input** | **Output Attesi** | **Post-condizioni Attese** | **Output Ottenuti** | **Post-condizioni Ottenute** | **Esito**  **(*FAIL*, *PASS*)** |
| 1 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, ” Esposito”, “TUTTI”) | 0 risultati | N/A | Nessun risultato | N/A | OK |
| 2 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, ATTIVO]  } | Ricerca(“”, “”, “SCADUTO” ) | 0 risultati | N/A | Nessun risultato | N/A | OK |
| 3 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, “ Rossi”, “ TUTTI”) | 1 risultati | N/A | {  [Mario, Rossi, ATTIVO]  } | N/A | OK |
| 4 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “”, “ SCADUTO” ) | 1 risultati | N/A | {  [Andrea, Giallo, SCADUTO]  } | N/A | OK |
| 5 |  | Tipo ricerca: COGNOME  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:>1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Rossi , ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “ Rossi”, “ TUTTI” ) | 2 risultati | N/A | {  [Mario, Rossi, ATTIVO],  [Luca, Rossi , ATTIVO]  } | N/A | OK |
| 6 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:4  Stato abbonamento: ENTRAMBI  Esito ricerca: >1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“”, “”, “ATTIVO” ) | 2 risultati | N/A | {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO]  } | N/A | OK |
| 7 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ATTIVO  Esito ricerca:>1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, ATTIVO]  } | Ricerca(“”, “”, “ ATTIVO ” ) | 3 risultati | N/A | {  [Mario, Rossi, SCADUTO ],  [Luca, Verde, SCADUTO ],  [Andrea, Giallo, SCADUTO ]  } | N/A | OK |
| 8 |  | Tipo ricerca: QUALSIASI  Numero abbonati:3  Stato abbonamento: SCADUTO  Esito ricerca:>1 | DB = {  [Mario, Rossi, SCADUTO ],  [Luca, Verde, SCADUTO ],  [Andrea, Giallo, SCADUTO ]  } | Ricerca(“”, “”, “ SCADUTO” ) | 3 risultati | N/A | {  [Mario, Rossi, SCADUTO ],  [Luca, Verde, SCADUTO ],  [Andrea, Giallo, SCADUTO ]  } | N/A | OK |
| 9 |  | Tipo ricerca: COGNOME  Numero abbonati:1  Stato abbonamento: ATTIVO  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO]  } | Ricerca( “”, “ Rossi”, “ TUTTI”) | 1 risultati | N/A | {  [Mario, Rossi, ATTIVO]  } | N/A | OK |
| 10 |  | Tipo ricerca: STATO ABBONAMENTO  Numero abbonati:0  Stato abbonamento: N/A  Esito ricerca:0 | DB = {  } | Ricerca( “”, “”, “ ATTIVO”) | 0 risultati | N/A | Nessun risultato | N/A | OK |
| 11 |  | Tipo ricerca: TUTTI I CAMPI VUOTI  Numero abbonati:2  Stato abbonamento: ENTRAMBI  Esito ricerca:2 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, SCADUTO],  } | Ricerca( “”, “”, “ TUTTI”) | 2 risultati | N/A | Nessun risultato | N/A | KO |
| 12 |  | Tipo ricerca: NOME  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “ Luca”, “”, “ TUTTI”) | 1 risultati | N/A | {  [Luca, Verde, ATTIVO],  } | N/A | OK |
| 13 |  | Tipo ricerca: NOME E COGNOME  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:1 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “Mario”, “Rossi”, “ TUTTI”) | 1 risultati | N/A | {  [Mario, Rossi, ATTIVO],  } | N/A | OK |
| 14 |  | Tipo ricerca: NOME E STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento:ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “Andrea”, “”, “ATTIVO”) | 0 risultati | N/A | Nessun risultato | N/A | OK |
| 15 |  | Tipo ricerca: COGNOME E STATO ABBONAMENTO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca( “”, “Verde”, “SCADUTO”) | 0 risultati | N/A | Nessun risultato | N/A | OK |
| 16 |  | Tipo ricerca:NOME, COGNOME, STATO  Numero abbonati:3  Stato abbonamento: ENTRAMBI  Esito ricerca:0 | DB = {  [Mario, Rossi, ATTIVO],  [Luca, Verde, ATTIVO],  [Andrea, Giallo, SCADUTO]  } | Ricerca(“Mario”, “Esposito”, “ATTIVO” ) | 0 risultati | N/A | Nessun risultato | N/A | OK |

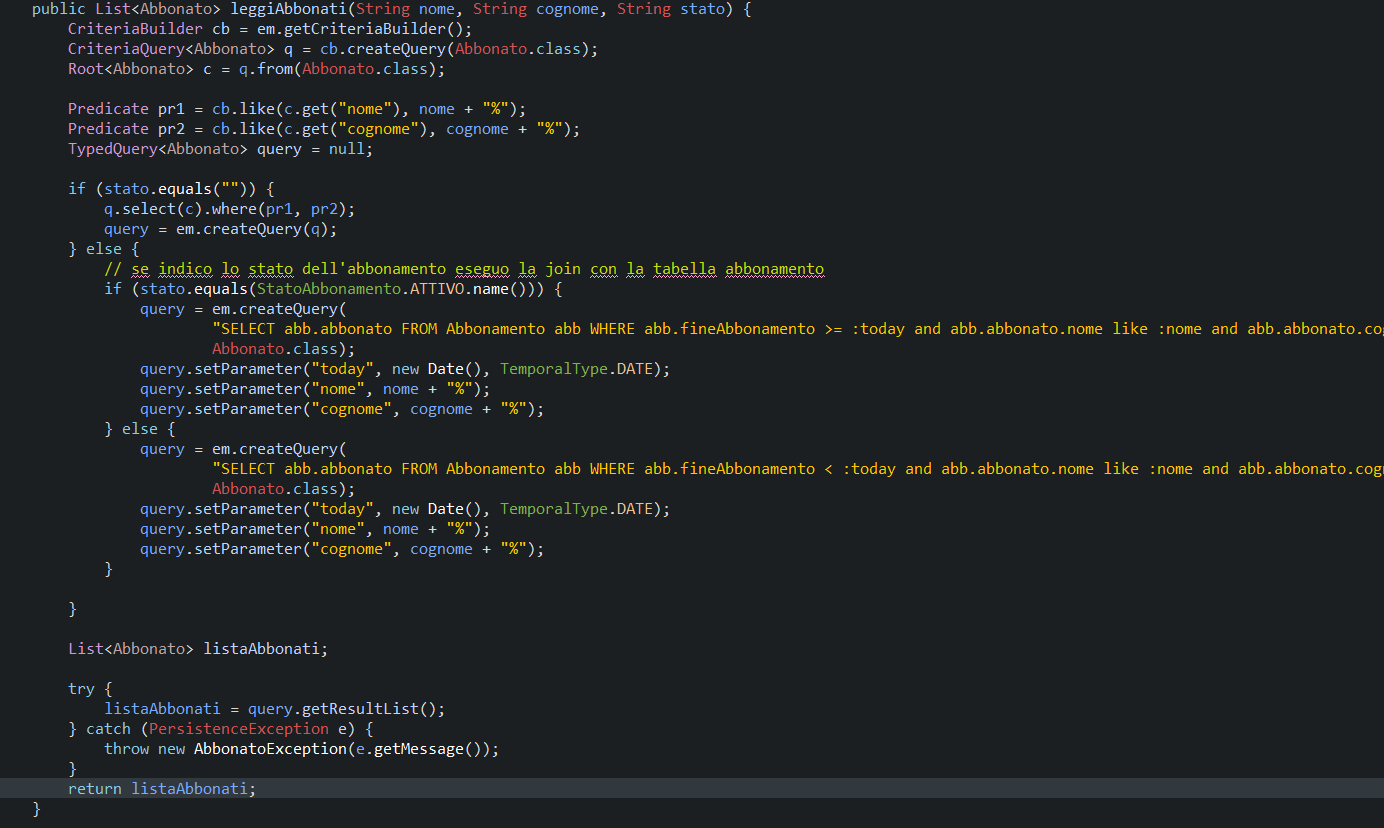
Il testing funzionale verifica che le funzionalità implementate abbiano il comportamento atteso. Il testing funzionale è eseguito a partire dalle interfacce dell’applicazione, in modalità black-box, in base ai test definiti in fase di analisi. Quando un test fallisce, allora esiste un errore. Se non è immediato individuare il punto esatto del codice che ha causato l’errore, è utile rieseguire il test in modalità debug, con la possibilità di eseguire in modo controllato ogni istruzione presente nel codice.

Per esempio, il test 11 inizialmente ha prodotto erroneamente come risultato il messaggio “Nessun risultato”, mentre il risultato atteso è la visualizzazione di tutti gli abbonati registrati. Per individuare l’errore, si è proceduto a riavviare l’applicazione con il debug attivo; è stato inserito un punto di debug nella classe boundary *AdminBoundaryImpl* sulla prima istruzione che viene eseguita nella procedura *visualizzaElencoAbbbonati*. Il debug viene attivato prima dell’esecuzione dell’istruzione marcata. Esaminando le istruzioni successive è emerso che nella procedura non veniva mai valorizzata la variabile *elencoEmpty* con il valore false quando la lista degli abbonati non è vuota. Quindi, è stata aggiunta l’istruzione mancante e rieseguito il test con esito positivo.

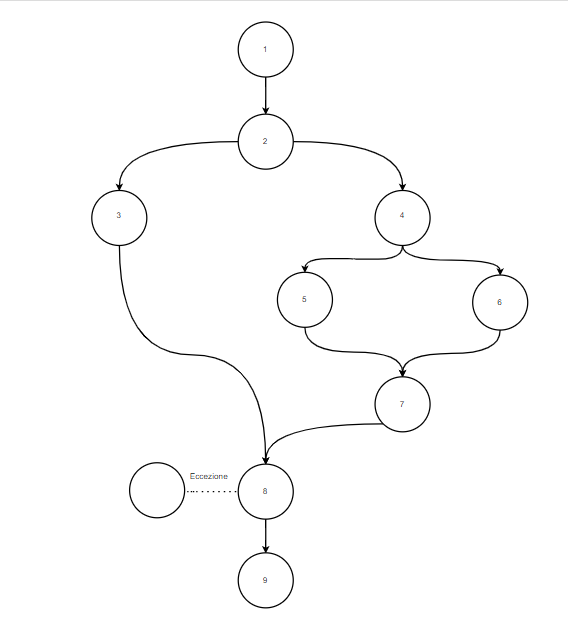
## Test strutturale

## Complessità ciclomatica

La funzione considerata è *leggiAbbonati(String nome, String cognome, String stato)* della classe **AbbonatoDAOImpl**.



Di seguito il CFG della funzione da testare con il testing strutturale.



La complessità ciclomatica della funzione può essere calcolata come il numero regioni del grafo, cioè è pari a 3. I cammini indipendenti sono:

1. 1-2-3-8-9
2. 1-2-4-5-7-8-9
3. 1-2-4-6-7-8-9

## Test di unità

I test da eseguire per coprire tutti i cammini indipendenti sono indicati dai seguenti test case:

**Test case cammino 1**:

valore(stato abbonamento) = “”

**Test case cammino 2**:

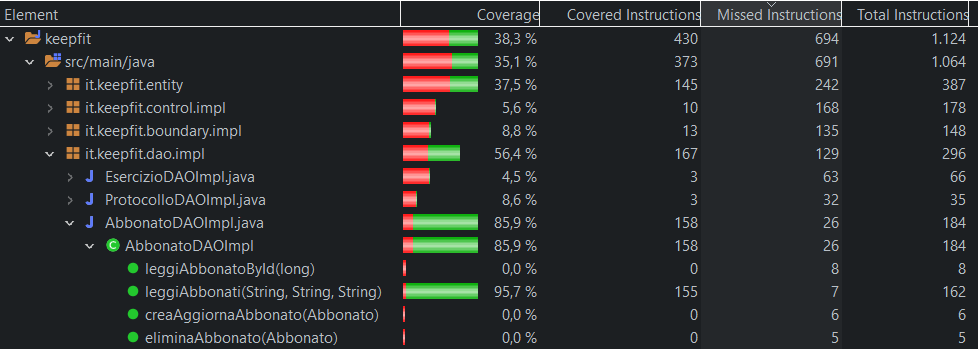
valore(stato abbonamento) = “attivo”

**Test case cammino 3**:

valore(stato abbonamento) = “scaduto”

## Test di unità con JUnit





Per la funzione *leggiAbbonati(String nome, String cognome, String stato)* è stata raggiunta una copertura del 95,7%.