



UNIVERSITÀ DI PISA

SCUOLA DI INGEGNERIA

Dipartimento Ingegneria dell'Informazione

CORSO LARGE-SCALE AND MULTI-STRUCTURED DATABASES
MSc in ARTIFICIAL INTELLIGENCE AND DATA ENGINEERING

REPORT TASK 3

Designing and Implementing an application
which interacts with a Graph-based DB

PRESENTATO DA:

CARUSO Alberto
TUMMINELLI Gianluca
Di NOIA Antonio
CALABRESE Pietro

Anno Accademico
2019/2020

Indice

1	Introduzione	1
2	Requisiti	1
2.1	Requisiti Funzionali	1
2.2	Requisiti non Funzionali	2
3	Analisi del sistema	2
3.1	Identificazione degli attori	2
3.2	Casi d'uso	3
3.3	Diagramma Classi	4
3.4	Dataset	5
3.5	Data Model	5
3.5.1	Graph Database SnapShoot	6
3.6	Mapping Queries	7
4	Implementazione	9
4.1	Queries Rilevanti	10
4.1.1	Ricerca Bandiere	10
4.1.2	Ricerca Calciatori Consigliati	11
4.1.3	Ricerca Osservatori Consigliati	12
4.1.4	Elimina Account Utente	13
5	Conclusioni	14
6	Manuale d'Uso	15

1 Introduzione

Questa applicazione ha lo scopo di mettere in contatto gli osservatori delle varie società, non solo con i calciatori che sono attenzionati per motivi di mercato, prendendo visione di alcune statistiche sommarie per individuare preliminarmente le caratteristiche del giocatore. Inoltre ha anche finalità di creare un rete tra i colleghi per avere una valutazione del lavoro svolto. Questa è possibile sfruttando le proprietà dei graph-database e le funzioni di scansione ad essi associate, che aiutano gli sviluppatori a definire una rete di contatti ed analizzare quest'ultima a loro volta.

2 Requisiti

2.1 Requisiti Funzionali

Requisiti per l'utente:

- Il sistema deve permettere ad un utente già registrato di poter effettuare il login attraverso la propria e-mail e la propria password.
- Se l'utente non è registrato, il sistema deve permettere all'utente di potersi registrare. A tale scopo l'utente deve inserire un insieme di informazioni base che includono una e-mail ed una password personali.
- Il sistema deve permettere ad un utente di poter effettuare il logout per poter uscire dal proprio profilo.
- Il sistema deve permettere ad un utente di poter modificare o cancellare il proprio profilo.
- Il sistema deve permettere all'utente di poter ricercare i calciatori da inserire nella lista dei calciatori da prendere in considerazione. Tale ricerca deve poter essere effettuata o direttamente tramite il nome ed il cognome del calciatore, oppure tramite una serie di filtri.
- Il sistema permette all'utente di poter visualizzare la lista dei giocatori più seguiti.

Requisiti per l'amministratore di sistema:

- Il sistema deve permettere all'amministratore di sistema di poter aggiornare la parte di database relativa ai calciatori.
- Il sistema deve permettere all'amministratore di sistema di poter gestire il cambio di società relativamente agli osservatori.

2.2 Requisiti non Funzionali

Sono stati individuati i seguenti requisiti non funzionali, al fine di rendere le prestazioni del sistema allineate con quelle di una buona operatività:

FRUIBILITÀ: Le operazioni necessarie per completare un compito devono essere facili da capire per un generico utente.

MANUTENIBILITÀ: Il codice deve essere leggibile e modulato per permettere l'aggiunta di nuove caratteristiche e l'individuazione di eventuali errori in modo facile.

SICUREZZA: Il software deve essere usato solo come previsto. Alcuni tipi di requisiti di privacy includono: crittografia dei dati per le tabelle del database. I requisiti di accesso definiscono i tipi / gruppi di account e i loro diritti di accesso. Un esempio di requisiti di accesso potrebbe essere quello di limitare ciascun account a un accesso alla volta o di limitare la distribuzione o l'utilizzo dell'applicazione.

PRESTAZIONE: Il software deve rispondere con minore latenza possibile.

SCALABILITÀ: Il software deve essere in grado di gestire una grande quantità di transazioni

3 Analisi del sistema

In questa sezione si analizza in modo approfondito le scelte tecniche, i requisiti, le funzionalità e gli attori del progetto.

3.1 Identificazione degli attori

In questa sezione, ci soffermiamo nel definire gli attori che sono stati identificati, e che possono interagire con il sistema.

L'applicazione è destinata agli osservatori, anche definiti utenti dell'applicazione, che risultano essere i principali attori, a questi si affianca all'amministratore di sistema.

Utente dell'applicazione dopo essersi registrato ed aver eseguito il login può effettuare le seguenti azioni:

- Seguire un calciatore.
- Visualizzare gli utenti per nome.

-
- Aggiungere un utente ai propri followers.
 - Visualizzare i dieci calciatori che hanno più followers.
 - Visualizzare i calciatori più seguiti dai followers dell’utente.
 - Visualizzare i calciatori più seguiti dai followers dell’utente tesserato per una società di raking uguale alla sua.
 - Ricercare un calciatore per nome e cognome.
 - Ricercare i calciatori che hanno giocato per una squadra in una determinata stagione.
 - Visualizzare i dieci calciatori che hanno giocato con meno squadre.
 - Dato uno specifico giocatore è possibile ottenere le seguenti informazioni:
 - Squadre per cui ha militato.
 - Calciatori con cui ha giocato.
 - Numero di follower.

Amministratore di sistema gestisce le associazioni tra osservatori e società ed aggiorna le informazioni contenute nel sistema.

3.2 Casi d’uso

Inizialmente ci siamo concentrati sul individuare i principali requisisti che il sistema deve soddisfare. Lo scopo principale del sistema è dare la possibilità all’osservatore di ogni società, di poter seguire giocatori, avere delle statistiche sul giocatore, e poter creare una rete tra i colleghi e i calciatori che fanno parte dell’applicazione.

Dunque le principali azioni che il sistema eseguirà sono:

- seguire un calciatore;
- aggiungere un utente ai propri followers;
- visualizzare i calciatori più seguiti dai followers dell’utente;
- ricercare un calciatore per nome e cognome.

Oltre a questo, per entrambi gli utenti si definiscono le operazioni basilari quali registrazione al servizio, login e logout. Inoltre si definisce il vincolo che email in atto di registrazione può essere usata per un unico account.

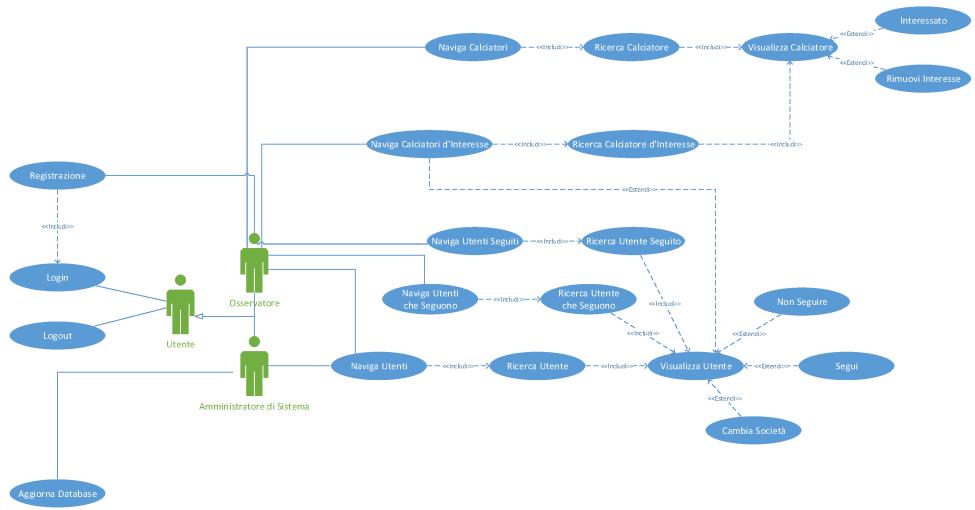


Figura 1: Use Cases Diagram

3.3 Diagramma Classi

Nella Figura2 viene mostrato il diagramma delle classi ottenuto dopo l'analisi dei requisiti e la definizione degli attori. Per favorire la leggibilità, i dettagli e le procedure di ogni singola classe sono state omesse. Infatti, l'obbiettivo principale è descrivere, attraverso l'uso delle associazioni, le relazioni che intercorrono tra le classi. Per fare questo è stato adottato il formalismo definito dal UML 2.

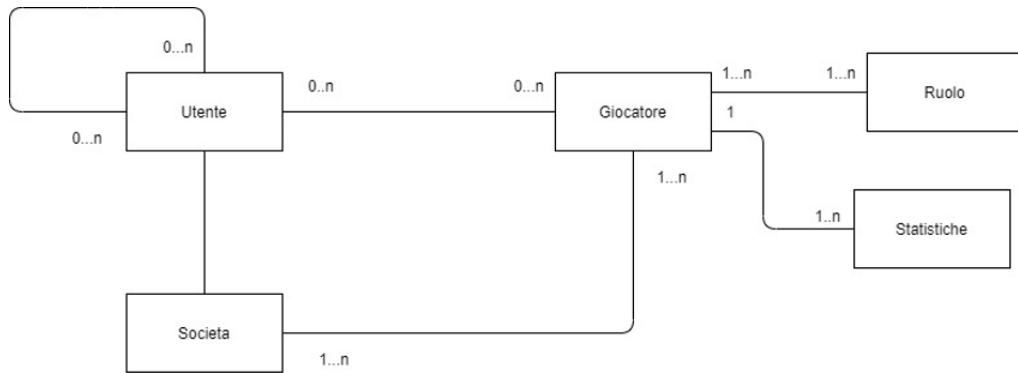


Figura 2: Class Diagram

3.4 Dataset

Il dataset per i dati relativi ai calciatori è stata sviluppata una funzione di scraping per ottenere i dati dal sito specializzato in statistiche calcistiche, www.transfermarkt.it; sono stati presi i dati relativi a tutti i giocatori sia dei campionati principali, sia delle serie minori di tutti i continenti. Si è ottenuto così un dataset iniziale di circa 300 Mb, che contiene 40833 calciatori, con alcune delle loro statistiche.

I dati così ottenuti organizzati in un file JSON, sono stati importati all'interno dell'applicazione e organizzati secondo i Data Model definiti delle sezione seguente.

3.5 Data Model

Visto quanto riportato nelle sezioni precedenti, tenuto conto dei requisiti del sistema e dei dati che si andranno a gestire, si sono progettati i data model come segue.

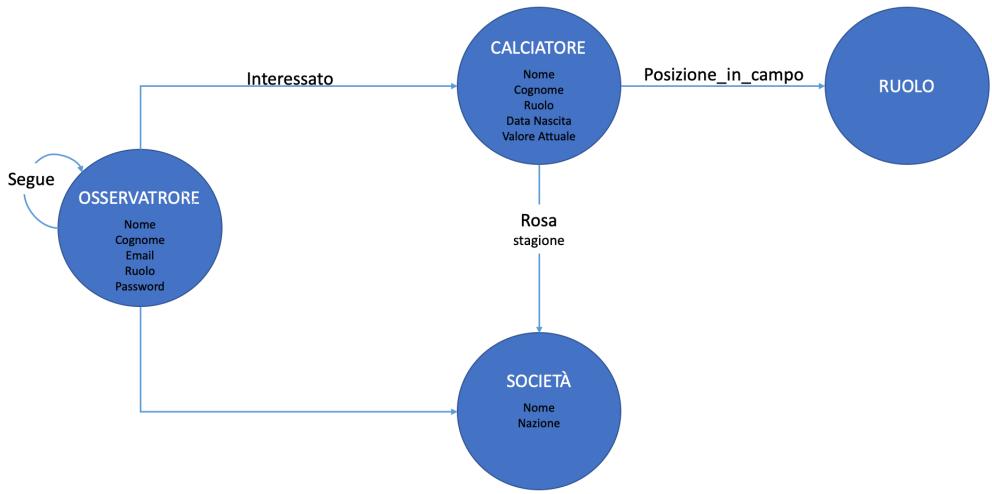


Figura 3: Graph-Model

Sono state definite quattro entità Osservatore, Calciatore, Società e Ruolo. Le relazioni ad esse associate, descrivono alcune il follow; mentre altre servono per tenere traccia degli storici delle rose o dei ruoli ricoperti.

3.5.1 Graph Database SnapShoot

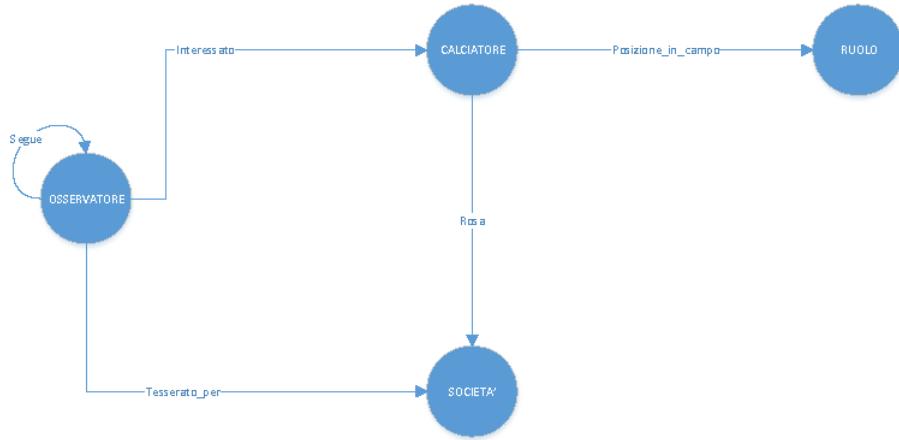


Figura 4: Graph-Database Snapshoot

Nella Figura 4 è mostrato uno snapshoot del graph database con le seguenti entità:

- **Utente**: il nodo utente ha i seguenti attributi: Nome, Cognome, Mail, Password
 - L'utente è collegato con il nodo calciatore tramite l'arco Segue che ha come attributo un campo TimeStamp che rappresenta l'istante in cui l'utente ha iniziato a seguire un calciatore.
- **Calciatore**: il nodo calciatore ha i seguenti attributi: Nome, DataDiNascita, LuogoDiNascita, Altezza, Nazionalità, Piede, LinkFoto, ValoreAttuale.
 - Il calciatore è collegato con l'entità Squadra tramite l'arco Gioca che ha i seguenti attributi: Stagione, Competizione, Presenze, PuntiPartita, Reti, Assist, Ammonizioni, DoppieAmmonizioni, Espulsioni, MinutiGiocati e un TimeStamp che rappresenta l'istante in cui è stata aggiornata la statistica.
 - Il calciatore è collegato con l'entità ruolo tramite l'arco PosizioneInCampo che l'attributo booleano Preferito che se è settato a true indica che il ruolo è il preferito dal giocatore.
- **Ruolo**: il nodo ruolo ha l'attributo NomeRuolo.
- **Società**: il nodo società ha i seguenti attributi: NomeSocieta, Nazione, Lega, LinkLogo.

3.6 Mapping Queries

In questa sezione si definiscono le principali queries da elaborare nel database e come vengono mappate in logica graph-database. Così come illustrato nella tabella di seguito riportata.

Graph-Centric Query	Domain-Specific Query
Quanti archi con etichetta segui entrano dentro il nodo calciatore	Quanti utenti seguono un calciatore?
Quali sono i nodi calciatore che hanno più archi entranti con etichetta segui ?	Quali sono i calciatori con più follower?
Quali sono i dieci nodi calciatore che sono collegati con meno nodi societa tramite archi con etichetta societa ?	Trova i calciatori che hanno giocato in meno squadre
Partendo dagli archi interessato che entrano nei nodi Calciatori che hanno come arco uscite un arco posizione che entra nel nodo Ruolo che ha come paramentro ruolo quello selezionato, per ogni nodo si contano gli archi entrati con etichetta INTERESSATO trovati e si restituiscono in modo decrescente gli n nodi Calciatori	Trova i calciatori più seguiti data una specifica posizione
Partendo dal nodo Utente s, con proprietà email selezionata, trova tutti i nodi Utente u, collegati con il nodo s tramite l'arco Segue . Trovare tutti gli Utenti che sono collegati ai nodi u, tramite archi Segue , e si restituiscono i nodi Utente	Dato un'osservatore trova gli osservatori più seguti dai suoi follow
Partendo dal nodo Utente s, con proprietà email selezionata, trova tutti i nodi Utente u, collegati con il nodo s tramite l'arco Segue . Trovare tutti i Calciatori che sono collegati ai nodi u, tramite archi Interessato , per ogni nodo si contano gli archi entrati con etichetta INTERESSATO trovati e si restituiscono in modo decrescente gli n nodi Calciatori	Dato un osservatore, trova i calciatori più seguiti dagli osservatori che egli segue

4 Implementazione

In questa sezione, si descrive, in modo più dettagliato, le scelte implementative adottata per lo sviluppo del sistema. Attualmente si tratta di una java app, basata su architettura Client-Server, il prototipo del sistema è stato sviluppato attenendosi al architettura multi-tier, questo per tenere separati il più possibile il lato Client da quello Server.

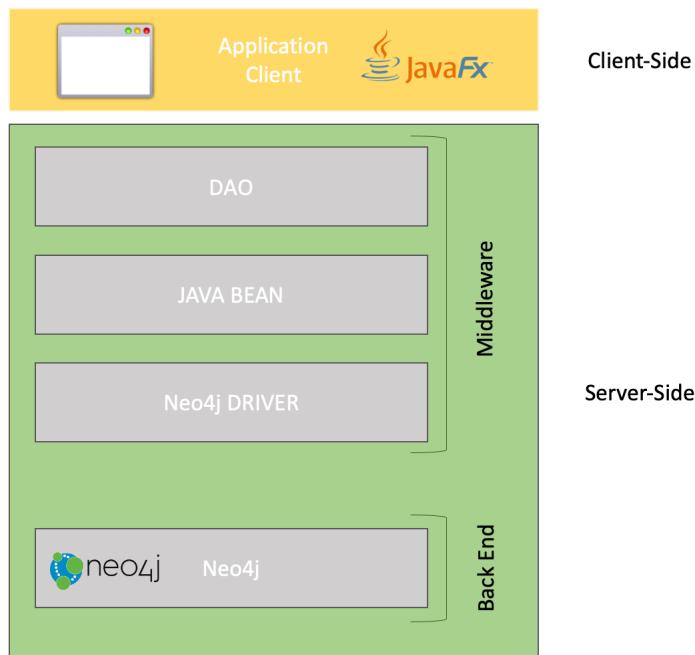


Figura 5: Multi-Tier Architecture Design

Come mostrato nella Figura 5, il lato Client è gestito con java application che si interfaccia con il lato server attraverso l'uso del framework Neo4j-driver. Mentre la grafica viene gestita attraverso il framework JavaFx. Per lo sviluppo della parte di Back End si è scelto di lavorare con dei Graph-database, nel caso specifico Neo4j, vista la buona affidabilità e la grande flessibilità nella gestione dei data model. Per assicurare la consistenza dei dati all'interno del DB, garantendo allo stesso tempo la concorrenza, il sistema è stato realizzato adottando il pattern architettonico MVC. Questo tipo di approccio risulta essere il metodo di controllo della concorrenza comunemente utilizzato dai sistemi di gestione per garantire accesso simultaneo al database e nei linguaggi di programmazione per implementare la memoria transazionale, senza questo ci potremo trovare di fronte a casi di inconsistenza dei dati all'interno del database.

4.1 Queries Rilevanti

In questa sezione viene riportata l'implementazione delle queries più significative per l'applicazione, sia dal punto di vista funzionale, che sia dal punto di vista didattico.

4.1.1 Ricerca Bandiere

La query ricerca tutti quei giocatori definiti bandiera, cioè che militano nella stessa squadra da almeno 10 anni.

```
3  public static List<InformazioniRicercaCalciatore> ricercaBandiere(String email) {
4      try(Session session=driver.session())
5      {
6
7          return session.writeTransaction(new TransactionWork<List>()
8          {
9              @Override
10             public List<InformazioniRicercaCalciatore> execute(Transaction tx)
11             {
12                 return transactionRicercaBandiere(tx, email);
13             }
14         });
15     }
16 }
17 private static List<InformazioniRicercaCalciatore>
18     → transactionRicercaBandiere(Transaction tx, String email)
19     {
20         List<InformazioniRicercaCalciatore> calciatoriCercati = new ArrayList<>();
21         Calendar calendar = Calendar.getInstance();
22         calendar.add(Calendar.YEAR, -27);
23         Long targetDate = calendar.getTime().getTime();
24
25         HashMap<String, Object> parameters =new HashMap<>();
26         parameters.put("email", email);
27         parameters.put("targetDate", targetDate);
28         StatementResult result=tx.run("match
29             → (e:Ruolo)-[:POSIZIONE]-(c:Calciatore)-[r:GIOCA]->(s:Societa)"
30             + " OPTIONAL MATCH (c:Calciatore)-[:INTERESSATO]-() "
31             + " with e,c,s,r,count(r) as cnt"
32             + " where toInt(c.dataNascita)<=$targetDate and cnt>10"
33             + " AND NOT ((c:Calciatore)-[:INTERESSATO]-(:Utente{email:$email}))"
34             + " return c, count(distinct s.nomeSocieta) as
35             → numeroSquadre,collect(e)[..1] as ruoloPrincipale,"
36             + " COUNT(DISTINCT i) AS seguitoDa"
37             + " order by numeroSquadre ASC limit 10 ",parameters);
38         while(result.hasNext())
39         {
40             Record record=result.next();
41             List<Pair<String,Value>> values = record.fields();
42
43             InformazioniRicercaCalciatore calciatoreRicercato=new
44             → InformazioniRicercaCalciatore();
45             for (Pair<String,Value> nameValue: values)
46             {
47                 if("c".equals(nameValue.key()))
48                 {
49                     Value value = nameValue.value();
```

```

46         calciatoreRicercato.setIdCalciatore(value.get("id").asString());
47         Date dataNascita = new
48             ↪ Date(Long.valueOf(value.get("dataNascita").asString()));
49         LocalDateTime ldt=LocalDateTime.ofInstant(dataNascita.toInstant(),
50             ZoneId.systemDefault());
51         calciatoreRicercato.setEta(ldt);
52         calciatoreRicercato.setNome(value.get("nome").asString());
53         calciatoreRicercato.setSquadra(value.get("squadra").asString());
54
55         ↪ calciatoreRicercato.setNazionalita(value.get("nazionalita").asString());
56         calciatoreRicercato.setValoreMercato(
57             Integer.parseInt(value.get("valoreAttuale").asString()));
58         ImageView fotoCalciatore = new ImageView(new
59             ↪ Image(value.get("linkFoto").asString()));
60         calciatoreRicercato.setImage(fotoCalciatore);
61     }
62     if("seguitoDa".equals(nameValue.key()))
63     {
64         Value value = nameValue.value();
65         calciatoreRicercato.setSeguitoDa(Integer.parseInt(value.toString()));
66     }
67     if("ruoloPrincipale".equals(nameValue.key()))
68     {
69         Value value = nameValue.value();
70         ↪ calciatoreRicercato.setRuoloPrincipale(value.get(0).get("ruolo").asString());
71     }
72     calciatoriCercati.add(calciatoreRicercato);
73 }
74
75 return calciatoriCercati;

```

4.1.2 Ricerca Calciatori Consigliati

La query ricerca tutti i calciatori, che in base ai criteri d'utilizzo dell'applicazione, vengo proposti all'utente.

```

79     public static List<InformazioniRicercaCalciatore> ricercaConsigliati(String email) {
80         try(Session session=driver.session())
81         {
82
83             return session.writeTransaction(new TransactionWork<List>()
84             {
85                 @Override
86                 public List<InformazioniRicercaCalciatore> execute(Transaction tx)
87                 {
88                     return transactionRicercaConsigliati(tx,email);
89                 }
90             });
91         }
92     }
93     private static List<InformazioniRicercaCalciatore>
94         ↪ transactionRicercaConsigliati(Transaction tx,String email)
95     {
96         List<InformazioniRicercaCalciatore> calciatoriCercati = new ArrayList<>();
97         HashMap<String, Object> parameters =new HashMap<>();
98         parameters.put("email", email);

```

```

98     StatementResult result=tx.run("match
99       (e:Ruolo)<-[POsizione]-(c:Calciatore)<-[i:INTERESSATO]-(:Utente)<-[SEGUE]-(os:Utente)"
100      + " where os.email=$email AND NOT
101        ((c:Calciatore)<-[INTERESSATO]-(:Utente{email:$email}))"
102      + " return c,collect(e)[..1] as ruoloPrincipale,COUNT(DISTINCT i) AS
103        seguitoDa"
104      + " order by seguitoDa DESC limit 20 ",parameters);
105    while(result.hasNext())
106    {
107      Record record=result.next();
108      List<Pair<String,Value>> values = record.fields();
109
110      InformazioniRicercaCalciatore calciatoreRicercato=new
111        ↪ InformazioniRicercaCalciatore();
112      for (Pair<String,Value> nameValue: values)
113      {
114        if("c".equals(nameValue.key()))
115        {
116          Value value = nameValue.value();
117          calciatoreRicercato.setIdCalciatore(value.get("id").asString());
118          Date dataNascita = new
119            ↪ Date(Long.valueOf(value.get("dataNascita").asString()));
120          LocalDateTime ldt=LocalDateTime.ofInstant(dataNascita.toInstant(),
121            ZoneId.systemDefault());
122          calciatoreRicercato.setEta(ldt);
123          calciatoreRicercato.setNome(value.get("nome").asString());
124          calciatoreRicercato.setSquadra(value.get("squadra").asString());
125
126          ↪ calciatoreRicercato.setNazionalita(value.get("nazionalita").asString());
127          calciatoreRicercato.setValoreMercato(
128            Integer.parseInt(value.get("valoreAttuale").asString()));
129          ImageView fotoCalciatore = new ImageView(new
130            ↪ Image(value.get("linkFoto").asString()));
131          calciatoreRicercato.setImage(fotoCalciatore);
132        }
133        if("seguitoDa".equals(nameValue.key()))
134        {
135          Value value = nameValue.value();
136          calciatoreRicercato.setSeguitoDa(Integer.parseInt(value.toString()));
137        }
138        if("ruoloPrincipale".equals(nameValue.key()))
139        {
140          Value value = nameValue.value();
141
142          ↪ calciatoreRicercato.setRuoloPrincipale(value.get(0).get("ruolo").asString());
143        }
144      }
145      calciatoriCercati.add(calciatoreRicercato);
146    }
147    return calciatoriCercati;

```

4.1.3 Ricerca Osservatori Consigliati

La query ricerca tutti gli osservatori, che in base ai criteri d'utilizzo dell'applicazione, vengo proposti all'utente.

```

147  public static List<InformazioniOsservatore> cercaConsigli(final String email) {
148    try(Session session=driver.session())

```

```

149         {
150             return session.writeTransaction((Transaction tx) ->
151                 consigliaOsservatore(tx,email));
152         }
153     }
154     private static List<InformazioniOsservatore> consigliaOsservatore(Transaction tx,
155         String email) {
156         List<InformazioniOsservatore> elems=new ArrayList<>();
157         HashMap<String, Object> parameters =new HashMap<>();
158         parameters.put("email",email);
159         StatementResult result=tx.run("match
160             (os1:Utente)-[:SEGUE]->(os2)-[r:SEGUE]->(os3)-[e:INTERESSATO]->()
161             + "where os1.email=$email return os3,count(DISTINCT r) as oss,
162             + count(DISTINCT e) as cont "
163             + "order by oss DESC",parameters);
164         while(result.hasNext()){
165             Record record=result.next();
166             List<Pair<String,Value>> values = record.fields();
167             InformazioniOsservatore info=new InformazioniOsservatore();
168             for (Pair<String,Value> nameValue: values) {
169                 if ("os3".equals(nameValue.key())){
170                     Value value = nameValue.value();
171                     info.setIdOsservatore(value.asNode().id());
172                     info.setCognome(value.get("cognome").asString());
173                     info.setNome(value.get("nome").asString());
174                     info.setSeguito(Boolean.TRUE);
175                     Societa societa=SocietaUtente(
176                         tx,value.get("email").asString(),Long.toString(info.getIdOsservatore()));
177                     if(societa!=null)
178                         info.setSquadra(societa.getNomeSocieta());
179                 if("cont".equals(nameValue.key())){
180                     Value value = nameValue.value();
181                     info.setCalciatoriSeguiti(value.toString());
182                 }
183             }
184         }
185     }

```

4.1.4 Elimina Account Utente

La query elimina l'account dell'utente.

```

192     public static int eliminaAccount(final String email,final String id){
193         try(Session session=driver.session())
194         {
195             return session.writeTransaction(new TransactionWork<Integer>()
196             {
197                 @Override
198                 public Integer execute(Transaction tx)
199                 {
200                     return transactionEliminaAccount(tx,email,id);
201                 }
202             }

```

```
203         });
204     }
205 }
206
207 private static int transactionEliminaAccount(Transaction tx, String email, String id){
208     HashMap<String, Object> parameters =new HashMap<>();
209     parameters.put("email",email);
210     StatementResult result=tx.run("MATCH(a:Utente) WHERE a.email=$email"+
211         " OPTIONAL MATCH (a)-[r]->()"+
212         " OPTIONAL MATCH ()-[r1]->(a)" +
213         " DELETE a,r,r1",parameters);
214
215     return 0;
216 }
```

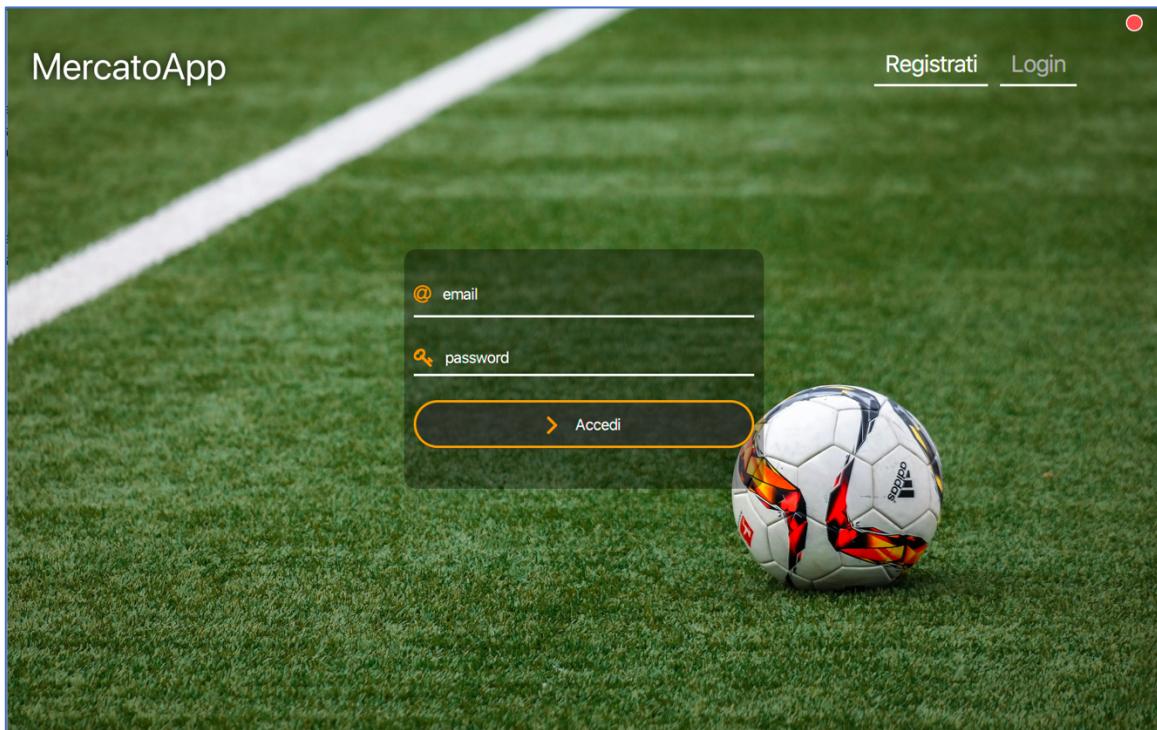
5 Conclusioni

In questo progetto è stato sviluppato un software per la gestione delle società di calcio, in particolare si è voluto l'attenzione sulla progettazione e l'implementazione del lato Back End, quindi della parte legata al database, visto anche le specifiche richiesta dal corso. Ciò non toglie che in caso di una reale implementazione del sistema di dovrebbe procedere ad alcune migliorie in modo da garantire la fruibilità del servizio a un largo numero di utenti, su un più vasto numero di dispositivi, quindi potrebbe essere sviluppata una web app basata su Java EE.

MANUALE D'USO

1. LOGIN

Ogni qual volta l'utente attiva l'applicazione tramite l'apposita icona presente sulla postazione di lavoro, viene evidenziata una finestra di "**Login**" mediante la quale è possibile effettuare l'accesso alle funzionalità dell'applicazione stessa.

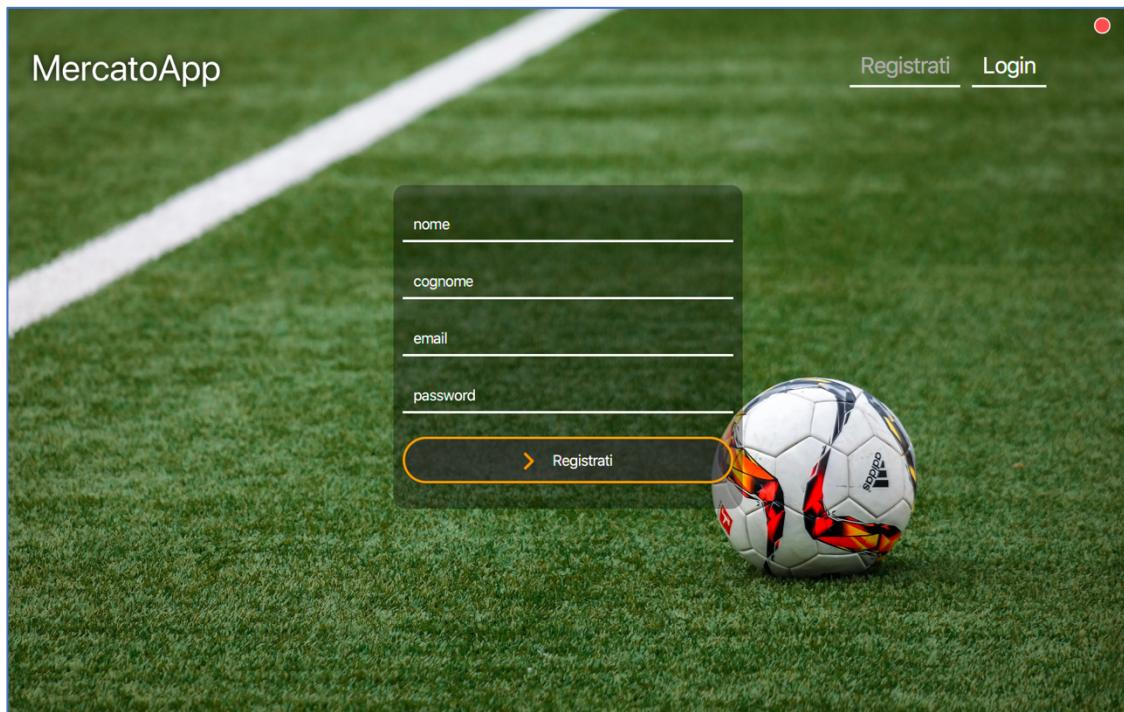


In particolare la finestra successiva di "**Login**" consente di effettuare le seguenti operazioni:

- creare un nuovo utente (cliccando sul pulsante "**Registrati**");
- effettuare l'accesso all'applicazione (cliccando sul pulsante "**Accedi**");
- annullare l'operazione che si sta effettuando, chiudendo l'applicazione (cliccando sul pallino rosso in alto a destra).

Registrazione

Nel caso in cui l'utente non è ancora registrato al servizio, deve definire la propria utenza di lavoro attraverso la selezione della voce "**Registrati**": in tal caso l'applicazione evidenzia la seguente schermata in cui è necessario indicare tutti i dati richiesti al fine di configurare correttamente la propria utenza.

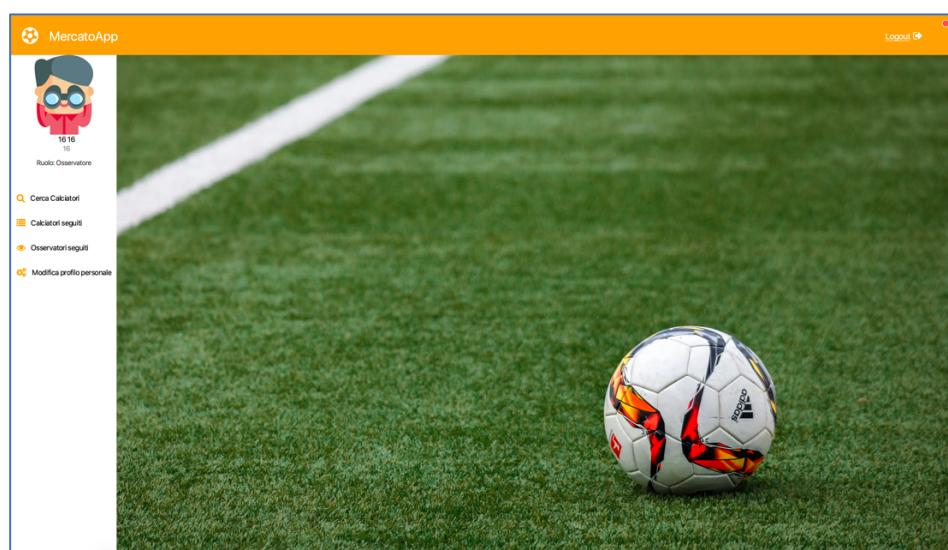


Tale finestra consente di inserire un nuovo utente specificando i seguenti dati:

- Nome
- Cognome
- Email, ovvero l'identificativo dell'utente che deve operare nell'ambito dell'applicazione.
- Password di sicurezza. Tale password può essere liberamente scelta dall'utente.

Effettuando il login con le credenziali appena impostate e cliccando sul pulsante “**Accedi**”, l'applicazione evidenzia la seguente “**Pagina di Benvenuto**” in cui sono presenti le funzionalità di seguito riportate:

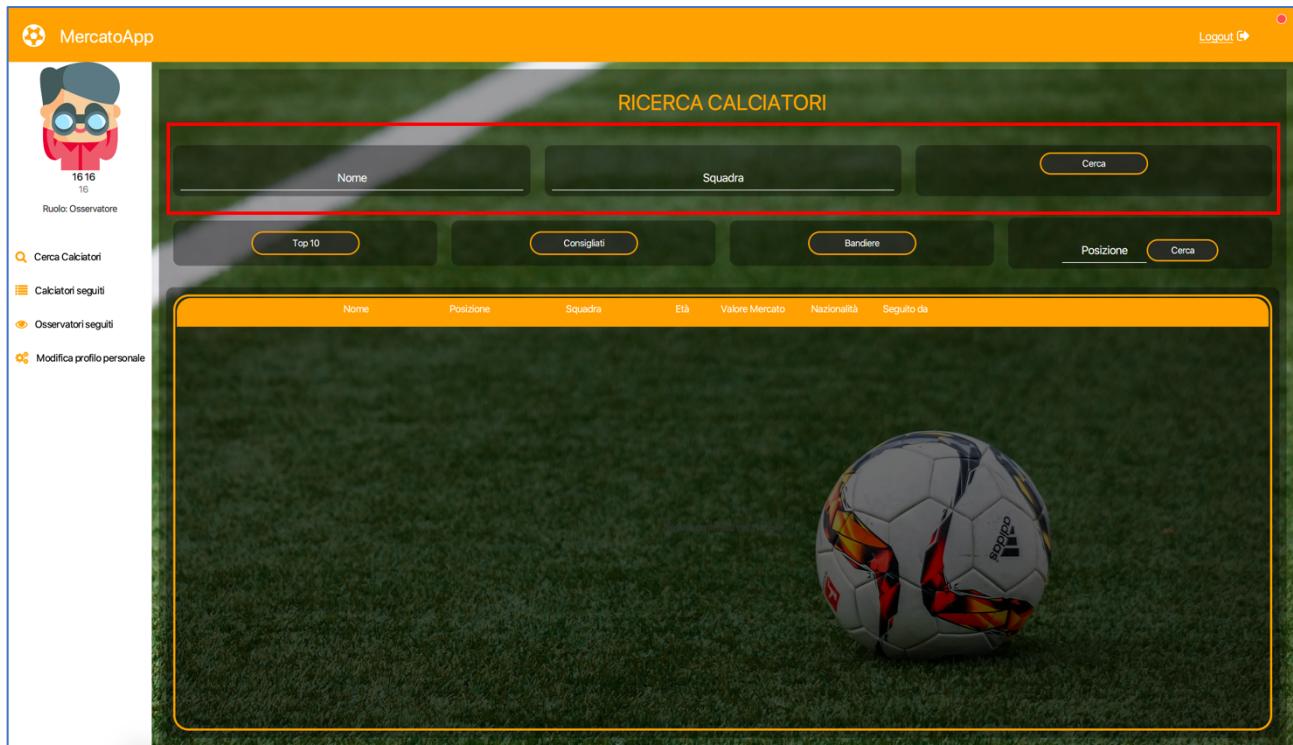
- Cerca Calciatori
- Calciatori Seguiti
- Osservatori Seguiti
- Modifica Profilo Personale



2. Funzioni dell'applicazione

Ricerca Calciatori

Dalla **Pagina di Benvenuto** cliccando sul pulsante “**Ricerca Calciatori**” si viene indirizzati alla seguente pagina.



In questa pagina l'utente può effettuare una ricerca basata su:

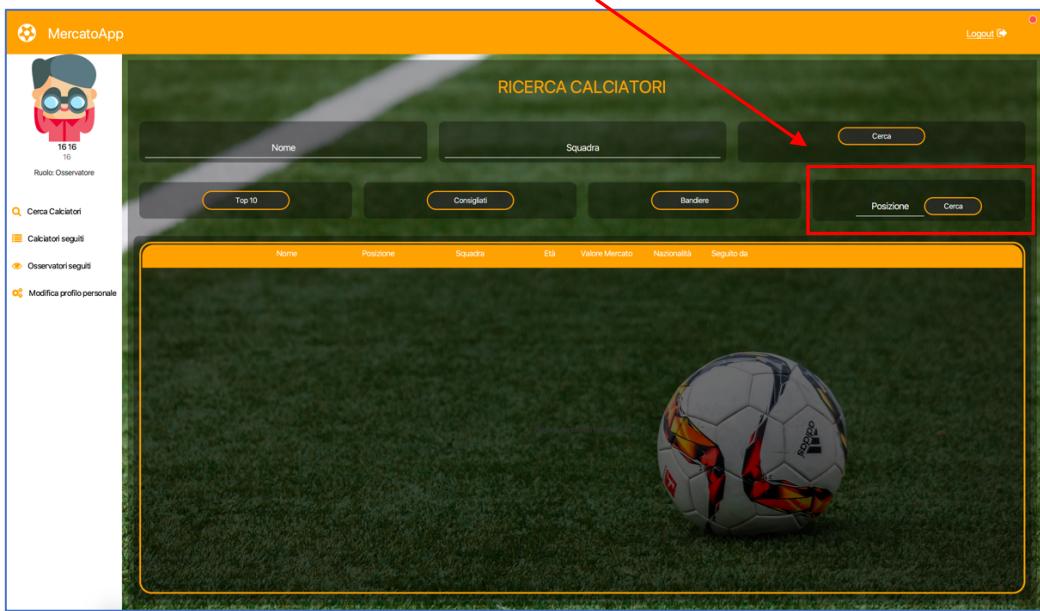
- Nome e Cognome Calciatore e/o Nome Squadra

e poi cliccando sul pulsante “**Cerca**”; come evidenziato nel box in rosso.

Altrimenti può effettuare una ricerca basata sul ruolo in campo del Calciatore

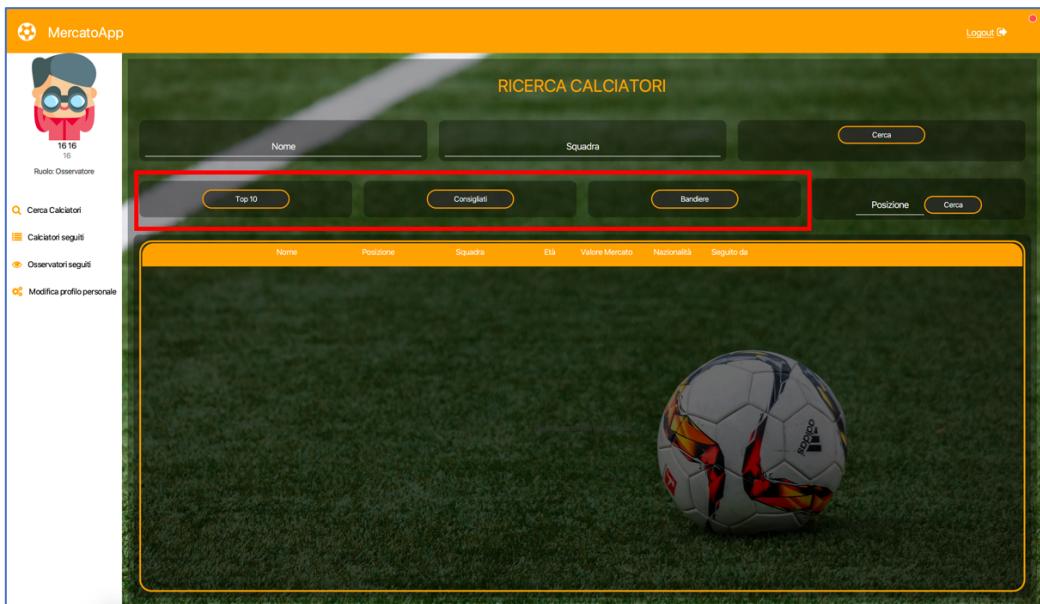
- Selezionando la “**posizione**”
- e poi cliccando sul pulsante “**Cerca**”

come evidenziato dall'immagine successiva.



Si può, inoltre, effettuare una ricerca basata su dei parametri ottenuti dai criteri d'utilizzo dell'utente; cliccando su uno dei tre pulsanti:

- **Top 10**
- **Consigliati**
- **Bandiere**



Una volta visualizzata la lista ricercata, l'utente verrà riportato alla pagina seguente, in cui cliccando sul pulsante evidenziato in rosso aggiunge il calciatore alla lista dei suoi seguiti.

MercatoApp

Ricerca Calciatori

Nome Squadra Cerca

Top 10 Consigliati Bancaire Posizione Cerca

Giannis Fetfatzidis, Ala destra, Aris Salonicco, 28, 1000000, Grecia, 8

Andrea Conti, Centrocampista, AC Milan, 26, 15000000, Italia, 8

Allan, Centrale, SSC Napoli, 28, 50000000, Brasile, 8

3. Gestione lista seguiti

Dalla barra di navigazione laterale, cliccando sul pulsante “**Calciatori Seguiti**” si viene indirizzati alla seguente pagina.

MercatoApp

Lista Giocatori Interesse

Nome Posizione Squadra Età Valore Mercato Nazionalità Seguito

Giorgio Chiellini, Difensore centrale, Juventus FC, 35, 5000000, Italia, Delete

Valentin Eyseric, Trequartista, Hellas Verona, 26, 1500000, Francia, Delete

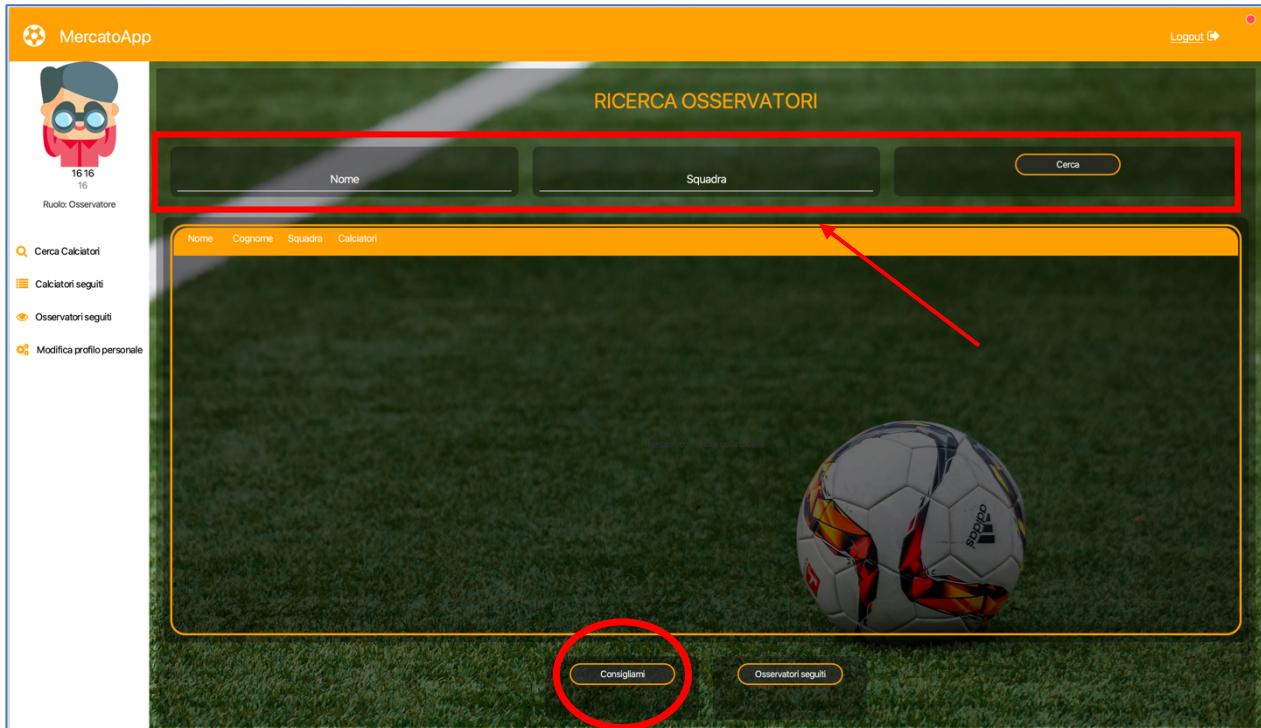
Koray GÄinter, Terzino destro, Hellas Verona, 28, 1500000, Germania, Delete

Guilherme, Trequartista, Olympiacos Pireo, 28, 4000000, Brasile, Delete

Qui si può visionare la lista dei calciatori seguiti, con alcune informazioni ad essi associate, e cliccando sul pulsante cerchiato in rosso, si può rimuovere il calciatore dai seguiti.

4. Gestione osservatori seguiti

Dalla barra di navigazione laterale, cliccando sul pulsante “**Osservatori Seguiti**” si viene indirizzati alla seguente pagina.



In questa pagina l’utente può effettuare una ricerca basata su:

- Nome e Cognome Osservatore e/o Nome Squadra

e poi cliccando sul pulsante “**Cerca**”; come evidenziato nel box in rosso.

Si può, inoltre, effettuare una ricerca basata su dei parametri ottenuti dai criteri d’utilizzo dell’utente; cliccando sul pulsante “**Consigliami**”, cerchiato in rosso.

L’utente sarà portato alla pagina con i risultati. Qui cliccando sul pulsante accanto a ogni risultato può aggiungere un osservatore alla lista dei suoi seguiti.

The screenshot shows the MercatoApp interface with a yellow header bar. On the left, there is a user profile icon with the number '16' and the role 'Osservatore'. Below the header, there is a search bar with the placeholder 'Ricerca OSSERVATORI' and two input fields: 'Nome' and 'Squadra' with the value 'Juventus FC'. A red arrow points from the top right towards the 'Osservatori seguiti' section. The main content area displays a table with columns: Nome, Cognome, Squadra, and Calciatori. The first row shows '21', '21', 'Juventu...', and '9'. To the right of the 'Calciatori' column, there is a circular icon with a person symbol, which is circled in red. Below the table, there are two buttons: 'Consigliami' and 'Osservatori seguiti'.

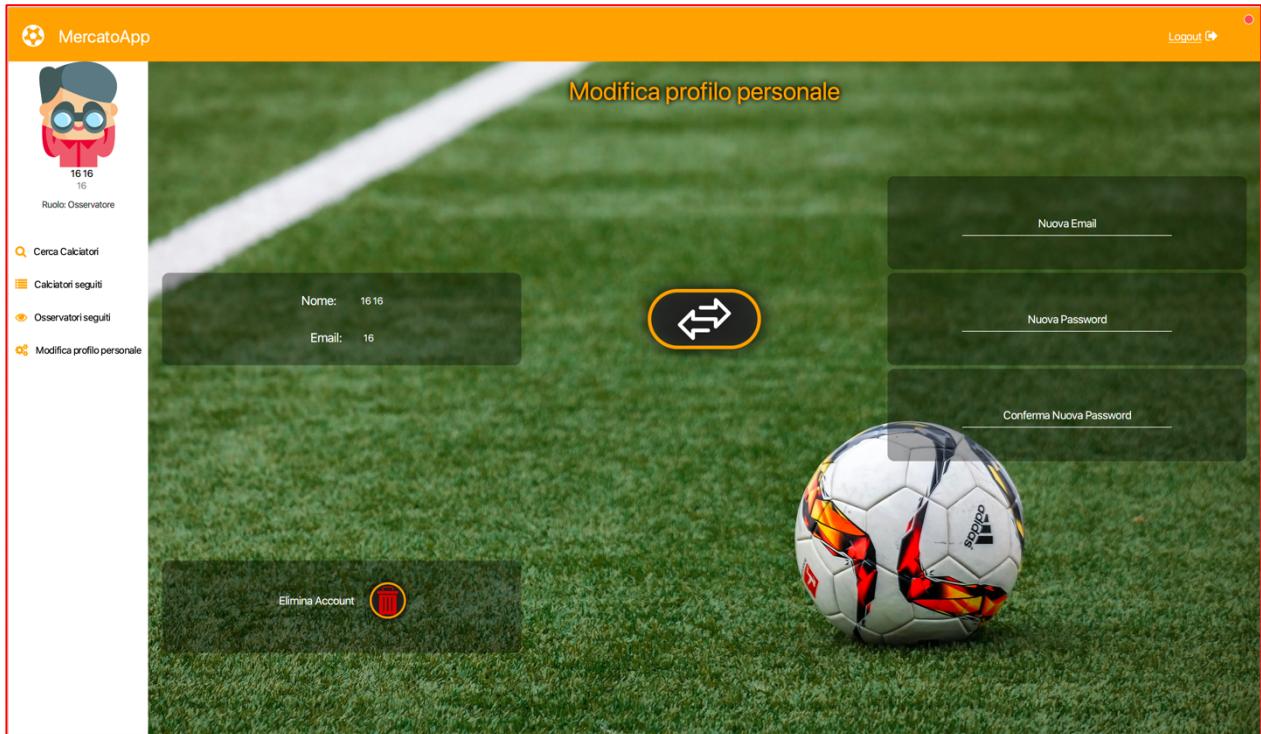
L'utente per gestire la lista degli osservatori che segue già, deve cliccare sul pulsante “**Osservatori Seguiti**”, sulla pagina “**Osservatori seguiti**”. Verrà così portato alla seguente pagina.

This screenshot shows the same MercatoApp interface as the previous one, but with multiple rows selected in the table. The first row is highlighted with a yellow background, and the circular icons in the 'Calciatori' column for all rows are circled in red. A red arrow points from the bottom left towards the first circled icon. The rest of the interface remains the same, with the yellow header, search bar, and navigation buttons.

Cliccando sul pulsante cerchiato in rosso, può rimuovere l'osservatore selezionato dalla lista dei seguiti.

5. Modifica Profilo Utente

Dalla barra di navigazione laterale, cliccando sul pulsante “**Modifica profilo personale**” si viene indirizzati alla seguente pagina.



In particolare la finestra di " **Modifica profilo personale**" consente di effettuare le seguenti operazioni:

- Eliminare account utente (cliccando sul pulsante "**Elimina Account**");
- Modificare i dati di accesso all'applicazione (compilando la form a destra).