# POLITECNICO DI TORINO

Dipartimento di Ingegneria Gestionale e della Produzione

**Corso di Laurea in Ingegneria Gestionale**
**Classe L-8**

Tesi di Laurea

## Tool per le simulazioni di mercato di una rosa calcistica

**Relatore**
Prof. Corno Fulvio

**Candidato**
Maturo Alessandro Giovanni

AA. 2021/2022

# Sommario

# 1 Proposta di progetto

## 1.1 Titolo della proposta

Tool per le simulazioni di mercato di una rosa calcistica.

## 1.2 Descrizione del problema proposto

Con il seguente software si vuole permettere all'utente finale di ottimizzare una rosa calcistica con determinate cessioni sul mercato valutando oppure imponendo vari parametri, alcuni economici come il valore di mercato del giocatore oppure il suo stipendio e altri legati alle performance con vari indicatori tratti da un data-set.
Il problema è molto frequente all'interno dei club calcistici che devono effettuare operazioni di mercato in modo da trarre profitto sulle vendite e allo stesso tempo cercare sostituti evitando di ridurre la forza della propria rosa.

## 1.3 Descrizione della rilevanza gestionale del problema

Questo tool sarebbe usato da un manager per provare a rinforzare la propria rosa negli indici in cui risulta carente. La grossa rilevanza gestionale del problema è che le valutazioni terranno conto delle risorse disponibili non simulando quindi operazioni di mercato irrealizzabili. Il software permetterà non solo di migliorare la propria rosa, ma permetterà anche, se voluto dall'utente, di limitare gli stipendi senza andare a ottenere un grosso decremento degli indici di performance legati alla squadra. Tutto ciò è possibile perché si potranno vedere come variano i parametri della propria squadra una volta che verranno prima simulate e poi nel caso effettuate le operazioni di mercato.

## 1.4 Descrizione dei data-set per la valutazione

Il data-set è stato tratto dal sito di Kaggle in cui sono presenti molti database che spaziano in diversi ambiti. Nel nostro caso, è tratto da un'applicazione chiamata Football Manager 2020 ed è presente al seguente link (https://www.kaggle.com/datasets/ktyptorio/football-manager-2020?resource=download) dove non vi è nessun vincolo di copyright.
All'interno del nostro database oltre che ai valori economici, quali valutazione sul mercato e stipendio percepito, troviamo diversi indici legati alle performance dei singoli giocatori, di questi ne utilizzeremo solo alcuni se no il programma non sarebbe facilmente comprensibile da un utente non esperto nel settore.

## 1.5 Descrizione preliminare degli algoritmi coinvolti

Questo software sarà realizzato in linguaggio Java con l'ausilio delle interfacce grafiche in JavaFX, inoltre saranno usati i pattern MVC e il pattern DAO.
Tra gli algoritmi coinvolti troviamo quelli basilari di ricerca di vari record che vengono effettuati tramite semplici interrogazioni SQL sul nostro database. Oltre a questi, troveremo algoritmi di ottimizzazione ricorsivi per trovare soluzioni ottimali che rispettano i vincoli imposti dall'utente. Inoltre, ci saranno anche algoritmi di simulazione, che con semplici operazioni matematiche, mostreranno come varierebbero i parametri dei diversi club con determinate operazioni di mercato. Per concludere saranno presenti diversi controlli per evitare problemi e aiutare l'utente nel caso si fosse dimenticato di inserire dei campi o effettuare operazioni.

## 1.6 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione permetterà all'utente di immedesimarsi in un vero e proprio manager di una società calcistica. Dopo aver selezionato il club di cui si vorrà migliorare o semplicemente modificare la rosa (con un semplice utilizzo di una coppia di combo box), l'utente potrà simulare operazioni di mercato. A partire dalle varie informazioni sulla società che verranno mostrate, l'utente avrà due possibilità: la prima consiste nel ricercare alcuni giocatori che rispettano determinati parametri e osservare il caso di loro acquisto come varierebbero gli indicatori della rosa, la seconda invece prevederà la simulazione di una vendita di un proprio giocatore e, in base al valore ottenuto da questa cessione, l'algoritmo cercherà dei sostituti in modo da ottimizzare i parametri richiesti dal manager che eventualmente potrebbero non essere i punti di forza del giocatore di cui si cerca la cessione, ma parametri economici per abbassare per esempio lo stipendio medio della rosa.

## 1.7 Precisazioni

Sarà previsto un primo algoritmo di simulazione dove l'utente potrà scegliere degli indici di performance nei quali la squadra risulta carente, e trovare, con questo, delle soluzioni che rispettino un budget inserito per gli acquisti oltre che a un vincolo salariale, imposto anche questo dall'utente. L'algoritmo proporrà una soluzione per migliorare la rosa e, nell'eventuale ipotesi in cui non sia possibile, si occuperà di comunicare che con quei parametri non sono presenti combinazioni per migliorare gli indici della squadra.
Inoltre, nel secondo algoritmo, quello principale, l'utente potrà selezionare dalla lista dei giocatori presenti in rosa un certo numero di calciatori appetibili per la vendita. Una volta selezionati e premuto un apposito bottone l'algoritmo calcolerà la migliore soluzione presente sul mercato che permetterà di ottimizzare gli indici della squadra non sforando il budget, in questo caso sarà il totale degli introiti ricavati dalla vendita (per semplicità si userà che il valore ricavato dalle cessioni è uguale alla somma dei valori attuali dei giocatori) e lo stipendio, ovvero la somma delle retribuzioni della possibile soluzione sarà minore o uguale della somma degli stipendi dei giocatori reputati 'cedibili' dall'utente.
In questa parte sto valutando di inserire la possibilità di scelta se andare a cercare una soluzione

che migliora la somma del valore complessivo degli indici oppure solo determinati indici selezionati proprio a partire dall'utente.
Vorrei anche provare a inserire, una volta calcolate le possibili soluzioni degli algoritmi, dei diagrammi a barre per indicare come varierebbero gli indici delle prestazioni, il valore della rosa e il monte ingaggi delle soluzioni ottime proposte, in modo che l'utente possa visualizzare con più facilità i cambiamenti per vedere se effettivamente apportare le modifiche alla propria squadra.

# 2 Descrizione del problema affrontato

L'applicazione permette all'utente di ottenere suggerimenti riguardanti le operazioni di mercato di una rosa calcistica.
Un manager che tenta di migliorare la rosa della propria società ha due grossi problemi da risolvere: il primo riguarda l'acquisto di giocatori, che dovranno migliorare determinati indici in cui la squadra risulta carente, il secondo invece riguarda la sostituzione dei propri calciatori con altri che permettano di migliorare i punti di debolezza della squadra. Ovviamente in entrambi questi punti, trattandosi di società calcistiche, ci saranno dei vincoli economici e tecnici da rispettare.
Per quanto concerne l'acquisto dei giocatori, il manager dovrà avere un budget massimo sia per il costo del cartellino che per lo stipendio. All'interno di questo tool, l'utente inserirà in input l'indice della squadra che vuole migliorare, il ruolo del giocatore, il costo massimo del cartellino e lo stipendio massimo che potrebbe percepire. Successivamente il programma fornirà i 5 migliori giocatori che rispettino i vincoli inseriti ordinati a seconda dell'indice scelto; nel caso non ci fossero soluzioni che rispettino i dati inseriti, il programma lo comunicherà all'utente.
Nel secondo punto si cerca una "soluzione ottima" per sostituire i calciatori con altri dello stesso ruolo aventi costo dei cartellini e somma degli stipendi minori. Allo stesso tempo, questi permetteranno di migliorare, o comunque non peggiorare, gli indici di performance della squadra. L'utente selezionerà i calciatori destinati alla vendita e il programma fornirà in output la combinazione migliore di calciatori che potrebbero sostituire quelli selezionati. In output verrà fornita la soluzione migliore, ovvero quella che potrebbe massimizzare gli indici, oppure in caso di soluzioni inesistenti, il programma lo comunicherà all'utente.

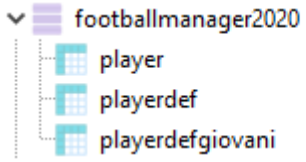
# 3 Descrizione del data-set utilizzato

Il database utilizzato è stato tratto da un'applicazione chiamata "Football Manager" in cui l'utente si immedesima nel ruolo di allenatore e/o manager sportivo provando a effettuare operazioni di mercato per migliorare la propria rosa.
I dati si riferiscono alla stagione sportiva 2020/2021 e riguardano i giocatori di tutto il mondo.

Il CSV del database è stato trovato sul sito Kaggle presso il seguente link:
https://www.kaggle.com/datasets/ktyptorio/football-manager-2020?resource=download

## 3.1 Struttura del database

Il database inizialmente era formato da una sola tabella chiamata player; successivamente è stato modificato per comodità di utilizzo selezionando solo i giocatori che dei cinque migliori campionati europei: Premier League, La Liga, Serie A, Bundesliga e Ligue 1.
Inoltre, il database è stato "pulito" in quanto erano presenti errori di codifica dovuti alla conversione dello stesso da un formato .csv a un formato SQL.
Successivamente è stato effettuato un filtraggio in cui sono stati eliminati tutti i calciatori con valore inferiore a 100000 € e età non superiore a 18 anni. Questa operazione si è resa necessaria per eliminare tutti i giocatori del settore giovanile.
Infine sono stati aggiornati anche tutti i valori di mercato dei giocatori del Chelsea in quanto riscontravano una valutazione pari a 0. I valori sono stati tratti dal sito Transfermarkt e fanno riferimento alla stagione 2020/2021.
Il risultato finale di queste modifiche è stato inserito all'interno della tabella playerdef.




## 3.2 Tabella player

È la tabella originale, ha la stessa struttura della playerdef trattata all'interno del punto 3.4, l'unica differenza è la presenza di errori di codifica e la presenza di tutti i campionati mondiali, non solo dei cinque migliori campionati europei.

## 3.3 Tabella playerdefgiovani

Contiene tutti i calciatori, compresi quelli dei settori giovanili, dei cinque migliori campionati europei. Anche questa struttura è analoga a quella della tabella playerdef trattata all'interno del punto 3.4.

## 3.4 Tabella playerdef

| Nome | Tipo di dati |
|---|---|
| number | INT |
| name | VARCHAR |
| position | VARCHAR |
| club | VARCHAR |
| division | VARCHAR |
| based | VARCHAR |
| nation | VARCHAR |
| height | DOUBLE |
| weight | INT |
| age | INT |
| pref_foot | VARCHAR |
| best_pos | VARCHAR |
| best_role | VARCHAR |
| value | INT |
| wage | INT |
| ca | INT |
| pa | INT |
| wor | INT |
| vis | INT |
| thr | INT |
| tec | INT |
| tea | INT |
| tck | INT |
| str | INT |
| sta | INT |
| tro | INT |
| ref | INT |
| pun | INT |
| pos | INT |
| pen | INT |
| pas | INT |
| pac | INT |
| 1v1 | INT |
| otb | INT |
| nat | INT |
| mar | INT |
| lth | INT |
| lon | INT |
| ldr | INT |
| kic | INT |
| jum | INT |
| hea | INT |
| han | INT |
| fre | INT |
| fla | INT |
| fir | INT |
| fin | INT |
| ecc | INT |
| dri | INT |
| det | INT |
| dec | INT |
| cro | INT |
| cor | INT |
| cnt | INT |
| cmp | INT |
| com | INT |
| cmd | INT |
| bra | INT |
| bal | INT |
| ant | INT |
| agi | INT |
| agg | INT |
| aer | INT |
| acc | INT |

Di seguito elenco solo gli attributi utilizzati all'interno del programma:

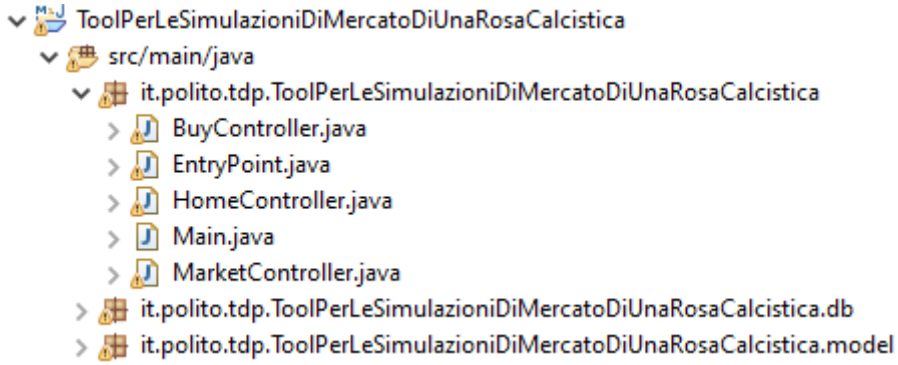
- number: identificativo univoco del giocatore
- name: nome del calciatore
- club: squadra di appartenenza del giocatore
- division: campionato in cui gioca il giocatore
- based: dove è situato il campionato
- height: altezza del giocatore (cm)
- weight: peso del giocatore (kg)
- age: età del giocatore
- best_pos: posizione/ruolo in cui il giocatore gioca meglio
- value: valore del giocatore (€)
- wage: stipendio lordo del giocatore (€/settimana)
- tec: indice di tecnica del giocatore (da 0 a 20)
- str: indice di forza del giocatore (da 0 a 20)
- pas: indice di qualità del passaggio del giocatore (da 0 a 20)
- mar: indice di qualità della marcatura del giocatore (da 0 a 20)
- pos: indice di qualità del posizionamento del giocatore (da 0 a20)

# 4 Descrizione strutture dati e algoritmi utilizzati

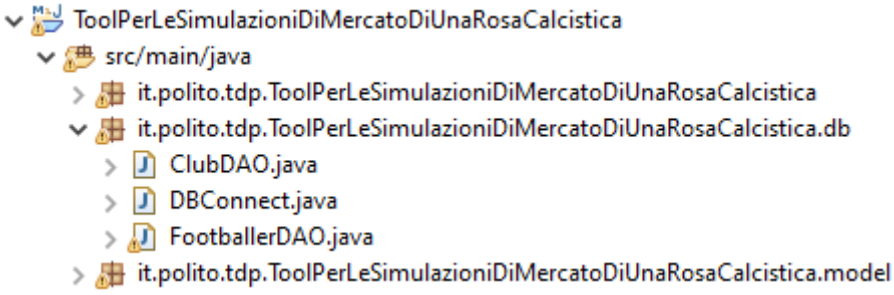
## 4.1 Descrizione strutture dati utilizzate

L'applicazione è stata realizzata utilizzando il linguaggio di programmazione Java per la gestione della logica applicativa e JavaFX per la gestione della componente grafica.
Il tutto è stato realizzato mediante il pattern MVC (Model-View-Controller) che permette di separare all'interno del software la logica applicativa, l'interfaccia grafica e la struttura dati.
Infatti, l'applicazione è suddivisa in tre package distinti:

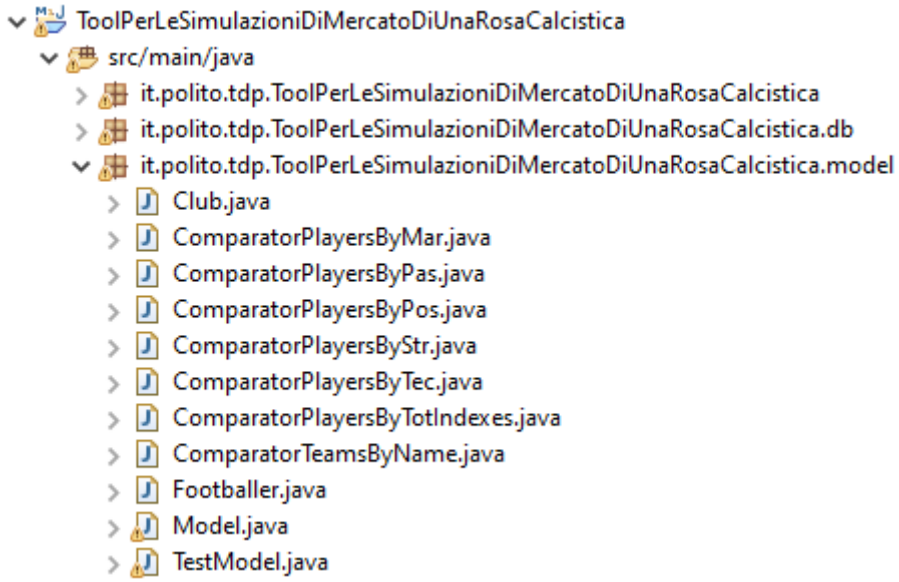
- Il package *it.polito.tdp.ToolPerLeSimulazioniDiMercatoDiUnaRosaCalcistica* contiene tutte le classi dedicate alla realizzazione della parte grafica dell'applicazione, dove l'utente inserirà i vari parametri di input e visualizzerà i risultati ottenuti.




- Il package *it.polito.tdp.ToolPerLeSimulazioniDiMercatoDiUnaRosaCalcistica.db* si comporta come DAO (Data Access Object) gestendo quindi l'interazione e l'estrazione dei dati dal database per il corretto funzionamento del software.




- Il package *it.polito.tdp.ToolPerLeSimulazioniDiMercatoDiUnaRosaCalcistica.model* si comporta da model, quindi, contiene tutte le classi legate alla componente algoritmica che gestiscono l'elaborazione dei dati.




## 4.2 Descrizione algoritmi utilizzati

Per la realizzazione del programma sono stati utilizzati molti algoritmi, sicuramente tra questi il ruolo più importante è ricoperto dall'algoritmo ricorsivo.

### 4.2.1 Algoritmo ricorsivo

*"In informatica viene detto **algoritmo ricorsivo** un algoritmo espresso in termini di se stesso, ovvero in cui l'esecuzione dell'algoritmo su un insieme di dati comporta la semplificazione o suddivisione dell'insieme di dati e l'applicazione dello stesso algoritmo agli insiemi di dati semplificati."*
(Fonte Wikipedia)

All'interno di questo tool l'algoritmo ricorsivo è stato utilizzato nella sezione di mercato. Essendo un algoritmo molto complesso, che richiama sé stesso più volte al suo interno sono state adottati alcuni filtraggi per permetterne un corretto funzionamento anche in presenza di più giocatori selezionati. Nella sezione di mercato è possibile, infatti, scegliere un certo numero di giocatori destinati alla vendita; a questo punto per ciascun giocatore verranno selezionati i migliori dieci nel loro ruolo che migliorano la media degli indici della squadra e rispettano due vincoli. Il valore del giocatore "possibile" e allo stesso tempo il suo stipendio devono essere inferiori o uguali al giocatore che è stato destinato alla vendita. Questa operazione verrà ripetuta più volte per ciascun giocatore scelto dall'utente, andando così a creare una lista di giocatori possibili sulla quale "ciclerà" l'algoritmo ricorsivo. L'algoritmo ricorsivo controllerà la soluzione parziale ogni volta che quest'ultima avrà dimensione uguale al numero di giocatori destinati alla vendita. Oltre a questo, vi sarà un controllo sui ruoli, ovvero il numero di giocatori per ciascun ruolo destinati alla vendita dovrà essere uguale a quello della soluzione. Infine, la soluzione non verrà scartata solo se il totale degli indici sarà maggiore o uguale (da controllare) al totale degli indici della precedente "soluzione ottima" presa in considerazione.

- **Metodo di inizializzazione della parte ricorsiva:**

  Verranno inizializzate le varie liste:

  - possible (i calciatori possibili sui quali ciclerà l'algoritmo ricorsivo)
  - best (la soluzione ottima)
  - partial (la soluzione parziale)

  L'attributo sizeSelected indicherà quanti giocatori sono stati selezionati per la vendita. Gli altri metodi presenti all'interno verranno commentati successivamente.

```
public List<Footballer> init(List<Footballer> selected) {

    /* ogni volta che si entra puliamo tutte le liste */
    possible = new LinkedList<>();
    best = new LinkedList<>();
    List<Footballer> partial = new LinkedList<>();

    this.ok=true;

    for(Footballer fi: selected) {
        this.getPossibleFootballers(fi);
    }

    /* se il giocatore non avrà calciatori possibili usciremo
     * returnando null e MarketController.java lo comunicherà all'utente */
    if(this.ok==false) {
        return null;
    }

    this.numP=0;
    this.numD=0;
    this.numC=0;
    this.numA=0;

    /* carico il numero di giocatori selezionati per ciascun ruolo */
    this.caricaRuoli(selected);

    /* Imposto come totale indici della soluzione migliore il totale degli indici dei giocatori selezionati */
    this.totBest=this.totIndexes(selected);
    int tot=0;

    this.sizeSelected=selected.size();

    recursion(partial, tot);

    return best;
}
```

- **Metodo ricorsivo:**
  Il metodo ciclerà sulla lista dei calciatori possibili e aggiungerà il calciatore solo se non ancora presente nella soluzione parziale. Ogni volta che la dimensione della soluzione parziale sarà uguale al numero di calciatori destinati alla vendita (l'attributo sizeSelected) si entrerà in una sezione con due if da superare per diventare "soluzione ottima". Il primo viene superato se la somma totale degli indici della soluzione parziale è maggiore della somma degli indici della soluzione migliore. Il secondo invece è un controllo sui ruoli, il metodo checkRuoli() che verrà spiegato successivamente.

```java
private void recursion(List<Footballer> partial, int tot) {

    if(partial.size()==this.sizeSelected) {
        if(tot>this.totBest) {
            if(checkRuoli(partial)) {
                this.best = new LinkedList<>(partial);
                this.totBest=tot;
            }
        }

    } else {

        for(Footballer fi: this.possible) {
            if(!partial.contains(fi)) {

                partial.add(fi);

                int contribute = fi.getMarking()+fi.getPassing()+fi.getPositioning()+fi.getStrength()+fi.getTechnique()
                tot=tot+contribute;

                recursion(partial, tot);

                tot=tot-contribute;
                partial.remove(partial.size()-1);

            }

        }

    }
`
```

- **Metodo getPossibleFootballers() del Model:**

  Questo metodo di tipo void aggiungerà alla lista dei calciatori possibili (attributo del main) i migliori 10 calciatori che avranno stipendio minore, valore minore e stesso ruolo del giocatore selezionato per la vendita. Allo stesso tempo la media degli indici dei calciatori da aggiungere dovrà essere maggiore della media degli indici dell'intera squadra. Questo metodo al suo interno richiama il metodo getPossibleFootballers() del DAO che verrà spegato successivamente.

```java
private void getPossibleFootballers(Footballer footballer) {

    /* media degli indici del calciatore passato */
    double meanIndexes = this.averageStats(footballer);

    List<Footballer> result = footballerDAO.getPossibleFootballers(footballer, meanIndexes);

    if(result!=null) {
        for(Footballer fi: result) {
            /* Lo aggiungo solo se non lo contiene già perchè lo stesso calciatore potrebbe andare
             * bene per più di uno dei giocatori selezionati dall'utente */
            if(!this.possible.contains(fi)) {
                this.possible.add(fi);
            }
        }
    } else {
        /* se non ci sono calciatori possibili imposto ok a false */
        this.ok=false;
    }

}
```

- **Metodo getPossibleFootballers() del DAO:**

Vengono selezionati tutti i giocatori con salario minore, valore minore, stesso ruolo e club diverso rispetto al giocatore di partenza. Di questi verranno aggiunti alla lista solo quelli che hanno la media dei 5 indici utilizzati dal programma superiore alla media degli indici della squadra. Dopo questa selezione verranno returnati solo i primi 10, fornendo così i 10 migliori

giocatori sulla carta visto che la selezione era stata fatta ordinando per p.number ovvero la classifica della media degli indici dei giocatori

```
public List<Footballer> getPossibleFootballers(Footballer footballer, double meanIndexes) {

    List<Footballer> footballers = new LinkedList<>();

    String sql = "SELECT p.number, p.name, p.best_pos, p.club, p.age, p.value, p.wage, p.tec, p.pas, p.mar, p.pos, p.str "
            + "FROM playerdef p "
            + "WHERE p.wage <= ? AND p.value <= ? AND p.best_pos = ? AND p.club <> ? "
            + "ORDER BY p.number";


    try {
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);

        st.setInt(1, footballer.getWage());
        st.setDouble(2, footballer.getValue()*1000000);
        st.setString(3, footballer.getBest_pos());
        st.setString(4, footballer.getClub());

        ResultSet rs = st.executeQuery();


        while(rs.next()) {


            double valueMln = ((double)(rs.getInt("value")))/(1000000);


            Footballer f = new Footballer(rs.getInt("number"),
                    rs.getString("name"),
                    rs.getString("best_pos"),
                    rs.getString("club"),
                    rs.getInt("age"),
                    valueMln,
                    rs.getInt("wage"),
                    rs.getInt("tec"),
                    rs.getInt("pas"),
                    rs.getInt("mar"),
                    rs.getInt("pos"),
                    rs.getInt("str"));
            /* Lo aggiungo solo se la media indici è superiore a quella passata*/
            if(this.getMediumStatsByPlayer(f)>=meanIndexes) {
                if(!f.equals(footballer)) {
                    footballers.add(f);
                }
            }
        }
        conn.close();
        st.close();
        rs.close();

    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    List<Footballer> bestTen = new LinkedList<>();

    if(footballers.size()>0) {

        for(Footballer fi: footballers) {
            /* prendo solo i migliori 10 */
            if(bestTen.size()<10) {
                bestTen.add(fi);
            }
        }
    }

    return bestTen;
}
```

- **Metodo checkRuoli():**
  Questo metodo, di tipo boolean, avrà come valore di ritorno true in caso il numero di giocatori destinati alla vendita sia uguale al numero di giocatori della soluzione parziale, questo ovviamente per ciascun ruolo. In caso contrario il valore di ritorno del metodo sarà false.

```
private boolean checkRuoli(List<Footballer> partial) {

    int cntP=0;     int cntD=0;     int cntC=0;     int cntA=0;

    for(Footballer fi: partial) {
        if(fi.getBest_role().compareTo("P")==0) {
            cntP++;
        }
        if(fi.getBest_role().compareTo("D")==0) {
            cntD++;
        }
        if(fi.getBest_role().compareTo("C")==0) {
            cntC++;
        }
        if(fi.getBest_role().compareTo("A")==0) {
            cntA++;
        }
    }

    if(cntP!=this.numP) {
        return false;
    }
    if(cntD!=this.numD) {
        return false;
    }
    if(cntC!=this.numC) {
        return false;
    }
    if(cntA!=this.numA) {
        return false;
    }
    return true;
}
```

- **Metodo caricaRuoli():**

Questo metodo di tipo void andrà a contare il numero di giocatori per ciascun ruolo tra quelli destinati alla vendita.

```
private void caricaRuoli(List<Footballer> selected) {

    for(Footballer fi: selected) {

        if(fi.getBest_pos().compareTo("P")==0) {
            this.numP++;
        }
        if(fi.getBest_pos().compareTo("D")==0) {
            this.numD++;
        }
        if(fi.getBest_pos().compareTo("C")==0) {
            this.numC++;
        }
        if(fi.getBest_pos().compareTo("A")==0) {
            this.numA++;
        }
    }

}
```

- **Metodo totIndexes():**

Questo metodo, di tipo int, avrà come valore di ritorno il totale di tutti e cinque gli indici per tutti i giocatori selezionati per la vendita.

```
private int totIndexes(List<Footballer> selected) {

    int tot = 0;

    for(Footballer fi: selected) {
        tot += fi.getMarking()+fi.getPassing()+fi.getPositioning()+fi.getStrength()+fi.getTechnique();
    }

    return tot;
}
```

# 5 Diagramma delle classi principali

Successivamente viene riportato il diagramma UML delle classi principali.


(Il file del diagramma è presente all'interno della cartella documenti per potere ingrandire e leggere le scritte più piccole)

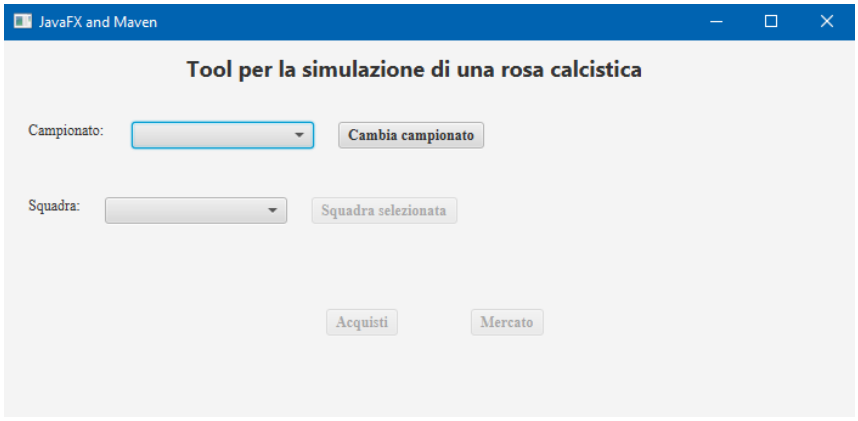
# 6 Videate dell'applicazione e collegamento al video

Il link dimostrativo del funzionamento dell'applicazione si trova presso il seguente link: https://youtu.be/-H3ETWBZ_48

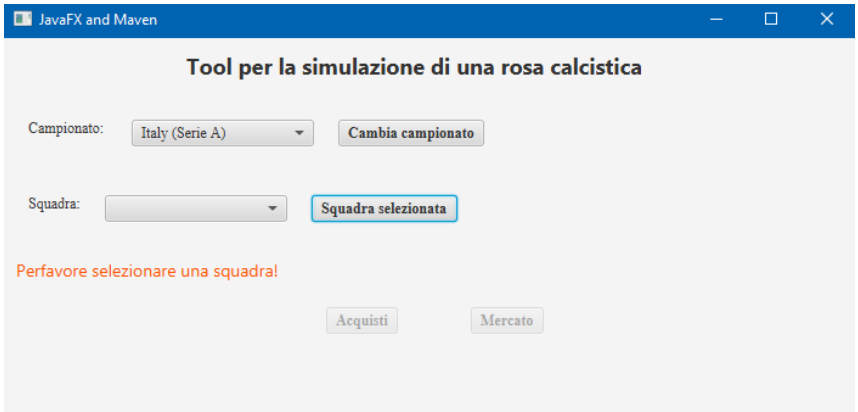
Videate del programma per mostrare il relativo funzionamento:

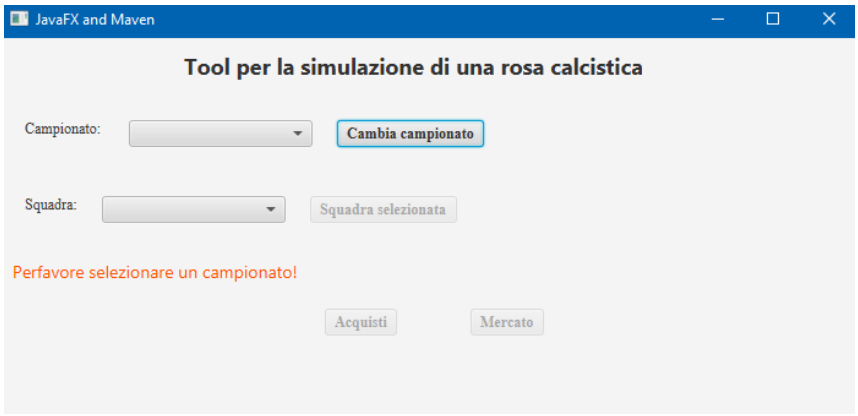
- **Sezione Home**

Vista base:

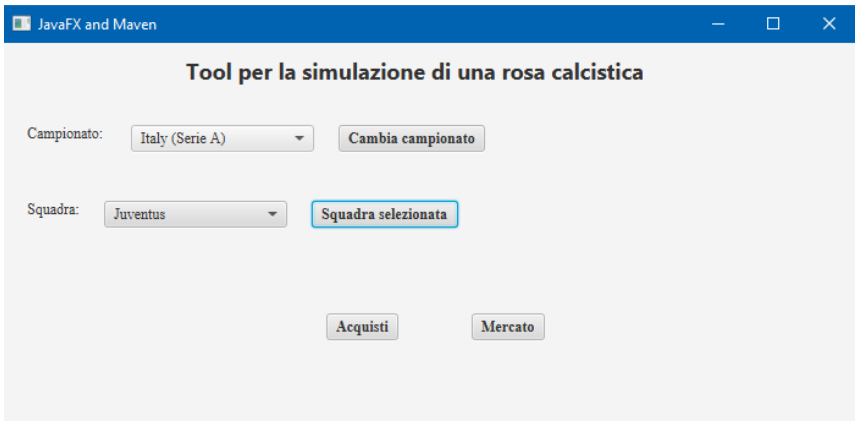



Sono presenti dei controlli per gli errori dell'utente:



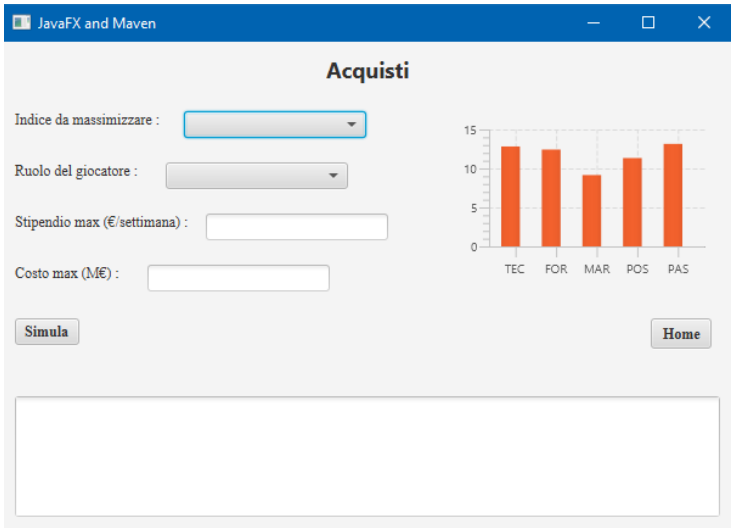



Una volta selezionata la squadra:

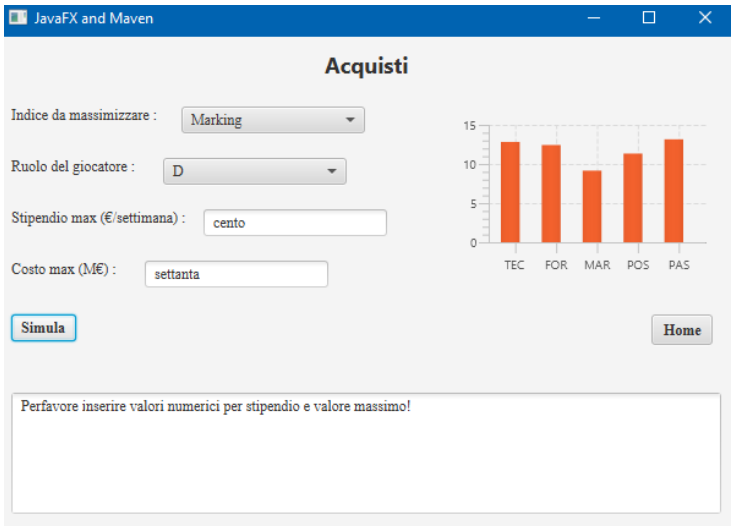


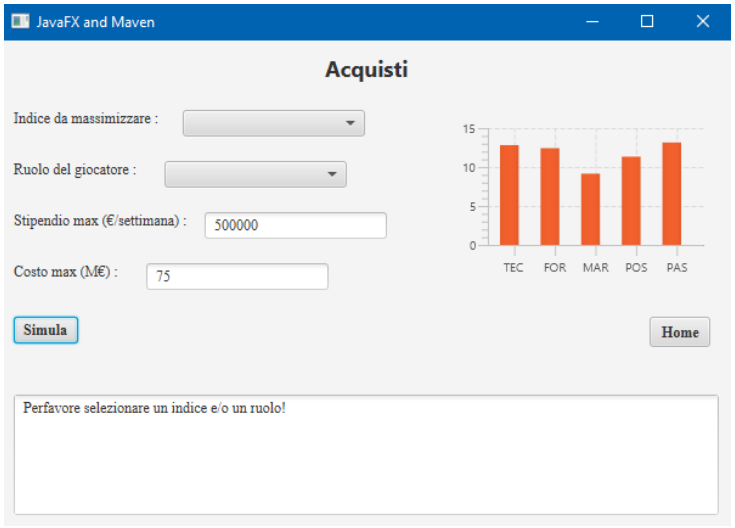

- **Sezione Acquisti**

Vista base:

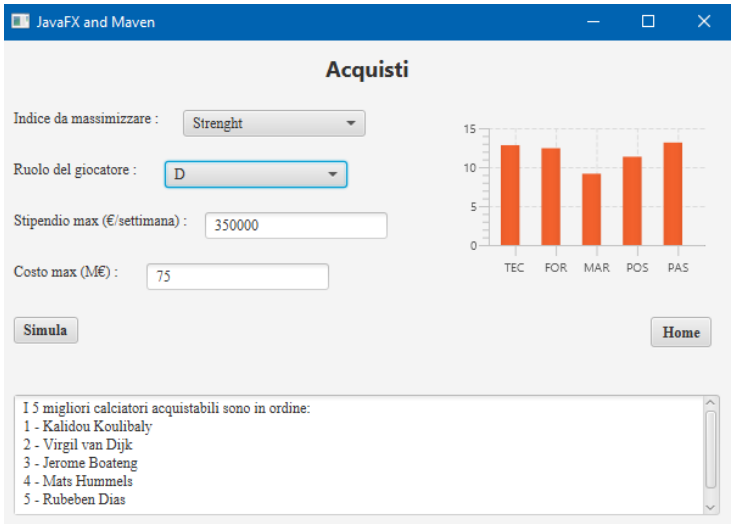



Sono presenti dei controlli per gli errori dell'utente:



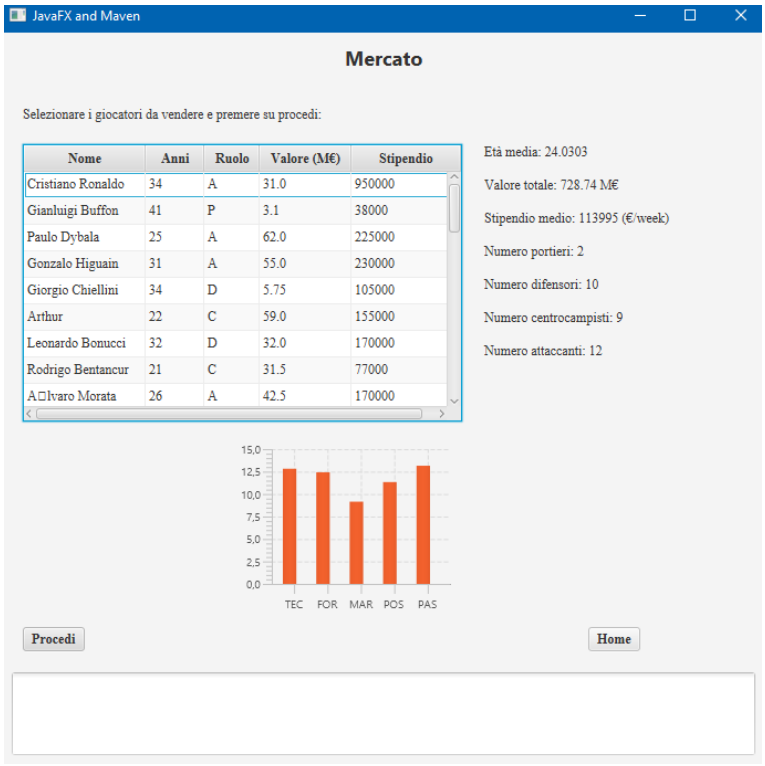




Come vengono mostrati i risultati:

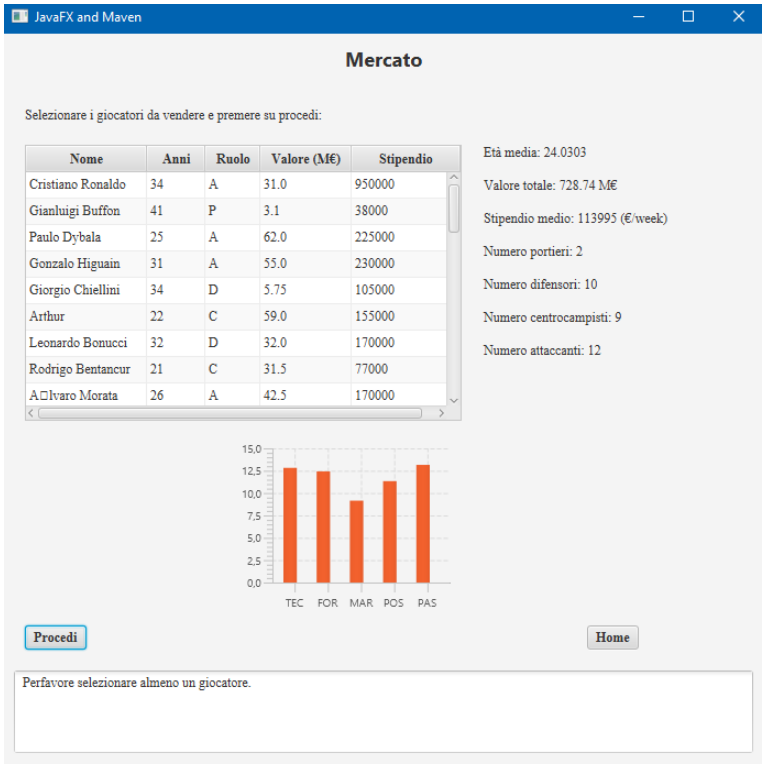


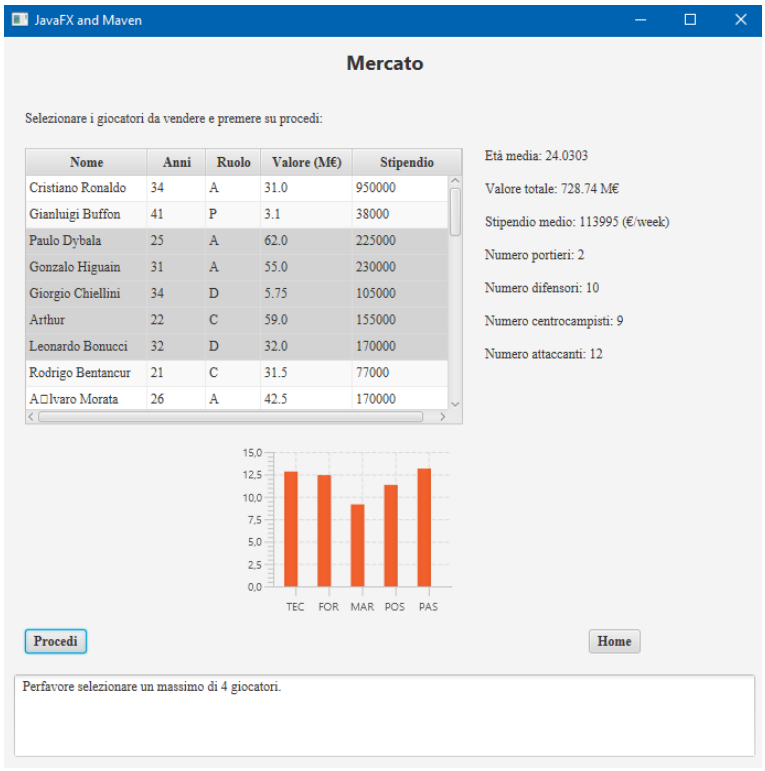

- **Sezione Mercato**

Vista base:

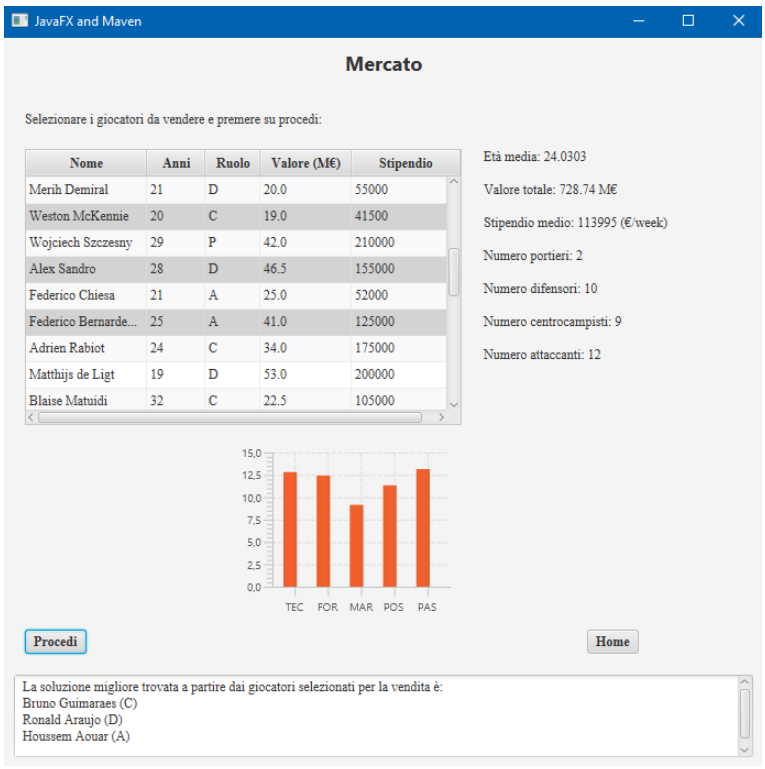



Sono presenti dei controlli per gli errori utente:






Come vengono mostrati i risultati:



# 7 Tabelle con risultati sperimentali

**Sezione acquisti**

| Squadra | Indice | Ruolo | Stipendio max (€) | Valore max (M€) | Risultati trovati |
|---|---|---|---|---|---|
| Juventus | Marking | D | 350000 | 75 | 1 - Kalidou Koulibaly<br>2 - Thiago Silva<br>3 - Raphael Varane<br>4 - Rubeben Dias<br>5 - Milan Skriniar |
| Juventus | Technique | D | 350000 | 75 | 1 - Virgil van Dijk<br>2 - Trent Alexander-Arnold<br>3 - Jerome Boateng<br>4 - David Alaba<br>5 - Thiago Silva |
| Juventus | Passing | D | 350000 | 75 | 1 - Trent Alexander-Arnold<br>2 - Virgil van Dijk<br>3 - Aymeric Laporte<br>4 - Joshua Kimmich<br>5 - Gerard Piquè |
| Juventus | Strenght | D | 350000 | 75 | 1 - Kalidou Koulibaly<br>2 - Virgil van Dijk<br>3 - Jerome Boateng<br>4 - Mats Hummels<br>5 - Rubeben Dias |
| Juventus | Positioning | D | 350000 | 75 | 1 - Mats Hummels<br>2 - Gerard Piquè<br>3 - Virgil van Dijk<br>4 - Thiago Silva<br>5 - Raphael Varane |

**Sezione mercato**

| Squadra | Giocatori selezionati | Giocatori consigliati |
|---|---|---|
| Roma | Gianluca Mancini (D)<br>Nicolò Zaniolo (A) | Ronald Araujo (D)<br>Lucas Paqueta (A) |
| Roma | Henrikh Mkhitaryan (A)<br>Pau Lopez (P) | Zlatan Ibrahimovic (A)<br>Dani Martin (P) |
| Roma | Lorenzo Pellegrini (C)<br>Edin Dzeko (A)<br>Chris Smalling (D) | Eduardo Camavinga (C)<br>Houssem Aouar (A)<br>Ronald Araujo (D) |
| Roma | Javier Pastore (A)<br>Bryan Cristante (C)<br>Amadou Diawara (C)<br>Carles Perez (A) | Houssem Aouar (A)<br>Alvaro Negredo (A)<br>Lucas Leiva (C)<br>Vicente Iborra (C) |
| Roma | Ibanez (D)<br>Bryan Cristante (C)<br>Amadou Diawara (C)<br>Carles Perez (A) | Non sono presenti soluzioni che migliorano gli indici della squadra. |
| Juventus | Gonazlo Higain (A)<br>Rodrigo Bentancur (C) | Mikel Oyarzabal (A)<br>William Carvalho (C) |
| PSG | Mauro Icardi (A)<br>Marquinos (D)<br>Nava (P) | Hamed Junior Traorè (A)<br>Gerard Piquè (D)<br>Jordan Pickford (P) |

# 8 Conclusioni e valutazione dei risultati ottenuti

Come tutte le applicazioni, anche questa presenta diversi punti di forza ma allo stesso tempo dei punti di debolezza.
La prima criticità è legata all'utilizzo dei valori di mercato dei giocatori per simulare i prezzi di acquisto e di vendita, ovviamente utilizzando questo sistema il prezzo del cartellino non sarà del tutto corretto. Quest'ultimo infatti dipende da molti fattori, per esempio: gli anni rimanenti del contratto, la volontà del giocatore, la disponibilità e l'appeal del club acquirente, ma soprattutto dall'ammortamento residuo a bilancio, parametro fondamentale per cercare di effettuare plusvalenze dalla vendita del giocatore.
Un'altra debolezza è legata alla semplificazione effettuata per permettere l'utilizzo dell'applicazione a un utente meno esperto. I ruoli sono stati ridotti dai 13 di partenza fino ai 4 fondamentali: portiere, difensore, centrocampista e attaccante. In questo modo l'algoritmo troverà delle soluzioni più generiche andando a cercare giocatori che agiscono in zone di campo leggermente diverse.
Un'ulteriore debolezza riguarda gli stipendi dei giocatori che sono espressi all'interno del database al lordo; un utente dovrà quindi informarsi sulle tassazioni dei vai paesi per estrapolarne il netto.
Potrebbero essere presenti delle lievi anomalie, queste sarebbe dovute al filtraggio per eliminare i calciatori dei settori giovanili. Sono stati cancellati tutti i calciatori con valore inferiore a 100000€ e con età non superiore a 19 anni. Questo filtraggio ha eliminato quasi tutti i giocatori del settore giovanile a parte alcuni che non rientravano in questi vincoli.
Allo stesso tempo sono presenti notevoli punti di forza; il programma permette di analizzare e cercare soluzioni all'interno di un database comprendente più di 2500 giocatori, una situazione che senza l'ausilio di mezzi informatici sarebbe molto difficile da gestire, se non impossibile.
Sicuramente la vera potenzialità di questo progetto sta nella sezione di mercato dove il programma utilizza un algoritmo ricorsivo andando a cercare combinazioni di migliaia di calciatori per sostituire quelli selezionati per la vendita.
Tra i punti di forza del programma abbiamo sicuramente un'interfaccia user-friendly che permette l'interazione anche con un utente non del tutto esperto del settore e in cui ogni errore viene segnalato all'utente evitando così problemi nell'esecuzione del codice.

Link al video dimostrativo: https://youtu.be/-H3ETWBZ_48

Link al progetto GitHub: https://github.com/TdP-prove-finali/MaturoAlessandro