

Politecnico di Torino

Dipartimento di Ingegneria Gestionale e della Produzione

Corso di laurea in Ingegneria Gestionale

Classe L-8, Ingegneria dell'informazione



Tool per la risoluzione automatica di Sfide Creazione Rosa FIFA

Relatore

Prof. Fulvio Corno

Candidato

Alberto De Benedictis

Sommario

1. <i>PROPOSTA DI PROGETTO</i>	3
2. <i>DESCRIZIONE DETTAGLIATA DEL PROBLEMA</i>	7
3. <i>DESCRIZIONE DEL DATA-SET</i>	10
4. <i>STRUTTURE DATI E ALGORITMI</i>	12
5. <i>DIAGRAMMA DELLE CLASSI PRINCIPALI</i>	16
6. <i>SCREENSHOT DELL'APPLICAZIONE</i>	17
7. <i>VALUTAZIONI SUI RISULTATI</i>	20

1. PROPOSTA DI PROGETTO

Studente proponente

s245313 Alberto De Benedictis

Titolo della proposta

Tool per la risoluzione automatica di Sfide Creazione Rosa FIFA

Descrizione del problema proposto

Nel celebre gioco di calcio per console è presente la modalità di gioco Ultimate Team, in cui ogni utente deve gestire le proprie risorse (crediti) per formare la sua squadra ideale, da utilizzare nelle competizioni amatoriali e professioniste.

Nelle cosiddette "sfide creazione rosa", viene richiesta una rosa di 11 giocatori (ognuno di questi deve essere acquistato sul mercato, se non è già in possesso dell'utente) che soddisfi determinati requisiti, da scambiare per una ricompensa in crediti (o altro).

Trovare la giusta combinazione degli 11 giocatori spesso può diventare una procedura lunga e onerosa, pertanto l'utilizzo di un algoritmo per la soluzione potrebbe giovare a un grande numero di utenti.

Descrizione della rilevanza gestionale del problema

La risoluzione in modo ottimale di un tale tipo di sfide è uno dei metodi più efficaci per ricavare un profitto in crediti (inteso come *RICOMPENSA_SFIDA_COMPLETATA - SPESA_GIOCATORI*) considerevole in poco tempo.

Considerando che EA Sports permette l'acquisto di "FIFA Points" a prezzi poco competitivi e con una resa in crediti molto bassa (*basti pensare che per poter acquistare un solo top player all'inizio del gioco, sarebbe necessaria una spesa di 200-300 €*), l'utilizzo di questa applicazione permetterebbe all'utente medio di accumulare abbastanza crediti per formare una squadra competitiva in poco tempo.

Descrizione dei data-set per la valutazione

<https://www.kaggle.com/stefanoleone992/fifa-20-ultimate-team-players-dataset>

Dataset pubblico, estratto dal sito www.futbin.com, nel quale sono presenti diverse statistiche rilevanti ai fini dell'analisi.

Per ogni giocatore, avremo a disposizione una serie di informazioni, tra cui il campionato e la squadra in cui gioca, la media degli attributi e, infine, il prezzo.

È stato utilizzato anche un dataset di moduli (creato ad hoc), che specifica l'elenco dei ruoli e delle connessioni tra i vari giocatori.

Descrizione preliminare degli algoritmi coinvolti

Per affrontare in modo efficiente il problema, la rosa di 11 giocatori può essere assimilata ad un grafo con 11 vertici. La struttura del grafo varierà in base alla scelta del modulo (input dell'utente). Sebbene i vertici (rappresentati dai giocatori) siano sempre 11, gli archi che collegano i vertici (e la densità del grafo, in generale) sono soggetti a cambiamenti in base al tipo di modulo utilizzato (difesa a 3, utilizzo di una sola punta, etc.).

L'algoritmo ricorsivo principale si occuperà di provare combinazioni di 11 giocatori che soddisfino i vincoli imposti dall'utente.

Il vincolo principale sarà il modulo da utilizzare: la scelta del modulo influenzera in modo decisivo l'esecuzione del programma, poichè a ogni modulo è associato un grafo di tipo diverso.

In particolare, questi potrebbero essere alcuni dei vincoli presenti:

INTESA DI SQUADRA: il valore dell'intesa della squadra deve essere superiore al valore minimo specificato. Questa viene calcolata sommando l'intesa di ogni singolo giocatore, tramite la funzione *calcolaIntesa*;

OVERALL TOTALE DELLA SQUADRA: punteggio medio della squadra, calcolato con una semplice funzione che, data una rosa (grafo), restituisce la media dei punteggi dei singoli giocatori (vertici);

NUMERO DI CAMPIONATI: nella squadra ci devono essere un numero minimo di giocatori provenienti da campionati diversi;

NUMERO DI NAZIONALITÀ: nella squadra ci devono essere un numero minimo di giocatori di nazionalità diverse;

QUALITÀ DEI GIOCATORI: la rosa deve essere composta da giocatori della qualità specificata o maggiore (oro, oro rari, argento);

L'intesa della squadra è un parametro che rappresenta l'affinità tra i giocatori in campo: come accade nel calcio giocato, giocatori che abbiano giocato insieme (nello stesso campionato, nella stessa squadra o magari in nazionale) hanno maggiore affinità.

Il valore dell'intesa varia da 0 a 10 per ogni giocatore, da 0 a 100 per la squadra.

Questo parametro viene modellizzato considerando degli archi pesati: il peso dell'arco in questo caso rappresenta la qualità del collegamento tra due giocatori. Nell'interfaccia del gioco viene espressa con collegamenti di colore diverso (rosso, arancione, verde). Nel programma si potrà fare riferimento a collegamenti di diverso peso.

In particolare:

COLLEGAMENTO ROSSO (peso -1): i due giocatori non hanno attributi comuni

COLLEGAMENTO GIALLO (peso 0): i due giocatori hanno la stessa nazionalità o giocano nello stesso campionato;

COLLEGAMENTO VERDE (peso +1): i due giocatori giocano nello stesso club, oppure giocano nello stesso campionato e sono della stessa nazionalità.

La funzione “calcola Intesa”, quindi, si occuperà di percorrere il grafo e di valutare l'intesa di ogni giocatore(vertice) in base al livello dei collegamenti con i vertici adiacenti (si rimandano ulteriori specifiche a una fase più avanzata del progetto).

Oltre all'algoritmo ricorsivo che ricerca una soluzione, è prevista l'implementazione di un'altra funzione che cerca di ridurre la spesa totale richiesta dalla soluzione trovata.

A livello operativo, la soluzione dovrà percorrere il grafo soluzione, e per ogni vertice cercare un'alternativa equivalente di prezzo minore. (Esempio: sostituire un centrocampista francese che gioca nella Premier League che costa 2000 crediti con un centrocampista della stessa nazione e campionato che però costa 1400).

Descrizione preliminare delle funzionalità previste per l'applicazione software

L'utente inserirà come parametri il modulo della rosa e l'intesa minima della squadra (vincoli obbligatori), più una serie di vincoli opzionali.

Nell'interfaccia sarà presente un pulsante "**Trova Soluzione**" che avvierà l'algoritmo di ricerca della risoluzione ottimale (rosa che soddisfi i vincoli).

Se viene trovata una soluzione soddisfa i vincoli, il programma restituirà la rosa ottimale con la lista dei nomi giocatori, accompagnata dai relativi prezzi; in caso contrario, l'utente verrà invitato a modificare i parametri in input.

Il secondo pulsante "**Riduci il costo**" può essere premuto solo dopo aver trovato una soluzione: questa funzione restituirà (se possibile) una variante della soluzione più economica, ottenuta sostituendo alcuni componenti della rosa.

2. DESCRIZIONE DETTAGLIATA DEL PROBLEMA

A Settembre 2019, in occasione del lancio di “FIFA 20”, ultimo titolo del famigerato videogame prodotto da EA Sports, è stato superato il traguardo dei 10 milioni di giocatori in tutto il mondo.

Da un articolo di **GQ Australia** (link: <https://www.gq.com.au/entertainment/tech/a-new-survey-has-revealed-the-insane-cost-people-are-going-to-to-get-good-at-fifa/news-story/31ea349aede3c3709771bc95a82f29ee>), risalente allo stesso anno, è emerso che più dell’80% degli utenti spende dei soldi per progredire più velocemente nel gioco, ovvero per riuscire a ottenere una squadra competitiva in un arco di tempo minore e ottenere un vantaggio sui propri avversari nelle competizioni online, siano esse amatoriali o a livello agonistico.

Ciò che spinge gli utenti a ricorrere a questo genere di soluzioni è l’effettiva difficoltà con cui si riescono a completare i vari obiettivi all’interno del gioco: al fine di ottenere le risorse necessarie per acquistare i giocatori più ambiti sul mercato, la maggior parte dei teenagers trascorre pomeriggi interi a giocare, dedicando una porzione rilevante del loro tempo libero al gioco su console. È del tutto comprensibile, quindi, che di fronte alla prospettiva di dover attendere settimane per acquistare il proprio giocatore preferito, molte persone preferiscano acquistare i cosiddetti “*FIFA Points*” (crediti virtuali) a delle tariffe tutt’altro che convenienti.

Vi sono, tuttavia, delle alternative ad un tale investimento di tempo o di denaro: tra queste annoveriamo il completamento delle “**Sfide Creazione Rosa**”, che consistono nello scambio di una rosa di 11 giocatori per una ricompensa variabile. Tali sfide rappresentano una considerevole opportunità per ricavare un ingente profitto, se si riesce a gestire sapientemente il proprio club e a individuare i giocatori più convenienti da scambiare. Ciononostante, spesso accade che si impieghi una quantità di tempo sproporzionata per individuare la corretta combinazione di giocatori in base ai vincoli imposti o che, al contrario, la lista di giocatori scelti abbia un costo esageratamente alto, che rende trascurabile la ricompensa ottenuta.

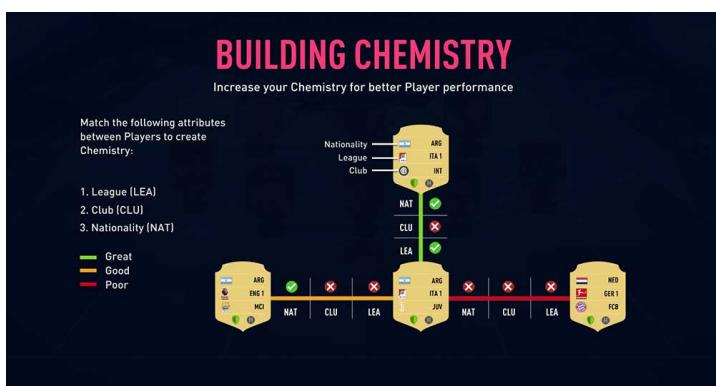
In questo frangente si rivela di fondamentale importanza l’ausilio di un’applicazione che riesca a individuare sistematicamente una lista di giocatori a partire dai vincoli imposti e che, allo stesso

tempo, quantifichi immediatamente l'investimento necessario, così che l'utente possa valutare costi e benefici di questa operazione.

In particolare, l'applicazione sviluppata richiede l'inserimento in input di alcuni parametri, quali il **modulo** da utilizzare, il **numero minimo di campionati** diversi da cui i giocatori possono provenire, la loro “**qualità**”, l'**overall** (ovvero il punteggio) minimo della squadra.



NOME: Kevin De Bruyne
OVERALL: 91
QUALITÀ: “Gold – Rare”



L'ultimo parametro da inserire è l'**intesa** della squadra, che risulta il più rilevante dal punto di vista algoritmico. Essa è il risultato della somma dell'intesa dei singoli giocatori, la quale a sua volta dipende dal peso dei collegamenti di ogni giocatore con i suoi “vicini” di ruolo.

Prendendo in esame l'esempio di **De Bruyne** mostrato nella foto, si può notare che questi ha un collegamento verde (di peso 1, giocando nello stesso club) con **Aguero**, un collegamento arancione con **Kantè** (di peso 0, giocano nello stesso campionato), e un collegamento rosso verso il basso (di peso -1, con un giocatore non belga che non gioca nella Premier League). Da questa combinazione di collegamenti otteniamo che l'intesa di De Bruyne è 10 (evidenziato in figura dalla linea azzurra

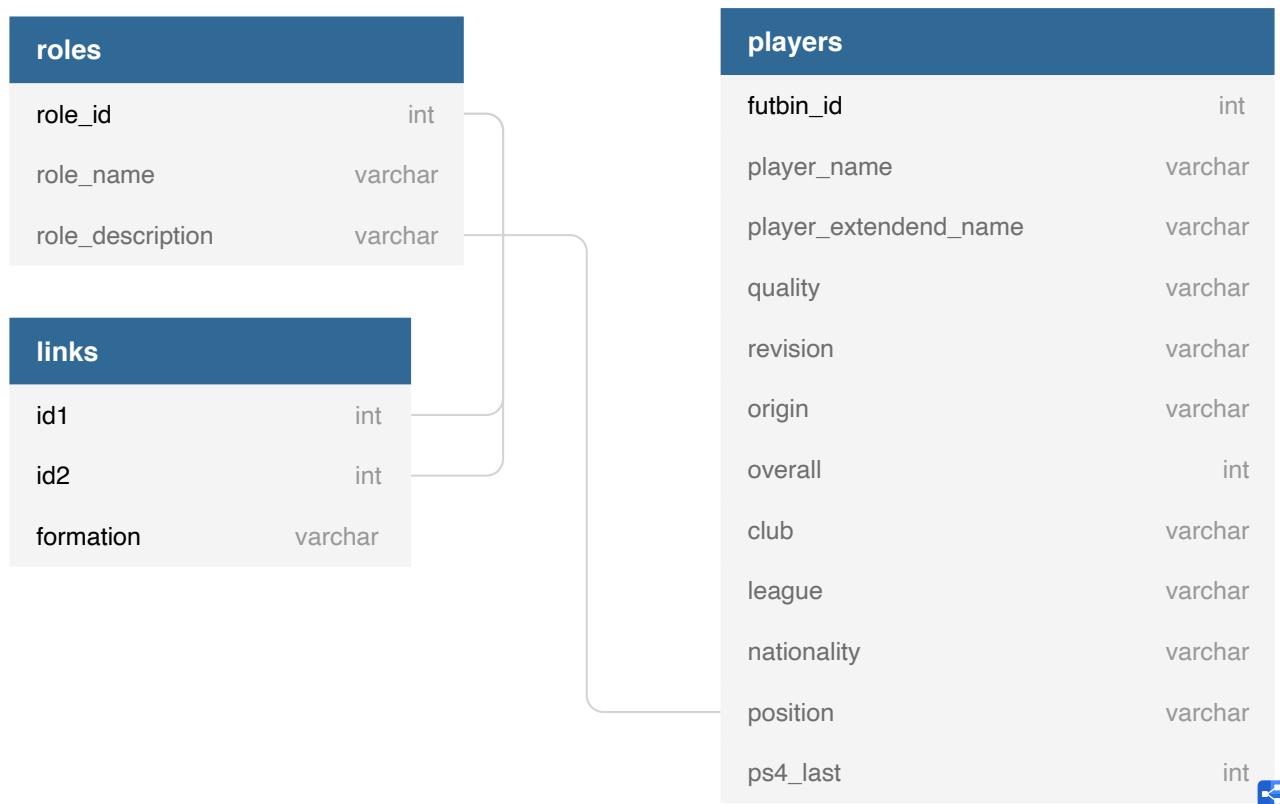
tratteggiata): tale valore andrà sommato a tutti i suoi compagni di squadra, per ottenere l'intesa totale della rosa.

Dal punto di vista degli altri parametri, invece, nella porzione di rosa possiamo individuare giocatori provenienti da un solo campionato (*Premier League*), di qualità “*Gold – Rare*”.

L'output dell'applicazione, pertanto, sarà composto da una lista di giocatori, dei quali verrà specificato nome, ruolo e prezzo, in modo da poterli inserire facilmente in una rosa. A fondo della lista di giocatori, inoltre, verrà restituito il prezzo complessivo, che rappresenta una stima dell'investimento per la sfida in questione. Per una resa efficace dell'applicazione, l'utente deve essere accorto nella valutazione della ricompensa: poichè questa è variabile, bisogna essere realistici e considerare sempre il “worst case”, in modo da assicurarsi che lo scambio dei giocatori sia sempre vantaggioso.

3. DESCRIZIONE DEL DATA-SET

Il data-set utilizzato comprende tre tabelle: **roles**, **links** e **players**.



La tabella **roles** contiene le informazioni riguardanti i ruoli dei giocatori, tra cui:

- **role_id**: un identificativo univoco del ruolo, che indica la sua posizione in campo;
- **role_name**: la sigla del nome del ruolo (Es: **CM** --- > *Central Midfielder*);
- **role_description**: descrizione intesa del ruolo, che facilita la comprensione dell'output.

La tabella **links** contiene tutti i collegamenti presenti tra i ruoli ogni modulo. In particolare:

- **id1**: identificativo del primo ruolo coinvolto nel collegamento;
- **id2**: identificativo del secondo ruolo coinvolto nel collegamento;
- **formation**: modulo nel quale è presente il collegamento tra il ruolo id1 e il ruolo id2

La tabella ***players*** contiene tutti i dati relativi ai giocatori dei 5 campionati principali, quali:

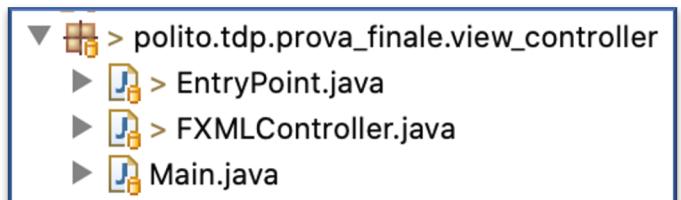
- **futbin_id**: identificativo univoco del giocatore;
- **player_name**: cognome del giocatore (o tipicamente ciò che figura sulla sua maglia);
- **player_extended_name**: nome “esteso” del giocatore;
- **quality**: qualità del giocatore (Gold, Silver, Bronze);
- **revision**: versione del giocatore, che può essere standard o “potenziata”;
- **origin**: se la versione del giocatore è stata rilasciata all’inizio del gioco o meno;
- **overall**: punteggio totale, calcolato come media dei suoi attributi;
- **club**: club in cui gioca il giocatore;
- **league**: campionato a cui appartiene il club;
- **nationality**: nazionalità del giocatore;
- **position**: sigla del ruolo in cui è solito giocare;
- **ps4_last**: ultimo prezzo di mercato (aggiornato a Gennaio 2020), rilevato sulla piattaforma PS4.

4. STRUTTURE DATI E ALGORITMI

L'applicazione è stata scritta in linguaggio Java, implementando i pattern MVC (Model – View – Controller) e DAO (Database Access Object). La struttura dell'applicazione, quindi, si compone di tre package diversi, ognuno delegato allo svolgimento di determinate funzionalità.

In particolare, a livello di package troveremo:

- **view_controller**: package che contiene l'interfaccia grafica (*view*) e il controller, che chiama i metodi del model per ricavare l'output da stampare.
- **db**: package che contiene le classi che si occupano della gestione del database. La classe *DBConnect* genera la connessione al localhost, mentre *PlayersDAO* contiene i metodi per ottenere i dati dal database;
- **model**: package che contiene la logica e i dati. Nella classe *Model* troviamo tutti i metodi indispensabili al funzionamento dell'applicazione, mentre le altre classi sono il supporto principale per la formazione delle strutture dati.



Mentre il **view_controller** e il **db** contengono dei metodi piuttosto elementari, il cui unico scopo è quello di fornire i dati al **Model** o di mostrare correttamente i risultati dei calcoli, è nel **model** che vengono adoperati diversi algoritmi per fornire il risultato corretto all'utente. Nella pagina seguente verranno mostrati i passaggi salienti dei tre algoritmi principali.

Come specificato nella proposta, l'applicazione svolge due funzioni: la prima è l'individuazione di una rosa che soddisfi i vincoli imposti, la seconda è la valutazione dell'esistenza di una soluzione alternativa più economica.

Per individuare la rosa viene utilizzato il metodo ***creaGrafo***, che a sua volta chiama il metodo ***ricorsione***.

```

56⊕  public List<TeamPlayer> creaGrafo(String formation, Integer intesa, Integer minLeagues, Integer nations,
57      String quality, Integer overall) {
58
59      this.min_GiocatoriPerRuolo = 3;
60
61      // carico la mappa dei ruoli
62      ruoliIdMap = new HashMap<>();
63      List<Ruolo> ruoli = this.dao.getRuoli();
64      for (Ruolo r : ruoli) {
65          ruoliIdMap.put(r.getId(), r);
66      }
67
68      this.grafo = new SimpleWeightedGraph<>(DefaultWeightedEdge.class);
69
70      // In base alla formazione scelta dall'utente, creo un GRAFO DI RIFERIMENTO:
71      // questo sarà formato da degli oggetti TeamPlayer con i ruoli definiti, ma i
72      // player vuoti, ovvero messi a null
73
74      // AGGIUNGONO VERTICI E ARCHI
75      List<CoppiaRuoli> coppie = this.dao.getCopie(formation);
76
77      for (CoppiaRuoli c : coppie) {
78
79          // prendo i ruoli della coppia
80          Ruolo r1 = ruoliIdMap.get(c.getId1());
81          Ruolo r2 = ruoliIdMap.get(c.getId2());
82
83          TeamPlayer p1 = new TeamPlayer(null, r1);
84          TeamPlayer p2 = new TeamPlayer(null, r2);
85
86          if (!this.grafo.containsVertex(p1)) {
87              grafo.addVertex(p1);
88          }
89          if (!this.grafo.containsVertex(p2)) {
90              grafo.addVertex(p2);
91          }
92          // inizialmente metto un peso fittizio, che aggiornerò a squadra terminata
93          if (!this.grafo.containsEdge(p1, p2)) {
94              Graphs.addEdge(this.grafo, p1, p2, 10);
95          }
96      }
}

```

Figura 1: metodo ***creaGrafo*** - 1° parte

- 1) Il metodo ***creaGrafo*** riceve i parametri immessi dall'utente nell'interfaccia grafica grazie al controller;
- 2) Dopo aver creato un idMap che colleghi id del ruolo all'oggetto ruolo corrispondente, si procede alla creazione del grafo;
- 3) In base al modulo scelto dall'utente (*String formation*), si richiedono al DAO tutti i collegamenti presenti: iterando sulla lista di collegamenti, si aggiungono di volta in volta vertici e archi al grafo, impostando un peso iniziale fittizio.

```

98     // Preparo la lista per la ricorsione
99     vertici = new ArrayList<>(this.grafo.vertexSet());
100    Collections.sort(vertici);
101
102    // Preparo una mappa contenente i giocatori che posso mettere
103    // in ogni ruolo in base ai vincoli dell'utente
104    mappaRuoli = new TreeMap<String, List<Player>>();
105
106    for (TeamPlayer p : vertici) {
107
108        String nomeRuolo = p.getRuolo().getName();
109
110        // se non ho già preso i giocatori (esempio CB, che compare almeno due volte)
111        if (!mappaRuoli.containsKey(nomeRuolo)) {
112
113            List<Player> giocatoriRuolo = new ArrayList<>();
114
115            giocatoriRuolo = this.dao.getPlayersByParameters(overall, quality, nomeRuolo);
116
117            // PROBLEMI GIOCATORI NON TROVATI
118            // cerco giocatori con overall crescente fin quando non ne trovo almeno 2 o 3
119            int incr_overall = overall;
120            String incr_quality = quality;
121            while (giocatoriRuolo.size() < this.min_GiocatoriPerRuolo && incr_overall <= 93) {
122
123                // Accordiamo qualità e overall per trovare i risultati
124                if (incr_overall == 64) {
125                    incr_quality = quality.replace("Bronze", "Silver");
126                } else if (incr_overall == 74) {
127                    incr_quality = quality.replace("Silver", "Gold");
128                } else if (incr_overall > 82) {
129                    // l'overall dei giocatori non-rari si ferma a 82
130                    incr_quality = quality.replace("Non-Rare", "Rare");
131                }
132                List<Player> giocatoriDaAggiungere = new ArrayList<>();
133
134                giocatoriDaAggiungere = this.dao.getPlayersByParameters(incr_overall, incr_quality, nomeRuolo);
135                incr_overall++;
136
137                if (giocatoriDaAggiungere.size() > 0) {
138                    for (Player daAgg : giocatoriDaAggiungere) {
139                        if (!giocatoriRuolo.contains(daAgg)) {
140                            giocatoriRuolo.add(daAgg);
141                        }
142                    }
143                }
144            }
145            // fin quando non ho almeno 2 o 3 giocatori per ruolo, cerco quelli con overall
146            // maggiore
147            if (giocatoriRuolo.size() < this.min_GiocatoriPerRuolo) {
148                System.out.println("Non è stato possibile trovare abbastanza giocatori nel ruolo " + nomeRuolo);
149                return null;
150            }
151            // riempio la mappa che segna le corrispondenze tra ruolo ed elenco giocatori
152            // sarà questo l'elenco di giocatori su cui farò la ricorsione
153            mappaRuoli.put(nomeRuolo, giocatoriRuolo);
154            System.out.println("Numero giocatori per il ruolo " + nomeRuolo + ": " + giocatoriRuolo.size());
155        }
156    }
157}

```

Figura 2: metodo *creaGrafo* - 2° parte

- 4) Dopo aver creato il grafo “di riferimento”, che contiene un insieme di vertici con giocatori NULL, si interroga il DAO per ottenere le liste di giocatori che andranno a comporre la squadra desiderata.
- 5) Per questo scopo si rivela efficace l’utilizzo di una map, le cui chiavi sono le sigle dei ruoli, mentre i valori sono le liste di giocatori che giocano nel ruolo definito nella chiave.
- 6) Il processo di reperimento dei giocatori si arresta nel momento in cui tra i valori della mappa abbiamo solo liste non vuote con 2 o più giocatori da poter provare.

Adesso è possibile inizializzare i valori per la ricorsione e chiamare la funzione ricorsiva.

```

187    private void ricorsione(List<TeamPlayer> parziale, int livello, List<Player> squadra, Integer intesa,
188                           Integer minLeagues, Integer nations, Integer overall) {
189
190        // CASO TERMINALE
191        if (trovata) {
192            return;
193        } else {
194            if (livello == 11) {
195                // DEBUG
196                // trovate++;
197                // System.out.println(trovate);
198
199                // CONTROLLO NUM CAMPIONATI
200                if (numeroCampionati(parziale) >= minLeagues) {
201
202                    // CONTROLLO OVERALL
203                    if (overallSquadra(squadra) >= overall) {
204
205                        // carico giocatori e pesi nel grafo
206                        caricaGiocatori(parziale);
207                        caricaPesi();
208
209                        // CONTROLLO INTESA
210                        if (getIntesaSquadra() >= intesa) {
211
212                            // this.numCampionatiSol = numeroCampionati(parziale);
213                            // this.overallSol = overallSquadra(squadra);
214                            // this.intesaSol = getIntesaSquadra();
215
216                            this.soluzione = new ArrayList<>(parziale);
217                            this.squadraSol = new ArrayList<>(squadra);
218                            this.prezzoSol = calcolaPrezzo(parziale);
219                            trovata = true;
220
221                            System.out.println("Controllo campionati superato: " + numeroCampionati(parziale));
222                            System.out.println("Controllo overall superato: " + overallSquadra(squadra));
223                            System.out.println("Controllo intesa della squadra superato: " + getIntesaSquadra());
224
225                        }
226                    }
227                }
228            }
229            resetGiocatoriPesi();
230        }
231    }
232    return;
233}
234
235 // CREAZIONE NUOVE SOLUZIONI
236 for (Player p : this.mappaRuoli.get(vertici.get(livello).getRuolo().getName())) {
237
238    if (!squadra.contains(p)) {
239
240        TeamPlayer tp = new TeamPlayer(p, vertici.get(livello).getRuolo());
241        parziale.add(tp);
242        squadra.add(p);
243        ricorsione(parziale, livello + 1, squadra, intesa, minLeagues, nations, overall);
244
245        // backtracking
246        parziale.remove(tp);
247        squadra.remove(p);
248
249    }
250}
251}
252}

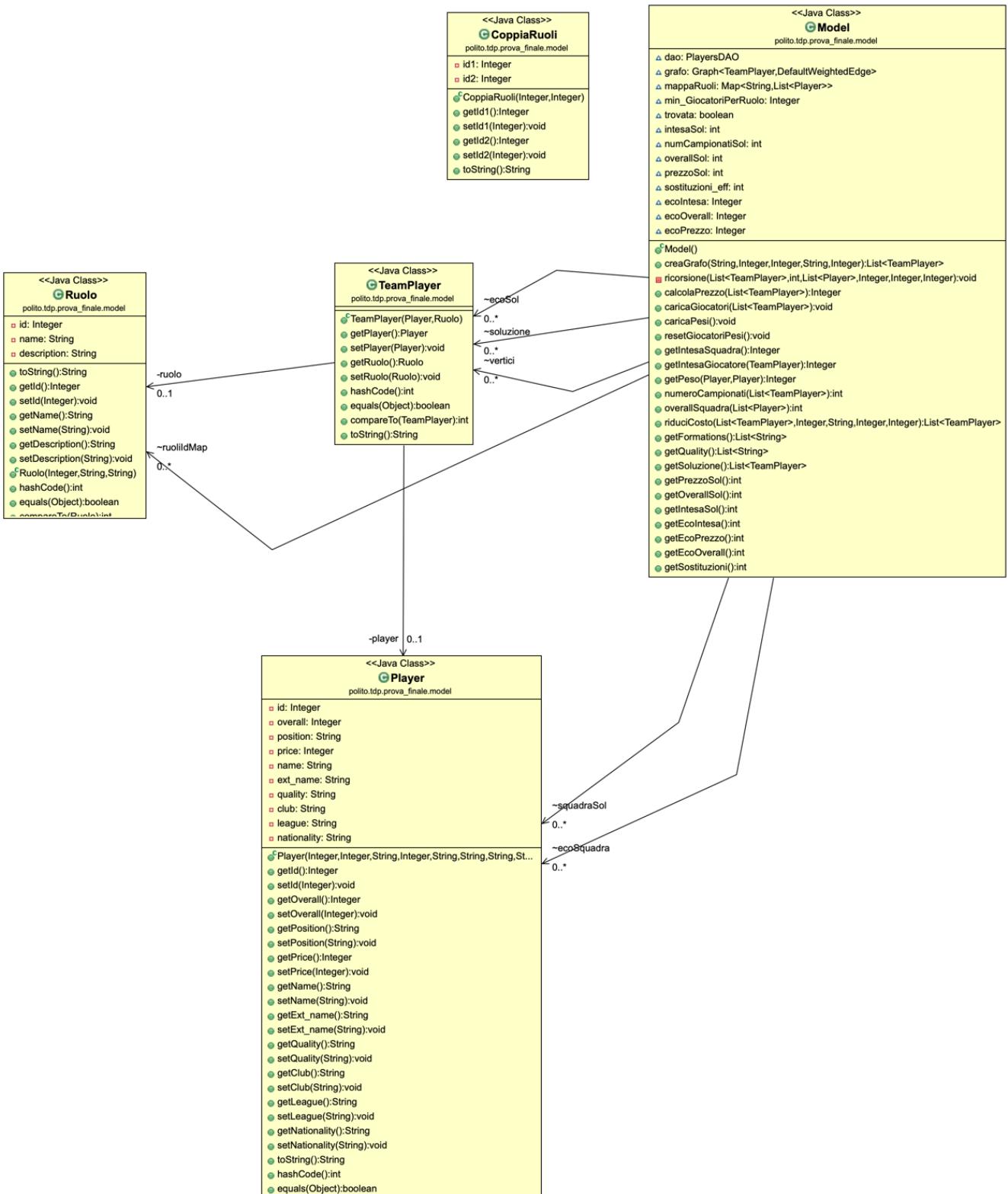
```

La funzione ricorsiva si sviluppa in due blocchi di codice:

- **CASO TERMINALE:** se si è giunti alla formazione di una squadra da 11 giocatori, se ne controllano le caratteristiche (caricando i giocatori nel grafo e valutando i collegamenti). In caso positivo, la soluzione viene salvata e la ricorsione viene terminata, altrimenti si procede;
- **CREAZIONE NUOVE SOLUZIONI:** scorrendo i valori della mappa che contiene le liste di ruoli, viene aggiunto ricorsivamente un giocatore alla lista, esplorando il ramo di soluzioni che ne scaturiscono e facendo backtracking.

La funzione **riduci Costo**, infine, ripete per certi versi il processo appena visto, iterando sulla lista di giocatori già formata e cercando di sostituire il giocatore corrente con un giocatore più economico.

5. DIAGRAMMA DELLE CLASSI PRINCIPALI



6. Screenshot dell'applicazione

Link al video dimostrativo dell'applicazione: <https://youtu.be/fsktwU4-mu4>

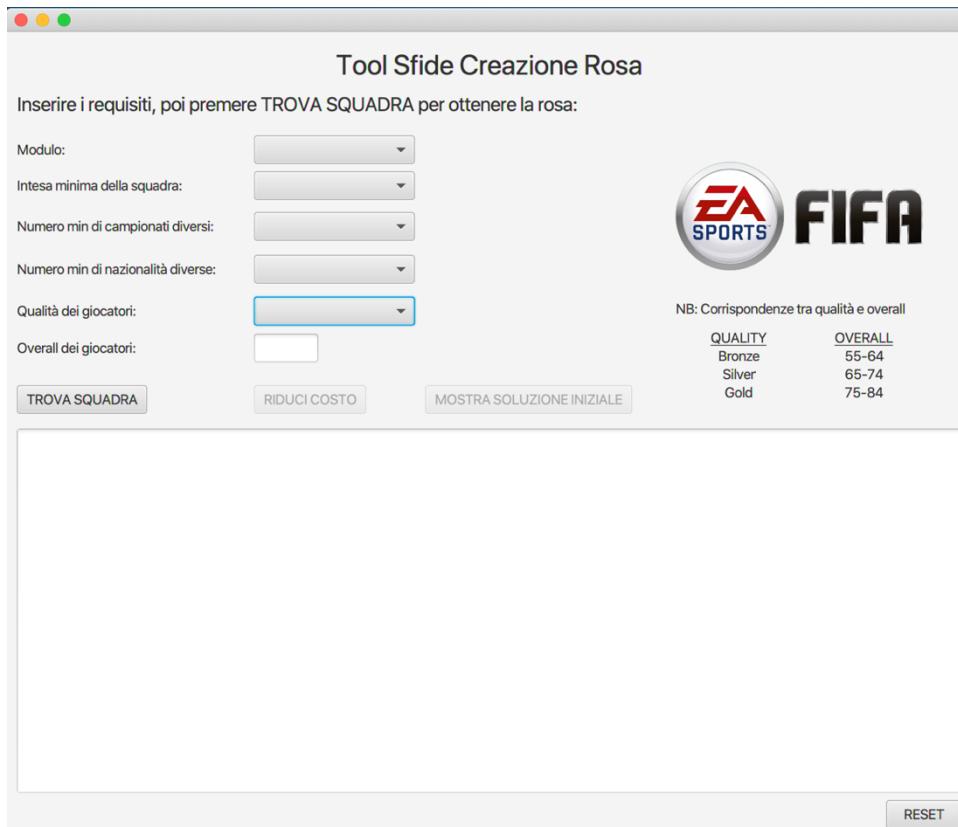


Figura 3: Schermata iniziale dell'applicazione

The screenshot shows the application after finding a team. The interface is identical to Figure 3, but the large central area now displays the results of the search. It starts with a message "SQUADRA TROVATA!" followed by "(modulo: 4-4-2, qualità: Gold - Rare, min campionati: 3, overall minimo: 77, min nazionalità: 6)". Below this, it shows "Totale: 77" and "Intesa: 77". The main table lists the players with their descriptions, roles, overall ratings, clubs, and prices. The table has columns for "DESCRIZIONE", "RUOLO", "OV NOME", "CLUB", and "PREZZO". The table shows 11 players: goalkeeper (GK), left full back (CB), right full back (CB), left back (LB), right back (RB), left central midfielder (CM), right central midfielder (CM), left midfielder (LM), right midfielder (RM), left striker (ST), and right striker (ST). The total cost is 8050 credits. At the bottom right is a "RESET" button.

DESCRIZIONE	RUOLO	OV NOME	CLUB	PREZZO
goalkeeper	GK	77 MENDY	Stade Rennais FC	700
left full back	CB	77 COLLEY	Sampdoria	750
right full back	CB	77 COADY	Wolverhampton Wanderers	800
left back	LB	77 MAX	FC Augsburg	750
right back	RB	77 WEISER	Bayer 04 Leverkusen	850
left central midfielder	CM	77 KUCKA	Parma	700
right central midfielder	CM	77 SCHØNE	Genoa	700
left midfielder	LM	77 YOUNES	Napoli	700
right midfielder	RM	77 MARUŠIĆ	Lazio	700
left striker	ST	77 PETAGNA	SPAL	700
right striker	ST	77 LASAGNA	Udinese	700

Figura 4: Output "squadra trovata"

Tool Sfide Creazione Rosa

Inserire i requisiti, poi premere TROVA SQUADRA per ottenere la rosa:

Modulo:	4-4-2
Intesa minima della squadra:	75
Numero min di campionati diversi:	3
Numero min di nazionalità diverse:	6
Qualità dei giocatori:	Gold - Rare
Overall dei giocatori:	77

FIFA

NB: Corrispondenze tra qualità e overall

QUALITY	OVERALL
Bronze	55-64
Silver	65-74
Gold	75-84

E' stata trovata una soluzione più economica, effettuando le seguenti sostituzioni:
 - BOYATA al posto di COADY nel ruolo CB, risparmiando 50 crediti
 - ALEX VIDAL al posto di WEISER nel ruolo RB, risparmiando 100 crediti

Totale: 77
 Intesa: 77
 Risparmio totale: 150 crediti

DESCRIZIONE	RUOLO	OV NOME	CLUB	PREZZO
goalkeeper	GK	77 MENDY	Stade Rennais FC	700
left full back	CB	77 COLLEY	Sampdoria	750
right full back	CB	76 BOYATA	Hertha BSC	750
left back	LB	77 MAX	FC Augsburg	750
right back	RB	76 ALEX VIDAL	D. Alavés	750
left central midfielder	CM	77 KUCKA	Parma	700
right central midfielder	CM	77 SCHØNE	Genoa	700
left midfielder	LM	77 YOUNES	Napoli	700
right midfielder	RM	77 MARUŠIĆ	Lazio	700
left striker	ST	77 PETAGNA	SPAL	700
right striker	ST	77 LASAGNA	Udinese	700

Figura 5: Output "soluzione economica"

Soluzione mostrata su console:



Figura 6: verifica della validità della soluzione

Tool Sfide Creazione Rosa

Inserire i requisiti, poi premere TROVA SQUADRA per ottenere la rosa:

Modulo:	3-4-3
Intesa minima della squadra:	80
Numero min di campionati diversi:	4
Numero min di nazionalità diverse:	8
Qualità dei giocatori:	Silver - Non-Rare
Overall dei giocatori:	71

NB: Corrispondenze tra qualità e overall

QUALITY	OVERALL
Bronze	55-64
Silver	65-74
Gold	75-84

TROVA SQUADRA
RIDUCI COSTO
MOSTRA SOLUZIONE INIZIALE

SQUADRA TROVATA!
(modulo: 3-4-3, qualità: Silver - Non-Rare, min campionati: 4, overall minimo: 71, min nazionalità: 8)
Totale: 71
Intesa: 80

DESCRIZIONE	RUOLO	OV NOME	CLUB	PREZZO
goalkeeper	GK	71 GOICOECHEA	Toulouse Football Club	500
left full back	CB	71 LEFORT	Amiens SC	700
right full back	CB	71 SALIBA	AS Saint-Étienne	700
central full back	CB	71 DANSO	Southampton	800
left central midfielder	CM	71 FARAGÒ	Cagliari	600
right central midfielder	CM	71 ANGBAN	Football Club de Metz	700
left midfielder	LM	71 PHILIPPOTEAUX	Nîmes Olympique	1400
right midfielder	RM	71 PAULINHO	Bayer 04 Leverkusen	1100
left wing	LW	72 NGUETTE	Football Club de Metz	600
right wing	RW	72 JONATHAN CAFÚ	FC Girondins de Bordeaux	500
central striker	ST	71 GANAGO	OGC Nice	550

Prezzo: 8150 crediti

RESET

Figura 7: output "squadra trovata #2"



Figura 8: verifica validità della soluzione #2

7. VALUTAZIONI SUI RISULTATI

L'applicazione sviluppata raggiunge il suo scopo principale: nella maggior parte dei casi, è in grado di fornire una soluzione al problema proposto, in un arco di tempo decisamente ragionevole.

Il prezzo indicato per ciascuna squadra costituisce una stima, poichè il database non è aggiornato in tempo reale, bensì fornisce un'istantanea dei prezzi risalente a Gennaio 2020.

Vi sono, tuttavia, dei casi limite, per cui risulta possibile il raggiungimento di un risultato definitivo: se si impongono dei vincoli molto stringenti, come ad esempio un alto numero di nazionalità (> 7) e un alto numero di campionati (< 3), è molto difficile che si riesca a trovare una squadra, se il requisito di intesa è superiore a 50-60. Ciò accade perché diversificando campionati e nazionalità in maniera incrementale, si aumenta allo stesso modo la probabilità che i collegamenti tra le varie coppie di giocatori sia ROSSO, ovvero di peso -1. Questo genere di situazioni tende a verificarsi con moduli rappresentati da grafi più densi, ovvero con un maggior numero di collegamenti tra i giocatori.

La principale criticità di questi casi risiede nel fatto che vi siano pochissime (se esistenti) combinazioni di giocatori che soddisfino questi vincoli: nel caso in cui ve ne fossero, tuttavia, bisognerebbe sempre considerare che la limitata capacità di elaborazione del proprio computer pone un limite ai risultati raggiungibili. Nel caso del sottoscritto, è stato verificato sperimentalmente che in alcuni minuti il computer riesce a “verificare” dai 10 ai 15 milioni di combinazioni di 11 giocatori: per avere un’idea delle effettive combinazioni possibili, basta considerare una media di 5 giocatori per ruolo; in questo caso, si avrebbero $5^{11} = 48.000.000$ di combinazioni da provare prima di avere la certezza sull’effettiva esistenza della soluzione.

La funzione riduci costo talvolta non riesce a trovare una soluzione più economica, perché i giocatori utilizzati (es: *giocatori Gold-Rare*) hanno già il prezzo minimo al quale li si può vendere, corrispondente a 700 crediti, quindi non possiamo trovare alternative equivalenti a 650 o meno.

In conclusione, come si evince dal video dimostrativo, l'applicazione fornisce un valido aiuto nelle situazioni ordinarie, in cui i vincoli sono perfettamente gestibili dal modello, permettendo agli utenti di risparmiare tempo per il calcolo della soluzione e, diverse volte, di ottenere anche una soluzione molto economica, in modo da poter trarre profitto dalla ricompensa ricevuta.



Quest'opera è distribuita con Licenza [Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale](#)