

Politecnico di Torino

Corso di Laurea in Ingegneria Gestionale

Classe L8 – Ingegneria dell'Informazione

A.A. 2020/2021

Sessione di Laurea Ottobre 2021



Politecnico di Torino

Applicazione per la creazione squadra Ultimate team (FIFA)

Relatore:
Prof. Fulvio Corno

Candidato:
Davide Borgatta
249465

INDICE

1.PROPOSTA DI PROGETTO _____	4
2.DESCRIZIONE DETTAGLIATA DEL PROBLEMA _____	7
3.DESCRIZIONE DEL DATA-SET_____	8
4.DESCRIZIONE DELLE STRUTTURE DATI E DEGLI ALGORITMI UTILIZZATI_____	10
5.DIAGRAMMA DELLE CLASSI PRINCIPALI_____	15
6.ESEMPI VIDEO DELL'APPLICAZIONE_____	16
7.VALUTAZIONI SUI RISULTATI E CONCLUSIONE FINALE_____	18

1.PROPOSTA DI PROGETTO

Studente proponente

s249465 Borgatta Davide

Titolo della proposta

Applicazione per la creazione squadra Ultimate Team (FIFA)

Descrizione del problema proposto

In FIFA, gioco di calcio per console, è usata da molti utenti la modalità Ultimate Team, in cui ogni utente deve creare la propria squadra ideale tramite i crediti, risorse virtuali a disposizione di ogni giocatore. Questi vengono accumulati partecipando a varie competizioni ed è quindi fondamentale avere una squadra competitiva per poter ambire a vincere più crediti possibili e migliorare sempre la propria formazione. La creazione della squadra è sempre molto difficoltosa sia per i vincoli di budget sia per quelli di intesa tra i singoli giocatori. L'applicazione si pone l'obiettivo di aiutare l'utente nella creazione della miglior squadra possibile partendo da quattro giocatori scelti dall'utilizzatore e costruendo intorno a loro la formazione migliore possibile rispettando i vincoli imposti. Inoltre, è importante che la squadra abbia un valore di intesa alto ed è quindi consigliato avere ogni giocatore con un minimo di intesa individuale pari a 7.

Descrizione della rilevanza gestionale del problema

Nella modalità Ultimate Team esiste una competizione chiamata FUT Champions dove utenti di tutto il mondo si sfidano in 30 partite on-line e si vincono premi in base al numero di vittorie conseguite. In questa modalità in cui si sfidano altri utenti è fondamentale avere una squadra competitiva e utilizzare al meglio i propri crediti per acquistare i giocatori giusti ed avere dei ricavi in termini di crediti. Di solito ogni utente ha in mente dei giocatori ideali senza i quali non può far a meno e perde molto tempo nel riuscire a trovare i restanti calciatori in modo da avere un valore di intesa alto e avere allo stesso tempo giocatori forti per cercare di vincere più partite possibili. Quindi il problema è di rilevanza manageriale in quanto con le risorse disponibili (crediti) bisogna riuscire a costruire la migliore squadra possibile, inoltre è fondamentale compiere questa operazione in un lasso di tempo ottimale.

Descrizione dei data-set per la valutazione

I dati relativi ai giocatori disponibili in FUT (Fifa ultimate team) derivano dal data-set al link:

https://www.kaggle.com/stefanoleone992/fifa-21-ultimate-team-players-and-prices-dataset?select=fut_bin21_players.csv

In questo data-set verranno considerati solo i giocatori oro in quanto unici ad esser utilizzati nel gameplay, inoltre verranno estratti gli attributi che serviranno alla risoluzione del problema:

id: identificativo univoco di ogni giocatore su FUT

nome: il nome, in gioco, del calciatore

rarity: attributo che può avere due valori: raro o non raro

overall: valore del giocatore

club: squadra dove il calciatore gioca

lega: campionato dove il calciatore gioca

nazionalità: nazionalità del calciatore

posizione: ruolo in campo del calciatore

prezzo: prezzo del calciatore in crediti

Inoltre, è stato creato un database di ruoli, questi sono utilizzati nelle varie formazioni usate per schierare gli undici giocatori ed in base al modulo sono stati creati i relativi collegamenti tra i vari ruoli.

Descrizione preliminare degli algoritmi coinvolti

Il programma riceve dall'utente: il modulo di 11 calciatori da usare, i 4 giocatori scelti, il budget massimo da utilizzare e l'opzione scelta sulla qualità dei giocatori.

A questo punto verrà creato un grafo pesato e non orientato con 11 vertici (gli 11 giocatori) e archi che varieranno in base al modulo selezionato. Come primo passo verranno inseriti i 4 giocatori nelle rispettive posizioni e un controllo andrà a verificare che i giocatori inseriti rispettino i vincoli, se non ci saranno problemi allora inizierà un algoritmo ricorsivo che andrà a valutare le combinazioni tra gli 11 giocatori e restituirà la migliore squadra rispettando i vincoli. Una funzione calcolerà l'overall della squadra facendo la media tra l'overall degli 11 calciatori mentre un'altra funzione andrà a calcolare l'intesa della squadra facendo la somma tra l'intesa individuale degli undici giocatori (valore che può arrivare al massimo a 100). Questi 2 valori calcolati verranno sommati per valutare la squadra con il punteggio più alto (la migliore).

I vincoli da rispettare sono: non superare il budget inserito dall'utente, non sostituire i 4 giocatori selezionati dall'utente e intesa di ogni giocatore almeno a 7.

L'intesa è un parametro che va a valutare le affinità tra i vari calciatori in rosa e varia da 0 a 10, un parametro molto importante in quanto con un valore inferiore a 7 le prestazioni del calciatore incomincerebbero a risentirne.

L'affinità del giocatore viene calcolata in base al peso dei suoi archi ed ognuno di esso nel grafo avrà un peso specifico in base a valutazioni di intesa dei calciatori valutata sui tre parametri: club, lega, nazionalità.

Tripla intesa, peso 2, giocatori che hanno in comune tutti e tre gli attributi.

Doppia intesa, peso 1, giocatori che hanno in comune due attributi mentre il terzo è differente.

Singola intesa, peso 0, giocatori che hanno in comune un unico attributo.

Nessun'intesa, peso -1, giocatori che hanno tutti e tre gli attributi differenti.

Descrizione preliminare delle funzionalità previste per l'applicazione software

L'utente come prima cosa andrà ad inserire il modulo della sua squadra ideale (selezionato tra varie opzioni), inserendo anche quattro giocatori da lui desiderati e il budget a sua disposizione per formare la squadra. A questo punto cliccando sul bottone verifica il programma andrà a controllare se vengono rispettati i vincoli di posizione (ad esempio non si possono scegliere due portieri inseriti dall'utente) e di budget. Se questi vincoli sono rispettati allora verrà data conferma positiva con messaggio a video, inoltre verrà anche visualizzato il budget restante per gli altri sette giocatori. L'utente a questo punto potrà con un altro bottone iniziare la ricorsione. Se tale procedura è riuscita a trovare la squadra migliore allora verranno mostrati a video gli 11 giocatori, il valore complessivo di overall e l'intesa. Se invece la ricorsione ha avuto esito negativo il programma avviserà l'utente che non ha abbastanza crediti per costruire la squadra e lo inviterà ad aumentare il budget o a rinunciare a uno dei suoi quattro giocatori desiderati per far posto a un calciatore che costi meno.

2.DESCRIZIONE DETTAGLIATA DEL PROBLEMA

I videogiochi negli ultimi anni hanno mutato il loro obiettivo, nati come un passatempo ludico ora per molti sono diventati un vero e proprio lavoro. I giocatori più bravi vengono pagati per partecipare a competizioni sia nazionali che mondiali, anche FIFA ha seguito questa tendenza, ogni società sportiva è rappresentata da un videogiocatore che è chiamato a far pubblicità e portare prestigio alla squadra in cui è stato ingaggiato. Sempre più persone vorrebbero intraprendere la carriera di ‘Pro-Player’ e far della propria passione un lavoro, per fare ciò il miglior modo per farsi notare da Aziende o Società Sportive che investono nel gaming online è senza dubbio partecipare alla FUT Champions. Tale competizione è aperta a qualunque giocatore di FIFA e si svolge ogni fine settimana, durante il quale ogni utente sfiderà altri utenti in base al numero di vittorie conseguite fino ad un massimo di trenta partite. Solo i migliori giocatori riescono ad ottenere un numero alto di vittorie e a qualificarsi a tornei internazionali dove vengono assunti come giocatori professionisti. Per raggiungere tali livelli oltre ad avere una grande abilità nel giocare bisogna anche esser ottimi manager per riuscire a gestire le proprie risorse virtuali e costruire la squadra migliore possibile per riuscire a scalare le classifiche. FUT però non è solo utilizzata da chi vuole diventare un giocatore professionista ma anzi una gran fetta degli utilizzatori sono semplici utenti che vogliono divertirsi online, sfidando amici in partite virtuali. L’applicazione si pone come obiettivo non solo quello di aiutare l’utente nella creazione della squadra migliore, ma di aiutare anche il videogiocatore alle prime armi che trova difficoltà nello schierare la formazione vista la sua scarsità di risorse virtuali. Spesso coloro che sono alle prime armi spendono soldi reali per progredire più velocemente ed avere più risorse virtuali, è quindi importante avere uno strumento che aiuti a trovare ottimi giocatori a basso prezzo ed investire efficacemente le proprie risorse. Le principali difficoltà nella creazione di una squadra oltre al prezzo dei vari calciatori risiede nei vincoli di intesa, molti utenti trovano difficoltà a schierare i giocatori che abbiano un’affinità adatta agli altri componenti, l’intesa influenza il rendimento del singolo giocatore ed è quindi importante anche questo aspetto. Questi due parametri, intesa e prezzo, sono difficili da gestire efficacemente senza dover sprecare una gran quantità di tempo. Il programma sviluppato vuole proprio ridurre tale spreco di tempo dando la possibilità ai videogiocatori di avere più tempo a disposizione per effettuare partite ed accumulare crediti.

3.DESCRIZIONE DEL DATA-SET

Il data-set utilizzato si compone di tre tabelle: giocatori, affinita e ruolo.

#	Nome	Tipo di dati
1	giocatoreID	INT
2	nome	VARCHAR
3	qualita	VARCHAR
4	revisione	VARCHAR
5	overall	INT
6	club	VARCHAR
7	lega	VARCHAR
8	nazionalita	VARCHAR
9	posizione	VARCHAR
10	prezzo_ps4	INT

La tabella giocatori è stata caricata dal data-set: https://www.kaggle.com/stefanoleone992/fifa-21-ultimate-team-players-and-prices-dataset?select=fut_bin21_players.csv , inizialmente in formato csv e successivamente convertito in formato sql per poter essere sfruttato all'interno dell'applicazione. Da tale data-set sono stati estratti nove attributi significativi e utili all'analisi proposta:

giocatoreID: identificativo univoco di ogni giocatore su FUT

nome: il nome, in gioco, del calciatore

revisione: attributo che può avere due valori: raro o non raro

overall: valore del giocatore, risultato delle varie statistiche degli attributi

club: squadra dove il calciatore gioca (ad esempio Barcellona)

lega: campionato dove il calciatore gioca (ad esempio LaLiga Santander)

nazionalita: nazionalità del calciatore

posizione: ruolo in campo del calciatore

prezzo_ps4: prezzo per la ps4 del calciatore in crediti

In questo data-set alcuni giocatori sono presenti due volte (ad esempio Chiesa), durante l'anno sportivo il giocatore si è trasferito in una squadra diversa e quindi esistono due versioni per lo stesso giocatore. Queste due versioni sono considerate come due giocatori diversi (giocatoreID diverso).

#	Nome	Tipo di dati
1	ruoloID	INT
2	abbreviazione	VARCHAR
3	nome	VARCHAR

#	Nome	Tipo di dati
1	modulo	VARCHAR
2	giocatore1	INT
3	giocatore2	INT

Le due tabelle ruolo e affinita sono state appositamente create.

La tabella ruolo è composta dai seguenti attributi:

ruoloID: identificativo numerico univoco per ogni ruolo

abbreviazione: nome abbreviato utilizzato su FIFA per il dato ruolo (ad esempio ATT)

nome: nome esteso del dato ruolo (ad esempio prima punta)

La tabella affinita lega per ogni modulo i vari giocatori mediante l'identificativo univoco del proprio ruolo:

modulo: il modulo usato per la formazione (ad esempio 4-4-2)

giocatore1: identificativo univoco del primo giocatore

giocatore2: identificativo univoco del secondo giocatore

4.DESCRIZIONE DELLE STRUTTURE DATI E DEGLI ALGORITMI UTILIZZATI

L'applicazione è realizzata mediante l'utilizzo del linguaggio di programmazione Java ed in particolare mediante l'utilizzo degli applicativi JavaFX. Sono stati implementati il pattern MVC (Model-View-Controller), questo per garantire una corretta divisione delle operazioni all'interno del software delegando al Model la logica applicativa mentre il Controller ha il compito di gestire l'iterazione con l'utente e mostrare a video i risultati. Inoltre, il pattern DAO (Data Access Object) ha il compito di gestire l'iterazione con il database.

Il programma è strutturato tramite tre package diversi.

```
▼ 📁 src/main/java
  ▼ 📁 it.polito.tdp.tesi
    > 📄 EntryPoint.java
    > 📄 FXMLController.java
    > 📄 Main.java
```

Il package it.polito.tdp.tesi si occupa mediante tre classi dell'avvio del programma e della gestione dell'interfaccia grafica.

La classe EntryPoint contiene i comandi tecnici necessari per la creazione della scena e del collegamento tra il Model (parte logica) e il Controller (parte grafica).

La classe FXMLController ha il compito di gestire l'iterazione con l'utente rispondendo ai vari comandi input con delle stampe a video.

La classe Main ha la funzione di avviare l'applicazione.







```
▼ 📁 it.polito.tdp.tesi.db
  > 📄 DBConnect.java
  > 📄 GiocatoriDao.java
  > 📄 TestDao.java
```

Il package it.polito.tdp.db si occupa dell'iterazione con il database ' giocatori_fut '.

La classe DBConnect contiene i metodi per la connessione tra l'IDE e il database. In questa classe è necessario modificare username e password una volta scaricato il progetto andando ad inserire le proprie credenziali.

La classe GiocatoriDao contiene tutti i metodi tramite i quali il programma andrà ad estrarre informazioni dal database e successivamente restituite.

La classe TestDao è utilizzata in modo precauzionale per verificare il corretto collegamento con il database.

- ▼  it.polito.tdp.tesi.model
 - >  Collegamenti.java
 - >  Giocatore.java
 - >  Model.java
 - >  Ruolo.java
 - >  Vertice.java

Infine il package it.polito.tdp.model si occupa della parte logica del programma.

Le classi Giocatore e Ruolo sono utilizzate per gestire i calciatori e i loro ruoli in campo, esse implementano le funzioni di hashCode ed equals, utilizzate per il confronto di uguaglianza in base a rispettivamente id del giocatore e id del ruolo.

La classe Vertice è utilizzata per la gestione dei vertici del grafo, essa si compone di due variabili: Giocatore e Ruolo. Anch'essa implementa le funzioni di hashCode ed equals utilizzando come variabile univoca quella del ruolo.

La classe Collegamenti è utilizzata per la creazione degli archi nel grafo, in base al modulo il DAO restituirà una lista di Collegamenti dove saranno presenti gli id dei ruoli utilizzati per formare gli archi.

Infine, la classe Model presenta il vero e proprio codice del programma, oltre alle funzioni per collegare il DAO al Controller presenta i due algoritmi per effettuare la verifica dei parametri inseriti dall'utente e la ricorsione per cercare la squadra migliore.

```

this.grafo = new SimpleWeightedGraph<>(DefaultWeightedEdge.class);

// ricavo i vertici e gli archi utilizzando il modulo
List<Collegamenti> archi = dao.getCollegamenti(modulo);

// controllo se il vertice è già presente, se non è presente lo inserisco
// controllo se il collegamento è presente, se non lo è inserisco l'arco
for(Collegamenti col : archi) {
    Vertice v1= new Vertice(idMapRuoli.get(col.getGiocatore1()), null);
    Vertice v2= new Vertice(idMapRuoli.get(col.getGiocatore2()), null);

    if(!this.grafo.containsVertex(v1)) {
        this.grafo.addVertex(v1);
    }
    if(!this.grafo.containsVertex(v2)) {
        this.grafo.addVertex(v2);
    }
    if(!this.grafo.containsEdge(v1,v2)) {
        // inizialmente metto un peso 2 che è il massimo possibile (utile per verifiche preliminari)
        Graphs.addEdge(this.grafo, v1, v2, 2);
    }
}

// scorro i giocatori e cerco di inserirli nel grafo
int inserito = 0;
for(Giocatore g : giocatori) {
    for(Vertice v : this.grafo.vertexSet()) {
        // controllo se il giocatore è già presente nel grafo
        if(! (v.getGiocatore() == null) ) {
            if(v.getGiocatore().equals(g)) {
                risultato = "ERRORE: \nHai selezionato due volte lo stesso giocatore";
                return risultato;
            }
        }
    }
    // inserisco il giocatore scorrendo gli archi del grafo e andando a contare se
    // è possibile inserire nel grafo tutti i giocatori (rispetto vincoli di modulo)
    if((g.getPosizione().equals(v.getRuolo().getAbbreviazione()) && v.getGiocatore() == null) {

        Integer idVertice = v.getRuolo().getId();
        for(DefaultWeightedEdge e : this.grafo.edgeSet()) {
            if (this.grafo.getEdgeSource(e).getRuolo().getId()==idVertice) {
                this.grafo.getEdgeSource(e).setGiocatore(g);
            }
            if(this.grafo.getEdgeTarget(e).getRuolo().getId()==idVertice) {
                this.grafo.getEdgeTarget(e).setGiocatore(g);
            }
        }
        inserito++;
        break; // il ciclo for viene interrotto evitando che un giocatore venga inserito due volte
    }
}
}

```

Inizialmente la funzione `verificaParametri` andrà a costruire il grafo andando ad inserire vertici ed archi, se non ancora presenti nella struttura, scorrendo la lista di Collegamenti ricevuta dal DAO. A questo punto il programma scorrerà la lista dei quattro giocatori voluti dall'utente, verificando prima che non si ripetano e successivamente se è libero un vertice consono alla posizione del calciatore. Se tale vertice risulta libero viene incrementata la variabile `inserito` che successivamente verrà utilizzata come controllo per verificare che tutti i giocatori siano stati inseriti, in caso contrario verrà avvisato l'utente mediante schermata a video dove è indicato che i calciatori selezionati non sono adatti al modulo o sono stati selezionati due giocatori per lo stesso ruolo. Tramite la funzione `getPeso` verranno aggiunti i pesi preliminari del grafo e successivamente tramite le funzioni `getIntesaSquadra` e `getPrezzo` verranno fatti i controlli sul rispetto dei vincoli dei primi quattro giocatori inseriti, se i controlli non vengono superati verrà avvisato l'utente con un opportuno messaggio di errore visualizzato a video. In caso contrario, se i vincoli sono rispettati allora apparirà a video l'invito di cliccare sul bottone *Cerca Squadra* per trovare i restanti componenti, inoltre verrà visualizzato il budget ancora a disposizione.

```
public List<Vertice> getSquadra(){

    //inizializzo lista dove andrò ad aggiungere i giocatori
    //inizializzo set per i ruoli e giocatori parziali
    List<Ruolo> ruoloParziale = new ArrayList<Ruolo>();
    Set<Vertice> giocatoriParziale = new HashSet<Vertice>();
    Set<Giocatore> listaGiocatori = new HashSet<Giocatore>();
    // aggiungo gli elementi ai set e alla lista appena creati
    for (Vertice v : this.grafo.vertexSet()) {

        if (!(v.getGiocatore() == null)) {
            giocatoriParziale.add(v);
            listaGiocatori.add(v.getGiocatore());
        }else {
            ruoloParziale.add(v.getRuolo());
        }
    }

    // inizializzo le variabili per la ricorsione
    this.intesaBest = 0;
    this.overallBest = 0.0;
    this.costoBest = 0;
    this.squadraBest = new ArrayList<>();

    this.ricorsione(ruoloParziale, giocatoriParziale, listaGiocatori, this.budgetRimasto);
}
```

La funzione `getSquadra` ha il compito di inizializzare le variabili utilizzate nella ricorsione e far partire la procedura vera e propria. Tale algoritmo avrà come caso terminale l'esaurimento dei ruoli da inserire (`ruoloParziale.size() == 0`), nel caso terminale verranno fatte le verifiche per capire se la squadra appena trovata è la migliore. La formazione migliore è stabilita come somma dell'overall e dell'intesa dell'intera squadra (ricordando però che l'intesa di ogni singolo giocatore deve essere almeno di 7). Confrontando questa somma con quella della squadra attualmente migliore si andrà a verificare se la formazione appena trovata dovrà essere inserita nelle variabili della `squadraBest`. Nel caso in cui il valore delle due squadre si eguagliano allora la formazione migliore verrà sostituita solo nel caso in cui il costo totale è inferiore di quello attuale. Nell'ipotesi in cui ci siano ancora ruoli da occupare e quindi non si entra nel caso terminale verrà fatta scorrere la lista dei giocatori, per ognuno di essi si andrà a verificare se ha un prezzo consono al budget rimasto e se è disponibile un vertice per il suo ruolo. Inoltre si andrà a verificare che il giocatore non sia già stato inserito nella formazione, se tali controlli vengono superati allora verrà rimosso il ruolo dalla lista dei ruoli da aggiungere e verrà aggiunto il calciatore sia nella lista dei vertici sia nella lista dei giocatori già in squadra, a questo punto verrà avviata una nuova ricorsione con le variabili appena aggiornate, dopodiché tramite la

procedura di backtracking verrà rimosso il calciatore dalle variabili sopra citate in modo da dare spazio all'aggiunta e alla verifica di nuovi giocatori.

```

Ruolo r = ruoloParziale.get(0);
for(Giocatore g : this.listaGiocatori) {
    // controllo che il giocatore non sia già inserito nella squadra
    if(!(listaGiocatori.contains(g))) {
        // controllo del prezzo giocatore
        if( (budget - g.getPrezzo()) >= 0 ) {
            // controllo se c'è un vertice libero che rispetta i requisiti di posizione
            if((r.getAbbreviazione().equals(g.getPosizione()))) {

                // rimuovo il ruolo dalla lista
                // inserisco giocatore, aggiorno il budget, eseguo nuova ricorsione
                List<Ruolo> ruoliRimasti = new ArrayList<Ruolo>(ruoloParziale);
                ruoliRimasti.remove(r);
                Vertice nuovoGiocatore = new Vertice(r,g);
                giocatoriParziale.add(nuovoGiocatore);
                listaGiocatori.add(g);
                Integer budgetR = budget - g.getPrezzo();

                // lancio ricorsione
                this.ricorsione(ruoliRimasti, giocatoriParziale, listaGiocatori, budgetR);

                // backtracking rimuovo il giocatore appena aggiunto

                giocatoriParziale.remove(nuovoGiocatore);
                listaGiocatori.remove(g);
            }
        }
    }
}

```

Inoltre, vengono implementate funzioni come `aggiungoPesoGrafo()` e `getPeso()` che vengono utilizzate per scorrere gli archi del grafo ed aggiornare il loro relativo peso in base al confronto tra gli attributi nazionalità, lega e club dei giocatori facenti parte dei vertici.

```

// funzione per caricare i pesi
public void aggiungoPesoGrafo() {
    for (DefaultWeightedEdge e : this.grafo.edgeSet()) {
        Giocatore g1 = this.grafo.getEdgeSource(e).getGiocatore();
        Giocatore g2 = this.grafo.getEdgeTarget(e).getGiocatore();

        if ( (g1 != null) && (g2 != null)) {
            // aggiorno il peso
            this.grafo.setEdgeWeight(e, this.getPeso(g1, g2));
        }
    }
}

// funzione che calcola il peso dell'arco
public int getPeso(Giocatore g1, Giocatore g2) {
    // tripla intesa, i tre attributi devono coincidere (stessa squadra implica stessa lega)
    if( (g1.getNazionalita().equals(g2.getNazionalita())) && (g1.getSquadra().equals(g2.getSquadra())) ) {
        return 2;
    } // doppia intesa, 2 attributi in comune
    else if ((g1.getSquadra().equals(g2.getSquadra())) ||
             (g1.getLegga().equals(g2.getLegga()) && g1.getNazionalita().equals(g2.getNazionalita()))) {
        return 1;
    } // singola intesa, un attributo in comune
    else if ((g1.getLegga().equals(g2.getLegga())) || g1.getNazionalita().equals(g2.getNazionalita())) {
        return 0;
    } // nessun attributo in comune
    else
        return -1;
}

```

Le funzioni `getIntesaSquadra()` e `getIntesaGiocatore(Vertex v)` vengono utilizzate per scorrere i vertici del grafo ed assegnare a ogni giocatore la sua intesa che è funzione dei pesi degli archi adiacenti al vertice e al grado del vertice stesso. I valori individuali vengono sommati per trovare l'affinità di squadra totale che potrà avere un massimo di 100, durante questa sommatoria avviene il controllo sull'intesa minima di ogni giocatore, infatti se inferiore a 7, la funzione restituirà il valore -1 che farà in modo che la squadra venga scartata in quanto non rispetti vincoli.

```
public int getIntesaSquadra() {
    int intesaTot = 0;
    for (Vertex v : this.grafo.vertexSet()) {
        int intesa = getIntesaGiocatore(v);
        // controllo che venga rispettato il vincolo di almeno 7 per l'intesa di ogni singolo giocatore
        if (intesa < 7) {
            return -1;
        } else {
            intesaTot += intesa;
        }
    }
    //il massimo di intesa per una squadra può essere 100
    if (intesaTot > 100) {
        intesaTot = 100;
    }
    return intesaTot;
}

public int getIntesaGiocatore(Vertex v) {
    int pesoFinale = 0;
    int grado = this.grafo.degreeOf(v);
    for (Vertex adiacente : Graphs.neighborListOf(this.grafo, v)) {
        int pesoArco = (int) this.grafo.getEdgeWeight(this.grafo.getEdge(v, adiacente));
        pesoFinale += pesoArco;
    }
    if (pesoFinale >= 0) {
        return 10;
    } else if (pesoFinale == -1 || (pesoFinale == -2 && grado > 2) || (pesoFinale == -3 && grado > 4)) {
        return 7;
    } else {
        return 4;
    }
}
}
```

Infine, vengono implementate altre funzioni banali, ma utili per restituire i valori di overall, prezzo, budget rimasto e intesa della squadra migliore trovata.

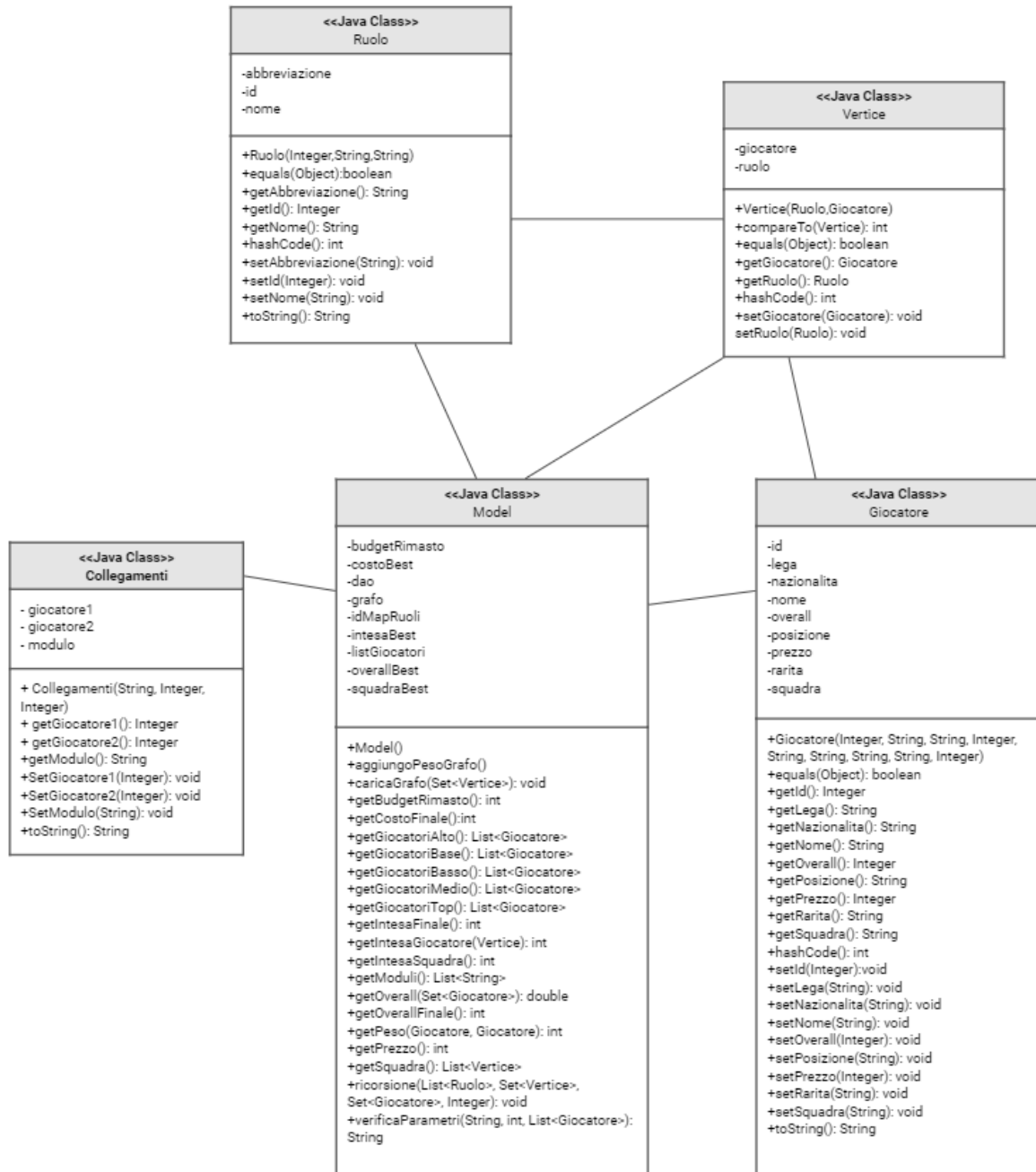
```
// funzione per restituire budget rimasto
public int getBudgetRimasto() {
    return this.budgetRimasto;
}

// funzione per restituire overall
public int getOverallFinale() {
    int risultato = Math.toIntExact(Math.round(this.overallBest));
    return risultato;
}

//funzione per restituire intesa
public int getIntesaFinale() {
    return this.intesaBest;
}

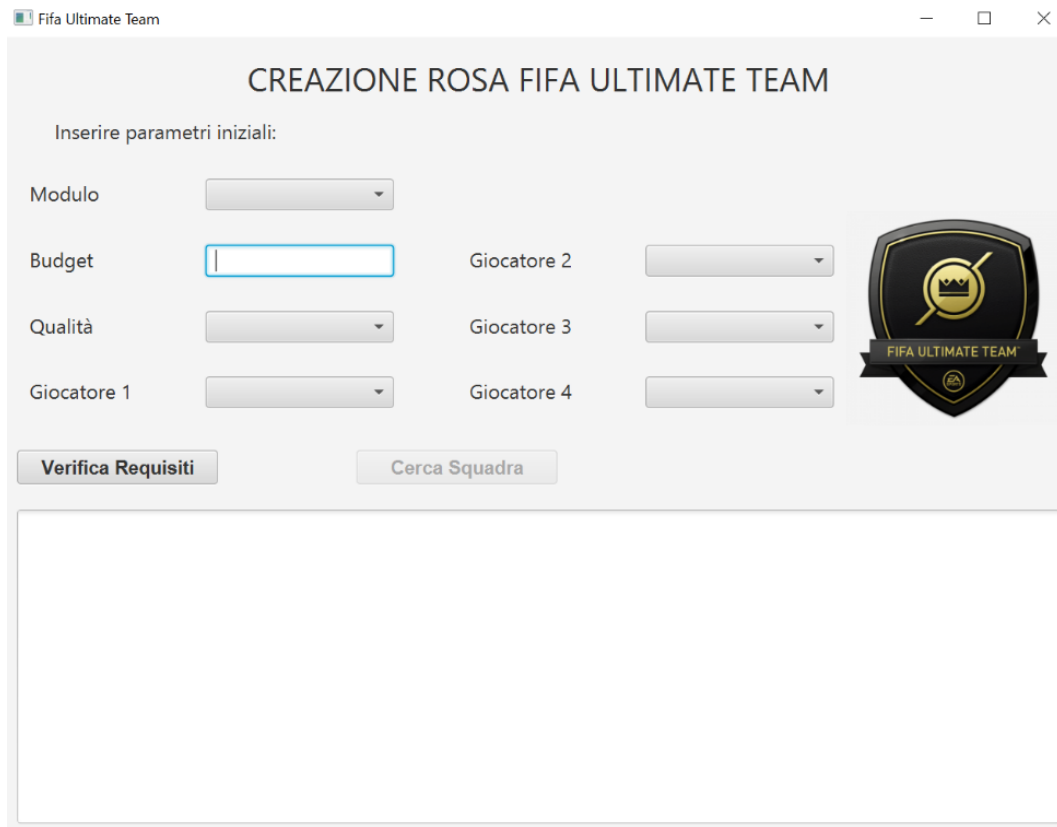
//funzione per restituire prezzo
public int getCostoFinale() {
    return this.costoBest;
}
```

5.DIAGRAMMA DELLE CLASSI PRINCIPALI



6.ESEMPI VIDEO DELL'APPLICAZIONE

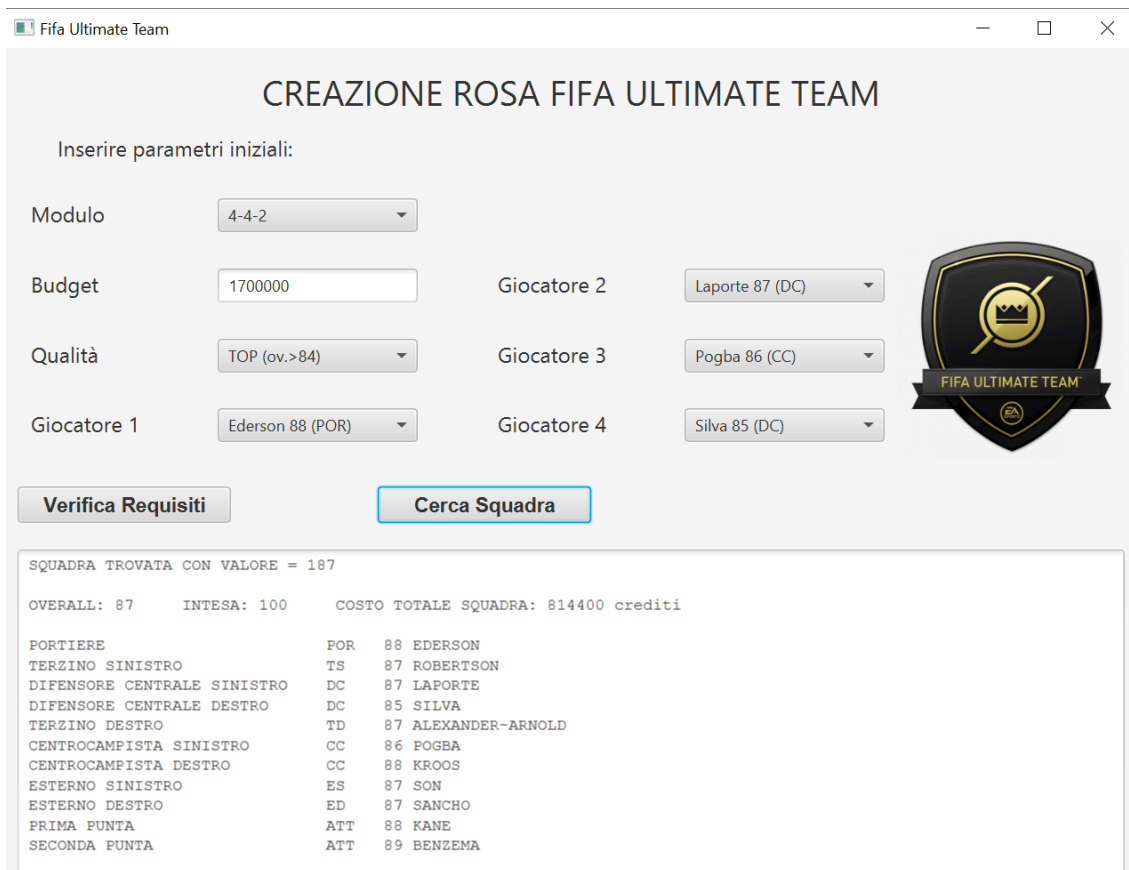
All'avvio dell'applicazione verrà mostrata la seguente schermata dove l'utente andrà ad inserire i vari parametri richiesti prima di verificarli tramite apposito bottone.



Dopo aver cliccato sul bottone *Verifica Requisiti*, se tale verifica ha avuto successo verrà mostrata a video la seguente schermata:



A questo punto il bottone *Cerca Squadra* verrà abilitato e grazie ad esso verrà lanciato il programma ricorsivo per costruire la squadra migliore. Verrà mostrato a video il valore, l'overall, l'intesa e il prezzo della squadra ottenuta oltre che gli 11 giocatori della formazione. (In questo caso, con tali parametri la squadra viene trovata in meno di 3 secondi).



CREAZIONE ROSA FIFA ULTIMATE TEAM

Inserire parametri iniziali:

Modulo: 4-4-2

Budget: 1700000

Qualità: TOP (ov.>84)

Giocatore 1: Ederson 88 (POR)

Giocatore 2: Laporte 87 (DC)

Giocatore 3: Pogba 86 (CC)

Giocatore 4: Silva 85 (DC)

Verifica Requisiti **Cerca Squadra**

SQUADRA TROVATA CON VALORE = 187

OVERALL: 87 INTESA: 100 COSTO TOTALE SQUADRA: 814400 crediti

PORTIERE	POR	88 EDERSON
TERZINO SINISTRO	TS	87 ROBERTSON
DIFENSORE CENTRALE SINISTRO	DC	87 LAPORTE
DIFENSORE CENTRALE DESTRO	DC	85 SILVA
TERZINO DESTRO	TD	87 ALEXANDER-ARNOLD
CENTROCAMPISTA SINISTRO	CC	86 POGBA
CENTROCAMPISTA DESTRO	CC	88 KROOS
ESTERNO SINISTRO	ES	87 SON
ESTERNO DESTRO	ED	87 SANCHEZ
PRIMA PUNTA	ATT	88 KANE
SECONDA PUNTA	ATT	89 BENZEMA

Il video dimostrativo su YouTube, dove è spiegato il funzionamento dell'applicazione, si trova al seguente link: https://youtu.be/3_Zu0uQcVVM.

7.VALUTAZIONI SUI RISULTATI E CONCLUSIONE FINALE

L'applicazione risulta funzionare in modo ottimale per la ricerca di una squadra con qualità 'Top' o 'Alta', queste sono le due specifiche più utilizzate dai videogiocatori che tendono ad usare calciatori con un overall elevato. Il tempo in cui viene mostrato a video la squadra migliore è di circa 5/10 secondi, questo tempo varia in base ai giocatori scelti dall'utente perché più questi giocatori hanno affinità diverse più aumenta la difficoltà della costruzione della formazione, inoltre anche il budget a disposizione fa variare i tempi di costruzione della rosa.

Per le qualità 'Medio', 'Basso' e 'Base' l'algoritmo risulta esser più lento in quanto a differenza delle prime due tipologie, queste raggruppano giocatori con prezzi e overall uguali e quindi vengono verificate un numero maggiori di squadre possibili. Nonostante ciò, il programma riesce a trovare comunque una soluzione in un tempo solitamente inferiore a un minuto, che è da considerarsi breve comparato invece al tempo che si impiegherebbe a costruire tale squadra manualmente.

Il buon funzionamento dell'applicazione è quindi correlato alla bravura dell'utente, esso deve fornire il giusto budget per i quattro giocatori voluti inizialmente e deve sceglierli in modo tale che siano più affini possibili. Ad esempio, se viene scelto come modulo il '4-4-2' e come difensore centrale e portiere due giocatori che non hanno nessun attributo di intesa comune allora il programma sarà vincolato a cercare un giocatore per l'altro ruolo di difensore centrale che abbia affinità con il portiere andando quindi ad escludere un gran numero di giocatori. Se invece i giocatori scelti presentano già alcune affinità allora per il programma sarà più facile trovare la squadra migliore e potrà valutare un maggior numero di calciatori.

In conclusione, il programma risulta esser di facile utilizzo e riesce agevolmente a risolvere il problema dell'individuazione della squadra migliore, l'unica difficoltà riscontrata durante i test dell'applicazione risiede nell'individuare i calciatori adatti al modulo selezionato (ad esempio nel '4-4-2' non viene usato il COC). Questa difficoltà è riscontrata solo per gli utilizzatori che non hanno dimestichezza con FIFA e non conoscono quindi i moduli, questo problema quindi può esser considerato secondario in quanto l'uso dell'applicazione è strettamente legata al videogioco per console FIFA e si presume che l'utente conosca i vari moduli e i ruoli calcistici.



Quest'opera è distribuita con Licenza [Creative Commons Attribuzione - Non commerciale - Condividi allo stesso modo 4.0 Internazionale](https://creativecommons.org/licenses/by-nc-sa/4.0/).