

POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Gestionale
Classe L8 – Ingegneria dell’Informazione



TOOL PER JOB PLACEMENT

Relatore

Prof. Fulvio Corno

Candidato

Navissano Gianmaria

257883

A.A. 2020/2021

Sommario

1.	Proposta di progetto	3
1.1	Descrizione del problema proposto.....	3
1.2	Descrizione della rilevanza gestionale del problema	3
1.3	Descrizione dei data-set per la valutazione.....	3
1.4	Descrizione preliminare degli algoritmi coinvolti	3
1.5	Descrizione preliminare delle funzionalità previste per l'applicazione software	4
2.	Descrizione dettagliata del problema affrontato	4
2.1	Recruiting.....	4
2.2	Ricerca di lavoro post lauream.....	5
3.	Descrizione del data-set utilizzato per l'analisi	5
3.1	Offerte di lavoro.....	5
3.2	Università	6
4.	Diagramma delle classi principali e descrizione degli algoritmi coinvolti	7
4.1	Lancio applicazione e controller	7
4.2	Database	7
4.3	Model	9
5.	Esempi di utilizzo dell'applicazione	15
5.1	Esempio di ricerca delle posizioni di lavoro	15
5.2	Esempio di ricerca delle università	15
5.3	Link del video YouTube	16
6.	Valutazione sui risultati ottenuti e conclusioni	17

1. Proposta di progetto

1.1 Descrizione del problema proposto

L'applicazione intende offrire uno strumento che possa permettere ad un manager in cerca di nuovi impiegati di individuare quali università siano più indicate a fornire i candidati ideali, a seconda dei requisiti e delle competenze richiesti per ottenere la posizione.

La stessa applicazione, divisa in due parti, permetterebbe nella sezione dedicata agli studenti di individuare quali professioni sarebbero più adatte alla preparazione ed alle preferenze del giovane utente in cerca di lavoro.

1.2 Descrizione della rilevanza gestionale del problema

La rilevanza gestionale del problema risiede nella reale necessità in cui può incorrere un ingegnere gestionale, alla ricerca della risorsa ideale per completare il suo team. Il problema descritto rappresenta una situazione che riguarda tutte le aziende, le quali in un certo momento della loro attività (certamente all'inizio, ma anche successivamente a fronte di crescite ed espansioni) si trovano in cerca di nuovi assunti ed hanno dunque bisogno di un criterio per selezionare i candidati. Logicamente nel caso in esame si considerano posizioni per le quali è solitamente richiesto un titolo di studio universitario.

Allo stesso modo degli studenti in procinto di terminare gli studi o recentemente laureati possono trovare in questa applicazione un ottimo strumento per individuare la professione più adatta a loro.

1.3 Descrizione dei data-set per la valutazione

Per quanto riguarda le università è stato individuato il sito <https://www.topuniversities.com> il quale offre un report annuale (dal 2018 al 2021) delle migliori università nel mondo secondo dei punteggi che tengono conto di diversi fattori quali il tasso di studenti esteri, la reputazione accademica, le citazioni per la facoltà, il tasso di assunzione. Tale report è reperibile presso Kaggle al link <https://www.kaggle.com/divyansh22/qs-world-university-rankings?select=2020-QS-World-University-Rankings.csv>

In merito alle professioni è invece presente presso Kaggle un data-set contenente offerte di lavoro da inizio luglio a fine agosto 2019 (per coerenza di periodo considerato viene considerato il ranking delle università relativo al medesimo anno) pubblicate sul sito Naukri. Tale data-set permetterebbe di individuare un grande numero di professioni con relative skills richieste, dettaglio della posizione offerta, esperienza richiesta e salario iniziale. In particolare, sono stati individuati i campi key_skills, Industry e Functional_Area i quali forniscono informazioni ragionevolmente dettagliate, utili per valutare la coerenza della posizione offerta con i requisiti in input nella fase algoritmica.

1.4 Descrizione preliminare degli algoritmi coinvolti

L'applicazione viene realizzata in linguaggio java con l'utilizzo dei pattern MVC (Model View Controller) e DAO (Data Access Object) in modo da risultare distinte le parti di codice relative ad interfaccia, logica applicativa ed accesso al database.

I principali algoritmi che verranno implementati saranno di tipo ricorsivo: l'obiettivo sarà quello di ottenere in output un elenco di università/professioni che soddisfi i vincoli in input, oltre ad un vincolo dimensionale (es. 5 elementi nella lista finale).

Il metodo ricorsivo utilizzerà un metodo (o più d'uno) apposito per il calcolo della bontà dell'elemento (università/professione) considerato in base ai valori di input e deciderà dunque se

aggiungerlo o meno alla soluzione parziale. Una volta esplorate tutte le possibili combinazioni verrà restituita la lista con gli elementi che meglio soddisfano i vincoli.

1.5 Descrizione preliminare delle funzionalità previste per l'applicazione software

L'applicazione, come anticipato, sarà divisa in due sezioni separate: una dedicata ai professionisti in cerca di candidati per ricoprire una determinata posizione e l'altra per gli studenti che desiderano sapere quali professioni siano più adatte a loro.

I professionisti potranno inserire i livelli di competenze richieste per le diverse aree tematiche (competenze IT, artistiche, manageriali, ecc) utilizzando degli slider nell'interfaccia. Sarà anche possibile selezionare una lingua richiesta, così facendo verranno mostrate tra i risultati solo università situate in paesi dove si parla la lingua scelta. In alternativa da un menù a tendina sarà possibile selezionare una professione tra quelle proposte, in questo modo non sarà necessario dover specificare le singole competenze.

Il risultato della ricerca sarà un elenco di università, ordinate dalla più alla meno adatta, di dimensione fissa (es. 5 atenei) che corrispondono ai criteri di ricerca e rappresentano dunque i campus dove cercare il candidato ideale per la posizione offerta.

Gli studenti, nella sezione dedicata, avranno la possibilità di compilare il proprio Curriculum Vitae con il titolo di studio triennale e magistrale ottenuto, oltre alla propria “best skill”, ovvero una competenza padroneggiata ad alti livelli. Infine, sarà possibile selezionare un ambito lavorativo di preferenza, il tutto attraverso l'utilizzo di menù a tendina.

Il risultato della ricerca sarà un elenco di professioni, ordinate dalla più alla meno adatta, che corrispondono ai criteri di ricerca e rappresentano dunque le professioni per le quali lo studente sarebbe più portato e che dovrebbe dunque ambire a ricoprire.

2. Descrizione dettagliata del problema affrontato

2.1 Recruiting

Il problema che questa applicazione si propone di risolvere rappresenta un elemento particolarmente critico nella vita di un'azienda. La decisione di assumere personale, in prima battuta, può essere influenzata da diversi fattori, come la necessità di espandersi a seguito di un aumento della domanda o il desiderio di ampliare i propri confini ed esplorare nuove aree di business o anche solo la nascita di una società. Questa decisione è dunque necessariamente seguita da un'altra scelta importante: dove cercare i candidati per la posizione offerta.

Si tratta di un problema tutt'altro che secondario dal momento che l'assunzione di un dipendente inadatto al lavoro in questione potrebbe causare gravi danni e portare ad inevitabili perdite di tempo e denaro all'azienda.

È stato logicamente individuato come migliore fonte di candidati con un'adeguata preparazione il sistema universitario. Di conseguenza si ritiene che la conoscenza dell'elenco università più indicate ad immettere nel mondo del lavoro i candidati adatti alla posizione offerta, sia di vitale importanza per affrontare con maggior consapevolezza i colloqui e le successive assunzioni. Tale elenco rappresenta precisamente l'output di questa applicazione, calcolato sulla base delle competenze richieste dal recruiter, inserite come valori di input.

2.2 Ricerca di lavoro post lauream

Terminati gli studi universitari si presenta ai neolaureati il difficile compito di trovare un lavoro che li possa soddisfare economicamente e psicologicamente, e per il quale abbiano una preparazione adeguata.

In vista di questo complicato ingresso nel mondo del lavoro, può essere estremamente utile poter fare affidamento su un algoritmo in grado di selezionare le offerte di lavoro più adatte alla preparazione ricevuta e, ovviamente, alle preferenze ed inclinazioni personali dei singoli soggetti. Questa applicazione, nella sua parte dedicata ai neolaureati, permette di ottenere in output un elenco di offerte di lavoro, selezionate in base al curriculum vitae del candidato o all'industria ed all'area di competenza opportunamente selezionate in input, coerentemente con le preferenze e gli interessi personali del candidato.

3. Descrizione del data-set utilizzato per l'analisi

3.1 Offerte di lavoro

La tabella jobs del database contiene un vasto elenco di offerte di lavoro pubblicate sul sito di Naukri nel periodo da inizio luglio a fine agosto 2019 e caratterizzate dai seguenti campi:

- Uniq_Id: identificativo alfanumerico univoco
- Crawl_Timestamp: data e ora di pubblicazione dell'offerta
- Job_Title: titolo dell'offerta
- Job_Salary: salario iniziale offerto
- Job_Experience_Required: anni di esperienza lavorativa richiesta
- Key_Skills: principali competenze necessarie
- Role_Category: categoria del ruolo ricoperto
- Location: città in cui è situato il posto di lavoro
- Functional_Area: area di competenza del lavoratore
- Industry: settore industriale di riferimento
- Role: ruolo ricoperto

Al fine di selezionare un'offerta di lavoro sono stati individuati i campi Functional_Area e Industry, l'insieme dei quali fornisce un adeguato livello di caratterizzazione delle offerte e permette di ottenere un insieme coerente di lavori.

Tool per job placement

[Student side](#) [Recruiter side](#)

Curriculum Vitae

Bachelor degree
Master degree
Best skill

Job Characteristics

Functional area
Industry

[Find jobs](#)

3.2 Università

Le informazioni relative alle università sono contenute all'interno di 7 diverse tabelle così suddivise:

- QSUniversityRankings contiene il ranking relativo al 2019 redatto da Top Universities, il quale si basa su numerosi criteri tra cui la reputazione accademica, il numero di citazioni per facoltà ed il tasso di internazionalità degli studenti, per calcolare il punteggio finale dal quale dipende il ranking annuale. I campi di questa tabella sono:
 - Ranking_2019
 - Ranking_2018
 - Institution_Name
 - Country
 - Classification
 - Subject_Range
 - Research_Intensity
 - Institute_Age
 - Status
 - Academic_Reputation_Score
 - Academic_Reputation_Rank
 - Employer_Reputation_Score
 - Employer_Reputation_Rank
 - Faculty_Student_Score
 - Faculty_Student_Rank
 - Citations_per_Faculty_Score
 - Citations_per_Faculty_Rank
 - International_Faculty_Score
 - International_Faculty_Rank
 - International_Students_Score
 - International_Students_Rank
 - Overall_Score
- Arts_And_Humanities, Engineering_And_Technology, Life_Sciences_And_Technology, Natural_Sciences, Social_Sciences_And_Management sono 5 tabelle ognuna relativa al ranking delle università per singola “Macro Subject”, ovvero le macro categorie delle materie trattate nelle facoltà. Gli unici due campi di queste tabelle sono
 - University_Name
 - Rank

- University_Rankings_By_Subject è la tabella dedicata al ranking delle università per materia. I campi della tabella sono, oltre al nome dell'università (University_name), i ranking relativi alle 46 materie selezionate nel QS World University Ranking By Subject.

4. Diagramma delle classi principali e descrizione degli algoritmi coinvolti

Per la realizzazione dell'applicazione è stato impiegato il pattern MVC (Model-View-Controller) la cui struttura permette di tenere separate la gestione dell'interfaccia utente, la logica applicativa e le operazioni legate all'interazione con il database.

In conseguenza di questa ripartizione del codice, il programma si divide in 3 pacchetti: it.polito.tdp.prova_tesi, database e model.

4.1 Lancio applicazione e controller

Il pacchetto it.polito.tdp.prova_tesi contiene le classi Main e EntryPoint, che si occupano di gestire il lancio dell'applicazione e la classe FXMLController, la quale gestisce l'interazione con l'interfaccia utente.

L'interfaccia è realizzata tramite Scene Builder e si trova all'interno del pacchetto src/main/resources in formato fxml. La classe FXMLController contiene tutti i metodi legati alle operazioni effettuate dell'utente (click di un bottone, selezione da un menù a tendina, etc...) e gestisce tutti i possibili errori di input.

Di seguito un esempio del metodo che gestisce la ricerca delle università (lato recruiter):

```
@FXML
void doFindUniversities(ActionEvent event) {
    this.resultUniversitiesTxt.clear();
    String language = this.languageBox.getValue();
    if(language == null) {
        this.resultUniversitiesTxt.appendText("You didn't select any language\n");
        return;
    }
    Job jobOnOffer = this.boxJobOnOffer.getValue();

    List<University> bestUniversities = new LinkedList<>();
    if(jobOnOffer==null) {
        String subject = this.mainSubjectBox.getValue();
        if(subject==null) {
            this.resultUniversitiesTxt.appendText("You didn't select a main subject\n");
            return;
        }
        bestUniversities = this.model.getBestUniversities(language, subject, (int)artsAndHumanitiesSlider.getValue(),
            (int)engineeringAndTechnologySlider.getValue(), (int)lifeSciencesAndMedicineSlider.getValue(), (int)naturalSciencesSlider.getValue(),
            (int)socialSciencesAndManagementSlider.getValue());
    } else {
        bestUniversities = this.model.getBestUniversities(jobOnOffer, language);
    }
    if(bestUniversities==null) {
        this.resultUniversitiesTxt.appendText("Oh no! Something went wrong\n");
    } else {
        this.resultUniversitiesTxt.appendText("Here are the best 5 universities that will provide suitable candidates for the position you are offering:\n");
        for(University u : bestUniversities) {
            this.resultUniversitiesTxt.appendText(u.toStringRes()+"\n");
        }
    }
}
```

Come si evince dal codice questa classe non gestisce nessuna parte della logica applicativa, ma si limita a controllare i dati di input e stampare gli output, forniti dai metodi della classe Model.

4.2 Database

All'interno di questo package si trova la classe DBConnect, la quale realizza una connessione con il database attraverso l'impiego dei metodi forniti dalla libreria HikariCP.

In particolare, questa libreria permette di fare “connection pooling”, ovvero di realizzare un insieme di connessioni aperte in fase di inizializzazione e gestite da questa classe. Tale tecnica permette di

risparmiare tempo prezioso nelle lunghe operazioni di apertura e chiusura delle connessioni, non più necessarie.

```
package database;

import java.sql.Connection;
import java.sql.SQLException;

import com.zaxxer.hikari.HikariConfig;
import com.zaxxer.hikari.HikariDataSource;

public class DBConnect {

    private static final String jdbcURL = "jdbc:mysql://localhost/naukrijobs";
    private static HikariDataSource ds;

    public static Connection getConnection() {

        if(ds == null) {
            HikariConfig config = new HikariConfig();
            config.setJdbcUrl(jdbcURL);
            config.setUsername("");
            config.setPassword("");

            // configurazione MySQL
            config.addDataSourceProperty("cachePrepStmts", "true");
            config.addDataSourceProperty("prepStmtCacheSize", "250");
            config.addDataSourceProperty("prepStmtCacheSqlLimit", "2048");

            ds = new HikariDataSource(config);
        }

        try {
            return ds.getConnection();
        } catch(SQLException e) {
            System.err.println("Errore connessione al DB");
            throw new RuntimeException(e);
        }
    }
}
```

Le classi JobsDAO e UniversitiesDAO invece si occupano di effettuare le operazioni sui dati relativi, rispettivamente, alle offerte di lavoro ed alle università, implementando il pattern DAO (Data Access Object).

JobsDAO permette di interrogare la tabella jobs del database. I metodi di questa classe sono 4, molto simili tra loro per struttura e funzionalità:

- getAllSkills() si occupa di fornire una lista di stringhe ognuna delle quali relativa ad un elenco di skills richieste per ogni posizione. Successivamente nella classe Model un metodo apposito divide questi elenchi in singole stringhe per permettere alla classe FXMLController di realizzarne la stampa all’interfaccia in un menù a tendina.
- getAllJobs() fornisce una lista di oggetti Job, i quali sono caratterizzati da 6 attributi: il titolo dell’offerta, l’industria di riferimento, l’area di competenza, il salario, le skills ed il ruolo ricoperto.
- getAllIndustries() e getAllFunctionalAreas() interrogano la tabella jobs per ottenere l’elenco di tutte le industrie (campo Industry) e le aree di competenza (Functional_Area).

UniversitiesDAO interroga numerose le tabelle relative alle università, mediante 9 metodi, anch’essi simili tra loro e per la maggior parte privi di valori di ritorno:

- getAllCountries() fornisce un elenco di stringhe, che rappresenta l’insieme dei paesi nei quali si trova almeno un’università presente a database.

- setAllUniversities(Map<String, University> idMapUni) gestisce la scrittura della mappa idMapUni, la quale contiene oggetti di tipo University relativi a tutte le università presenti nella tabella QSUniversityRankings.
- setMacroSubjectsRankings(Map<String, University> idMapUni) è un metodo che ne richiama altri 5 agendo da “passa carte”. Tali metodi implementano la scrittura, per ogni oggetto University all’interno della mappa idMapUni, del ranking relativo alle 5 “MacroSubjects”:
 - setArtAndHumanitiesRank(Map<String, University> idMapUni)
 - setEngineeringAndTechnologyRank(Map<String, University> idMapUni)
 - setLifeSciencesAndTechnologyRank(Map<String, University> idMapUni)
 - setNaturalSciencesRank(Map<String, University> idMapUni)
 - setSocialSciencesAndManagementRank(Map<String, University> idMapUni)
- setAllSubjects(Map<String, University> idMapUni) infine è un metodo realizzato per la scrittura dei 46 attributi relativi ai ranking per ogni materia degli oggetti University nella idMapUni.

Si riporta di seguito a titolo d’esempio il metodo setArtAndHumanitiesRank(Map<String, University> idMapUni) della classe UniversitiesDAO.

```
public void setArtAndHumanitiesRank(Map<String, University> idMapUni) {
    String sql = "SELECT q.institution_name, a.rank "
        + "FROM qsuniversityrankings q, Arts_And_Humanities a "
        + "WHERE q.institution_name IS NOT NULL AND q.ranking_2019 <>'N/A' "
        + "AND q.institution_name = a.university_name";
    try {
        Connection conn = DBConnect.getConnection();
        PreparedStatement st = conn.prepareStatement(sql);
        ResultSet rs = st.executeQuery();

        while(rs.next()) {
            if(idMapUni.containsKey(rs.getString("q.institution_name"))) {

                University u = idMapUni.get(rs.getString("q.institution_name"));
                u.setArts_And_Humanities_Rank(rs.getInt("a.rank"));

            }
        }
        conn.close();
    } catch(SQLException e) {
        e.printStackTrace();
    }
}
```

4.3 Model

All’interno del package model vi sono numerose classi con strutture e funzionalità ben diverse. Le classi Job e University rappresentano gli oggetti principali dell’applicazione e se ne fa logicamente un grande utilizzo in tutte le parti del codice. La classe Language rappresenta una struttura che viene utilizzata per associare ad ogni lingua i vari paesi in cui è parlata. Le classi JobsComparator e UniversityByName implementano l’interfaccia Comparator<> e si occupano di gestire, rispettivamente, l’ordinamento delle offerte di lavoro (in base alla skill selezionata) e delle università (in base al valore dell’attributo tempoScore).

Model è la classe che implementa tutta la logica applicativa e per questa ragione è di gran lunga la più corposa dell’applicazione.

I suoi numerosi metodi gestiscono 2 tipologie di operazioni: forniscono i dati all’interfaccia (FXMLController) per riempire i menu a tendina ed implementano complesse operazioni algoritmiche per individuare le liste di università ed offerte di lavoro da fornire per l’output.

Rientrano nella prima categoria i seguenti metodi, relativi all’interfaccia lato studenti:

- getAllSkills() sfruttando il metodo omonimo della classe JobsDAO ottiene una lista di elenchi di skills, che poi vengono divise utilizzando String.split() ed inserite in una lista da ritornare al controller per riempire l’apposito menu a tendina.
- getAllIndustries() fornisce una lista con tutti settori industriali per cui esiste un’offerta di lavoro nel database.
- getAllFunctionalAreas() fornisce una lista con tutte le aree di competenza per cui esiste un’offerta di lavoro nel database.
- getAllUni() fornisce una lista con tutte le università presenti nel database, sfruttando una mappa (idMapUni) inizializzata nel costruttore della classe.

I seguenti metodi permettono il riempimento di menu a tendina dell’interfaccia lato recruiter:

- getAllJobs() fornisce una lista con tutte le offerte di lavoro presenti nel database, sfruttando una lista (jobs) inizializzata nel costruttore della classe.
- getLanguages() fornisce una lista di tutte le lingue parlate negli stati per i quali è presente almeno un’università nel database. Questa estrazione è realizzata sfruttando una mappa (langMap) inizializzata nel costruttore della classe attraverso il metodo setLanguages(List<String> countries) il quale associa ad ogni stato le lingue che vi sono parlate:

```
public Map<String, Language> setLanguages(List<String> countries) {  
    Map<String, Language> result = new HashMap<>();  
  
    for(String country : countries) {  
        switch(country) {  
            case "Argentina":  
                if(!result.containsKey("Spanish"))  
                    result.put("Spanish", new Language("Spanish", country));  
                else result.get("Spanish").addCountry(country);  
                break;  
            case "Australia":  
                if(!result.containsKey("English"))  
                    result.put("English", new Language("English", country));  
                else result.get("English").addCountry(country);  
                break;  
            case "Austria":  
                if(!result.containsKey("German"))  
                    result.put("German", new Language("German", country));  
                else result.get("German").addCountry(country);  
                break;  
            case "Azerbaijan":  
                if(!result.containsKey("Azerbaijani"))  
                    result.put("Azerbaijani", new Language("Azerbaijani", country));  
                else result.get("Azerbaijani").addCountry(country);  
                break;  
        }  
    }  
}
```

- getAllSubjects() fornisce una lista di tutte le 46 materie contenute nella tabella University_Rankings_By_Subject.

La seconda tipologia di metodi della classe Model rappresenta il cuore dell’applicazione dal momento che gestisce il calcolo delle liste finali delle migliori università ed offerte di lavoro. Per quanto riguarda le università i metodi getBestUniversities(), cercaUni() e calcolaScore() esistono in due versioni. La prima viene utilizzata quando si sceglie di utilizzare gli slider presso l’interfaccia per le “Macro Subject”:

Select the requested level of preparation in the following macro subject

Arts & Humanities	<input type="range"/>
Engineering & Technology	<input type="range"/>
Life Sciences & Medicine	<input type="range"/>
Natural Sciences	<input type="range"/>
Social Sciences & Management	<input type="range"/>

- **getBestUniversities(String language, String subject, int artsAndHumanitiesValue, int engineeringAndTechnologyValue, int lifeSciencesAndMedicineValue, int naturalSciencesValue, int socialSciencesAndManagementValue)** riceve in input la lingua parlata nell'università, la materia di maggiore importanza ed i valori degli slider ad interfaccia relativi alle "Macro Subject". Sfruttando il metodo setValidUniversities(String language) viene per prima cosa inizializzata la lista validUniversities la quale contiene soltanto le università situate in paesi dove si parla la lingua selezionata; successivamente viene richiamato il metodo ricorsivo cercaUni(Set<University> parziale, String subject, int artsAndHumanitiesValue, int engineeringAndTechnologyValue, int lifeSciencesAndMedicineValue, int naturalSciencesValue, int socialSciencesAndManagementValue, double score, int L) ed infine viene ordinata la lista bestUniversities che verrà stampata in output dal controller.
- **cercaUni(Set<University> parziale, String subject, int artsAndHumanitiesValue, int engineeringAndTechnologyValue, int lifeSciencesAndMedicineValue, int naturalSciencesValue, int socialSciencesAndManagementValue, double score, int L)** è la versione del metodo ricorsivo utilizzata per calcolare le migliori 5 università basandosi sui valori in input degli slider delle 5 "Macro Subject". La struttura di questo metodo è tipica della ricorsione: il primo blocco di codice corrisponde al caso terminale, ovvero ogni volta che la lista parziale raggiunge la dimensione desiderata (5 elementi). In corrispondenza di questa casistica si valuterà se aggiornare o meno la lista "bestUniversities" ed il "bestScore" a seconda che lo "score" sia migliore (maggiore) o meno del best score attuale. Il blocco di istruzioni successive riguarda la generazione di sottoproblemi, realizzata attraverso l'aggiunta e la NON-aggiunta alla soluzione parziale dell'elemento in posizione L (livello di ricorsione) della lista validUniversities. Si è preferita questa struttura in quanto comporta una complessità algoritmica esponenziale, a differenza di un metodo ricorsivo basato su dei cicli (for) il quale avrebbe avuto una complessità fattoriale, portando a prestazioni ben peggiori dell'applicazione.
- **calcolaScore(University u, int artsAndHumanitiesValue, int engineeringAndTechnologyValue, int lifeSciencesAndMedicineValue, int naturalSciencesValue, int socialSciencesAndManagementValue, String subject)** viene richiamato ogni volta che viene aggiunto un elemento alla lista parziale dal metodo cercaUni per calcolare il punteggio associato all'università passata come parametro. Tale punteggio dipende dall'attributo overall_Score di University, dal ranking della materia principale passata come parametro (subject) e dai ranking delle macro materie pesati con i valori degli slider passati come parametri. Il contributo dei ranking è dato dalla seguente formula:
$$\text{score} = (\text{double})\text{multi} * (1.0 / (\text{double})\text{subjectRank})$$
 nella quale "multi" è un moltiplicatore del valore di 1000, per avere punteggi nell'ordine delle centinaia e subjectRank è il ranking per la materia. Dopo aver calcolato il punteggio il metodo aggiorna l'attributo tempScore dell'università e poi ritorna il valore ottenuto.

```

public double calcolaScore(University u, int artsAndHumanitiesValue,
    int engineeringAndTechnologyValue, int lifeSciencesAndMedicineValue,
    int naturalSciencesValue, int socialSciencesAndManagementValue, String subject) {

    double score = 0.0;
    int multi = 1000; //moltiplicatore

    //il punteggio è dato dalla somma del ranking generico, dato dall'overall score
    score += u.getOverall_Score();

    //a cui va sommato il punteggio dato dalla Main Subject
    score += (double)multi*(1.0/(double)u.getSubject(subject));

    //e dalla somma dei rankings per Macro-subject

    if(u.getArts_And_Humanities_Rank()>0) {
        score += (double)multi*(double)artsAndHumanitiesValue*(1.0/(double)u.getArts_And_Humanities_Rank());
    }
    if(u.getEngineering_And_Technology_Rank()>0) {
        score += (double)multi*(double)engineeringAndTechnologyValue*(1.0/(double)u.getEngineering_And_Technology_Rank());
    }
    if(u.getLife_Sciences_And_Technology_Rank()>0) {
        score += (double)multi*(double)lifeSciencesAndMedicineValue*(1.0/(double)u.getLife_Sciences_And_Technology_Rank());
    }
    if(u.getNatural_Sciences_Rank()>0) {
        score += (double)multi*(double)naturalSciencesValue*(1.0/(double)u.getNatural_Sciences_Rank());
    }
    if(u.getSocial_Sciences_And_Management_Rank()>0) {
        score += (double)multi*(double)socialSciencesAndManagementValue*(1.0/(double)u.getSocial_Sciences_And_Management_Rank());
    }

    u.setTempScore(score);

    return score;
}

```

Nel caso invece si scelga di selezionare la posizione di lavoro offerta, verranno automaticamente azzerati i valori degli slider e saranno richiamati i metodi precedenti nelle loro versioni alternative:

Select the position on offer

By selecting the position on offer, there will be no need for the previous input values, except for the language requirement



- **getBestUniversities(Job jobOnOffer, String language)** riceve in input soltanto la lingua selezionata e l'offerta di lavoro. Questo metodo per prima cosa sfrutta `getSubjectByIndustryAndFunctionalArea(jobOnOffer.getIndustry(), jobOnOffer.getFunctional_Area())` per ottenere la lista di materie determinanti per il lavoro offerto, successivamente inizializza la lista `validUniversities`, dunque richiama il metodo ricorsivo `cercaUni(Set<University> parziale, List<String> subjects, double score, int L)` ed infine ordina la lista `bestUniversities`, ritornandola poi al controller.
- **getSubjectByIndustryAndFunctionalArea(jobOnOffer.getIndustry(), jobOnOffer.getFunctional_Area())** utilizza 2 costrutti switch per associare il settore industriale e l'area di competenza passati come parametri a 2 o più materie, che li caratterizzino.

```

public List<String> getSubjectByIndustryAndFunctionalArea(String industry, String functional_Area){

    List<String> result = new LinkedList<>();

    switch(industry) {
        case "Accounting, Finance":
            result.add("Accounting_and_Finance");
            break;
        case "Advertising, PR, MR, Event Management":
            result.add("Communication_and_Media_Studies");
            break;
        case "Agriculture, Dairy":
            result.add("Agriculture_and_Forestry");
            break;
        case "Animation, Gaming":
            result.add("Computer_Science_and_Information");
            break;
        case "Architecture, Interior Design":
            result.add("Architecture_and_Built_Environment");
            result.add("Art_and_Design");
            break;
    }
}

```

- **cercaUni(Set<University> parziale, List<String> subjects, double score, int L)** è molto simile alla versione alternativa ed omonima di questo metodo, con le sole differenze dei valori di input e della versione del metodo calcolaScore(...) utilizzata.
- **calcolaScore(University u, List<String> subjects)** calcola il punteggio relativo all'università passata come parametro basandosi sull'attributo overall_Score dell'oggetto University e dei ranking nelle subject passate come parametro (lista subjects) secondo le stesse modalità computazionali e lo stesso moltiplicatore utilizzati nella versione alternativa di questo metodo. Dopo aver calcolato il punteggio il metodo aggiorna l'attributo tempScore dell'università e poi ritorna il valore ottenuto.

Per quanto concerne il calcolo della lista delle posizioni di lavoro (lato studente) esso è realizzato dal metodo **getBestJobs(...)** disponibile anch'esso in 2 versioni. La prima viene richiamata quando sono selezionati in input la “Industry” e la “Functional Area” relative alla posizione cercata:

Job Characteristics

Functional area

Industry

Find jobs

- **getBestJobs(String industry, String functional_Area, String skill)** riceve in input la industry e la functional_Area scelte dal neolaureato e, scorrendo la lista “jobs”, inizializzata nel costruttore di Model, ottiene la lista delle posizioni di lavoro che corrispondono a quella coppia di valori.

In alternativa, l'utente può inserire il proprio Curriculum Vitae, composto dall'università frequentata in triennale (Bachelor Degree) e da quella frequentata in magistrale (Master Degree), alle quali si aggiunge la “Best skill” ovvero la sua migliore abilità.

Curriculum Vitae

Bachelor degree

Master degree

Best skill

- **List<Job> getBestJobs(University uBachelor, University uMaster, String skill)** permette di associare alle università in input la/le materie nelle quali tali università eccellono maggiormente (secondo i valori dei ranking per materia). Così facendo è possibile utilizzare il metodo **getSubjectByIndustryAndFunctionalArea(...)** per ogni coppia di industry e functional_Area per individuare quali settori industriali e quali aree di competenza siano più adatti alla preparazione fornita da tali università. Una volta ottenute queste due liste (industries e functionalAreas), vengono selezionate le offerte di lavoro dalla lista “jobs” che corrispondono a tutte le coppie di industry – functional_Area delle due liste.

```

public List<Job> getBestJobs(University uBachelor, University uMaster, String skill){

    //individuo le subject principali delle università in input
    List<String> subjectUBachelor = new LinkedList<>();
    List<String> subjectUMaster = new LinkedList<>();

    if(uBachelor!=null)
        subjectUBachelor = uBachelor.getBestSubject(this.getAllSubjects());
    if(uMaster!=null)
        subjectUMaster = uMaster.getBestSubject(this.getAllSubjects());

    //individuo le industry legate alle subject trovate
    List<String> industries = new LinkedList<>();
    for(String i : this.getAllIndustries()) {
        for(String subjectByIndustry : this.getSubjectByIndustryAndFunctionalArea(i, ""))
            for(String subjectBachelor : subjectUBachelor) {
                if(subjectBachelor.equals(subjectByIndustry) && !industries.contains(i))
                    industries.add(i);
            }
        for(String subjectMaster : subjectUMaster) {
            if(subjectMaster.equals(subjectByIndustry) && !industries.contains(i))
                industries.add(i);
        }
    }
}

//individuo le functional_Area legate alle subject trovate
List<String> functionalAreas = new LinkedList<>();
for(String fa : this.getAllFunctionalAreas()) {
    for(String subjectByFArea : this.getSubjectByIndustryAndFunctionalArea("", fa)) {
        for(String subjectBachelor : subjectUBachelor) {
            if(subjectBachelor.equals(subjectByFArea) && !functionalAreas.contains(fa))
                functionalAreas.add(fa);
        }
        for(String subjectMaster : subjectUMaster) {
            if(subjectMaster.equals(subjectByFArea) && !functionalAreas.contains(fa))
                functionalAreas.add(fa);
        }
    }
}

if(industries.isEmpty() || functionalAreas.isEmpty())
    return null;

//infine individuo i job legati a quelle industry e functionalArea
List<Job> result = new LinkedList<>();

for(Job j : this.jobs) {
    for(String i : industries) {
        for(String fa : functionalAreas) {
            if(j.getFunctional_Area().equals(fa) && j.getIndustry().equals(i))
                result.add(j);
        }
    }
}

if(result.isEmpty())
    return null;

//ordiniamo i lavori presenti in result in base alla presenza della best skill
Collections.sort(result, new JobsComparator(skill));

return result;
}

```

5. Esempi di utilizzo dell'applicazione

5.1 Esempio di ricerca delle posizioni di lavoro

Si desidera conoscere quali posizioni di lavoro siano più adatte alla preparazione accademica ricevuta. Vengono inserite in input le università frequentate in triennale e magistrale, per esempio nel caso di uno studente particolarmente brillante si tratta del King's College e di Harvard. Viene poi selezionata come best skill il recruiting, che rappresenta oltre che un'abilità un interesse dell'utente.

Di seguito viene presentato l'output fornito dall'applicazione:

Tool per job placement

Student side X Recruiter side

Curriculum Vitae

Bachelor degree: KING'S COLLEGE LONDON (KCL)
Master degree: HARVARD UNIVERSITY
Best skill: recruiting

Job Characteristics

Functional area:
Industry:

Find jobs

JOB TITLE: Urgent: IT Recruiter / Technical Recruiter @ Malad (W)
ROLE: Staffing Specialist/ Manpower Planning

JOB TITLE: Business Development Executive/Manager- Recruitment
ROLE: Recruitment Manager

JOB TITLE: HR Executive - Recruitment - Female
ROLE: Recruitment Executive

JOB TITLE: Female Recruitment Executive @ Kandivali east
ROLE: Recruitment Executive

JOB TITLE: SR/ Attending Cnslt
ROLE: Medical Officer

5.2 Esempio di ricerca delle università

Un recruiter desidera conoscere le università che hanno la maggiore probabilità di formare il candidato adatto alla posizione offerta. Specifica, dunque, l'importanza delle macro materie: in particolare 4/5 per Engineering & Technology, 2/5 per Life Sciences & Medicine e 3/5 per Social

Sciences & Management. La lingua scelta è l'italiano e, infine, viene selezionata come materia principale Architecture and Built Environment.

Viene riportato di seguito l'output fornito dall'applicazione:

Tool per job placement

Student side Recruiter side X

Select the requested level of preparation in the following macro subject

Arts & Humanities

Engineering & Technology

Life Sciences & Medicine

Natural Sciences

Social Sciences & Management

Select the language requirement

Italian

Select the main subject requirement

Architecture and Built Envi...

Select the position on offer

By selecting the position on offer, there will be no need for the previous input values, except for the language requirement

Here are the best 5 universities that will provide suitable candidates for the position you are offer:
POLITECNICO DI MILANO, Italy SCORE = 410.44
POLITECNICO DI TORINO, Italy SCORE = 152.57
SCUOLA SUPERIORE SANT'ANNA PISA, Italy SCORE = 47.7
SCUOLA NORMALE SUPERIORE DI PISA, Italy SCORE = 46.4
UNIVERSITÀ DI BOLOGNA (UNIBO), Italy SCORE = 45.9

5.3 Link del video YouTube

Di seguito il link al video di presentazione del funzionamento dell'applicazione:

<https://youtu.be/9bTc8BW3bDY>

6. Valutazione sui risultati ottenuti e conclusioni

Si ritiene che i risultati ottenuti siano particolarmente soddisfacenti nella parte dell'applicazione dedicata alle università, la quale si basa su un solido database dall'elevata complessità, che permette di conoscere il "valore" dei singoli atenei per le singole materie con grande precisione. Si segnala tuttavia che, a causa della notevole dimensione di tale data-set, risulta estremamente oneroso dal punto di vista algoritmico il calcolo dell'output nel caso venga selezionata una lingua parlata in paesi con un grande numero di università.

Sfortunatamente il data-set relativo alle offerte di lavoro non offre lo stesso livello di precisione ed affidabilità, è stato comunque sfruttato al meglio delle sue capacità e riporta risultati soddisfacenti nella maggior parte dei casi.

Un elemento migliorabile dell'applicazione potrebbe dunque essere sicuramente l'utilizzo di un diverso data-set per le posizioni lavorative.

In conclusione, gli output forniti dall'applicazione rappresentano sicuramente un elemento di grande utilità per le due categorie di utenti e si considera dunque raggiunto con successo l'obiettivo di questo progetto.



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