Politecnico di Torino

Dipartimento di Ingegneria Gestionale e della Produzione
Corso di Laurea Triennale in Ingegneria Gestionale
Classe L8 – Ingegneria dell'Informazione
A.A. 2022/2023

JavaFX Remote Learning Tool

Software per la generazione automatica di un piano formativo personalizzato

Relatore
Prof. Fulvio Corno

Candidato
Erika Francesca Caragnano

Matricola
281287

24 Novembre 2023

Indice dei contenuti

Capitolo 1 – Proposta di Progetto

1.1 Studente proponente

s281287 Caragnano Erika Francesca

1.2 Titolo della proposta

JavaFX Remote Learning: software per la selezione di piani formativi personalizzati.

1.3 Descrizione del problema proposto

La digitalizzazione della formazione rappresenta un potente catalizzatore per l'espansione dell'accesso democratico alla conoscenza. Attraverso l'incremento della flessibilità, l'abbattimento delle barriere economiche, la personalizzazione del percorso formativo e la riduzione dei costi operativi, l'istruzione può diventare un servizio pubblico universalmente accessibile a persone di tutte le età e condizioni socioeconomiche.

Nell'ambito di tale visione, si prevede lo sviluppo di un'applicazione JavaFX dedicata all'analisi di database provenienti da piattaforme di apprendimento online, come Udemy, che offrono corsi completamente fruibili a distanza. Questo software avrà la capacità di navigare il catalogo dei corsi disponibili, permettendo agli utenti di eseguire ricerche mirate e filtri specifici (ad esempio, per Materia, Argomento, Popolarità, Livello di difficoltà, Prezzo, Valutazione, ecc.), nonché di comporre manualmente un piano di studi personalizzato.

In aggiunta, il programma fornirà analisi dettagliate e statistiche riguardo alla composizione del piano di studi selezionato, inclusi parametri come la spesa totale, il livello di interdisciplinarità, la durata complessiva, tra gli altri.

Il fulcro dell'applicazione sarà costituito da un algoritmo ricorsivo per la creazione di piani formativi su misura, calibrati in base al tempo e al budget disponibili dichiarati dall'utente, nonché in relazione ai suoi obiettivi educativi e professionali. Questo strumento si pone come un passo innovativo verso una formazione personalizzata e accessibile, riflettendo l'impegno verso un'istruzione più inclusiva e adattabile alle esigenze individuali.

1.4 Descrizione della rilevanza gestionale del problema

La formazione è una forma di investimento spesso soggetta a forti limitazioni di tempo, spazio e denaro. In regime di scarsezza di queste risorse, il Remote Learning può rendere il sapere maggiormente fruibile.

Questo tool si propone di aiutare studenti, lavoratori e amatori da tutto il mondo nella selezione automatica del piano formativo più adatto alle proprie esigenze, lasciando che questi scelgano un argomento e ottengano la migliore selezione di corsi disponibile a catalogo che tenga conto dei corsi già seguiti e del loro livello di preparazione iniziale.

La logica di generazione del piano formativo in videostreaming segue un principio di gradualità dell'apprendimento verso stadi di conoscenza via via incrementali.

1.5 Descrizione dei data-set per la valutazione

Il software interagirà con un data-set proveniente dalla piattaforma di condivisione di dati Kaggle all'indirizzo https://www.kaggle.com/datasets/thedevastator/udemy-courses-revenue-generation-and-course-anal

I corsi, originariamente suddivisi su quattro tabelle afferenti ad una materia diversa, sono stati assemblati in una sola tabella che dispone di 3521 istanze.

I campi del dataset sono i seguenti:

campo	***descrizione***	***variabile***
course_id	ID univoco del corso	INT
course_title	Titolo del corso	VARCHAR
url	URL del corso sul sito Udemy	VARCHAR
price	Prezzo per l'acquisto del singolo corso	DOUBLE (dollari)
num_subscribers	Numero di utenti già iscritti al corso	INT
num_reviews	Numero di recensioni del corso	INT
num_lectures	Numero di lezioni di cui il corso è composto	INT
level	Livello di preparazione richiesto	VARCHAR (All levels, Beginner level, Intermediate level, Expert level)
rating	Indice di gradimento degli iscritti al corso	DOUBLE (percentuale 0-100%)
content_duration	Durata dell'intero corso	DOUBLE (ore)
timestamp	Data di pubblicazione del corso	DATE (YYYY-MM-DD)
subject	Materia di riferimento del corso	VARCHAR (Web Development, Graphic Design, Musical Instruments, Business Finance)

Il dataset, originariamente in formato CSV, non ha necessitato di particolari modifiche per l'importazione in HeidiSQL ed è ben strutturato dal momento che non presenta campi vuoti,

inconsistenze o problemi di comprensibilità. La licenza per l'utilizzo dei dati non è specificata alla sorgente, quindi deduco che l'utilizzo sia libero.

Per garantire all'utente di poter memorizzare le proprie informazioni, si intende creare una tabella utenti contenente le informazioni degli utenti registrati (l'applicazione sorvolerà la registrazione di nuovi utenti, essendo poco significativa agli scopi di questa prova).

1.6 Descrizione preliminare degli algoritmi coinvolti

L'applicazione viene realizzata in linguaggio Java e segue i pattern MVC (Model View Controller) e DAO (Data Access Object) per gestire la separazione tra interfaccia utente, logica applicativa e accesso ai dati. L'interfaccia utente disporrà di due sezioni:

Sezione di selezione manuale dei singoli corsi:

Questa sezione prevede semplici operazioni di lettura e aggiornamento di mappe e liste (l'aggiunta di un corso al piano formativo implica la rimozione del corso dalla lista dei futuri corsi acquistabili, ad esempio) e scrittura di query di complessità variabile a seconda dei criteri di filtraggio dei corsi selezionati dall'utente. Per la stampa delle statistiche descrittive dei corsi che popolano il piano, lascerò interagire il software con alcuni grafici a disposizione su Scene Builder.

Sezione di generazione del piano formativo:

Alla pressione del bottone "Genera", il software applicherà al dataset i filtri selezionati dall'utente e genererà un grafo semplice, orientato e non pesato, i cui nodi saranno i corsi a catalogo e gli archi connetteranno ciascun corso a tutti i corsi di difficoltà pari o immediatamente superiore. In questo modo:

corsi *All Levels* ---> corsi *All levels* e *Beginner Level*
corsi *Beginner Level* ----> corsi *Beginner Level* e corsi *Intermediate Level*
corsi *Intermediate Level* ---> corsi *Intermediate Level* e corsi *Expert Level*
corsi *Expert Level* ---> corsi *Expert Level*

Per mezzo di un algoritmo ricorsivo, il software implementerà il tipico "Knapsack Problem" per l'individuazione del piano formativo che risolve uno o più criteri di ottimizzazione secondo una gerarchia a scelta dell'utente, si compone di corsi di difficoltà gradualmente crescente (più precisamente, non decrescente) e infine rispetta i vincoli di tempo e denaro dichiarati dall'utente.

1.7 Descrizione preliminare delle funzionalità previste per l’applicazione software

Sezione di selezione manuale dei singoli corsi:

In questa sezione dell'interfaccia, l'utente potrà scegliere i filtri e popolare manualmente il proprio piano. Inoltre per mezzo di un grafico a torta (o altro, da definire) visualizzare la multidisciplinarietà dei corsi scelti, oppure la variabilità nella difficoltà dei corsi, oppure ancora le fasce di prezzo.

Sezione di generazione del piano formativo:
Per l'implementazione dell'algoritmo ricorsivo, all'utente verrà chiesto di indicare:

1. La/le materia/e che vuole studiare; (opzionale)
2. Una parola chiave per filtrare la ricerca per argomento (con la funzione contains() di Java o con LIKE "%string%" di SQL sul titolo del corso). Questa scelta deriva dall'assunzione che una stessa chiave di ricerca (es: "Pattern") possa tirar fuori argomenti di materie diverse, incrementando la multidisciplinarietà (Pattern musicali, finanziari, di programmazione, di design). (opzionale)
3. Le dimensioni complessive (in dollari) del portafoglio che intende mettere a disposizione per il proprio piano formativo. (obbligatorio)
4. Il tempo totale (in ore) che intende dedicare all'apprendimento (obbligatorio).
5. Criterio di ottimizzazione tra:
    - massimizzare il numero di corsi
    - massimizzare il rating medio dei corsi
    - massimizzare la popolarità dei corsi

Capitolo 2 – Descrizione dettagliata del problema affrontato

2.1 Introduzione al contesto operativo/aziendale

Nell'era della digitalizzazione, l'istruzione e la formazione professionale si confrontano con la crescente necessità di accessibilità e personalizzazione. Il contesto aziendale attuale richiede soluzioni che riescano a sopperire alla carenza di risorse quali tempo e capitale, fornendo al contempo un percorso formativo mirato e flessibile. Tale esigenza si manifesta in maniera accentuata nel settore del Remote Learning, dove la vastità e la diversificazione dell'offerta formativa rendono la selezione di corsi adeguati una sfida complessa per gli utenti.

Il problema specifico affrontato dal nostro software si radica nella difficoltà di ottimizzare la scelta dei percorsi didattici in funzione delle esigenze personali, professionali e finanziarie degli individui. Il software si colloca in un contesto operativo dove le organizzazioni educative sono chiamate a confrontarsi con la transizione verso l'e-learning e la formazione a distanza. Questa transizione, accelerata da emergenze globali come la pandemia di COVID-19, ha messo in evidenza sia la necessità di metodologie formative adattabili e scalabili sia l'opportunità di raggiungere un pubblico più ampio attraverso l'utilizzo di piattaforme digitali.

Di conseguenza, la necessità è quella di sviluppare un tool che non solo faciliti la navigazione e la selezione dei corsi, ma che lo faccia attraverso un algoritmo capace di personalizzare il piano formativo basandosi sui parametri di tempo, budget, obiettivi formativi e preferenze dell'utente.

2.2 Analisi del problema affrontato

<u>Introduzione alle funzionalità:</u>
Il sotto-problema che il tool si propone di risolvere è l'estrazione e la manipolazione di dati eterogenei per la generazione automatica di un piano formativo di difficoltà incrementale che massimizzi tre parametri secondo una gerarchia scelta dall'utente e rispetti i sopracitati vincoli di tempo e denaro.

<u>Input:</u>
Gli input consistono in variabili che fungono da filtro quali il prezzo, la materia di studio, l'argomento (come stringa contenuta nel titolo, in mancanza di dati più granulari relativi ai contenuti dei corsi), il livello di difficoltà e la durata dei corsi.

Output:
Il programma genera un piano di studi personalizzato e statistiche descrittive che supportano l'utente nel confronto tra piani formativi diversi.

Scelte implementative e criticità:
Per non imporre margini troppo stringenti alla personalizzazione, è stata presa la decisione strategica di non imporre alcun filtro come mandatorio per il funzionamento dell'applicazione: questa scelta, d'altro canto, comporta la creazione di un grafo grande e denso, che combinato alla complessità del problema multi-obiettivo affrontato con la ricorsione, potrebbe richiedere all'applicazione un tempo eccessivamente lungo prima che questa converga ad una soluzione. Pertanto, è fortemente consigliata l'applicazione del maggior numero di filtri.

Potenzialità:
La potenzialità dell'approccio risiede nella capacità di offrire una soluzione scalabile e personalizzata (i dati sono prelevati da Udemy, ma l'algoritmo potrebbe essere applicato a qualunque piattaforma di remote learning).

Rilevanza e obiettivi:
Il software è inoltre progettato per arricchire l'esperienza utente nell'esplorazione e selezione dei corsi, consentendo un'applicazione di filtri che segue una logica graduale. Questo permette all'utente di effettuare scelte mirate e confrontare piani senza la necessità di ripartire da zero, agevolando un processo di selezione intuitivo e flessibile.

Capitolo 3 – Descrizione del dataset

Il dataset ***udemy*** è costituito da tre tabelle:

- una tabella ***corsi*** proveniente dalla piattaforma Kaggle alla quale non sono state apportate modifiche (vedi 1.5);
- una tabella ***utenti*** creata con lo scopo di permettere un utilizzo reiterato e multiutente dell'applicazione attraverso l'autenticazione.
- una tabella ***acquisti*** inizialmente vuota e popolata dinamicamente durante l'utilizzo.

La tabella **corsi** ha la stessa struttura descritta nel paragrafo 1.5. Non ha richiesto ulteriori operazioni di pulizia, se non il ridimensionamento delle variabili in HeidiSQL.

Ho creato la tabella **utenti** per tenere traccia dei piani formativi acquistati dall'utente e per simulare l'interazione con una piattaforma di e-learning.

campo	***descrizione***	***variabile***
nome	Nome dell'utente	VARCHAR
***co*gnome**	Cognome dell'utente	VARCHAR
***u*sername**	Username dell'utente per l'autenticazione (chiave primaria)	VARCHAR
password	Password per l'autenticazione	VARCHAR
portafogli	Budget con cui l'utente si è registrato	DOUBLE (dollari)

La tabella acquisti è la relazione che lega l'entità Utente all'entità Corso.

campo	***descrizione***	***variabile***
couse_id	Chiave primaria della tabella **corsi**	INT
username	Chiave primaria della tabella **utenti**	VARCHAR

L'utilizzo delle due tabelle che ho aggiunto non è funzionale agli algoritmi di cui il programma si compone: è utile però a garantire che l'applicazione sia sempre in grado di restituire all'utente i progressi effettuati attraverso gli acquisti precedenti.

Capitolo 4 – Descrizione delle strutture dati e degli algoritmi coinvolti

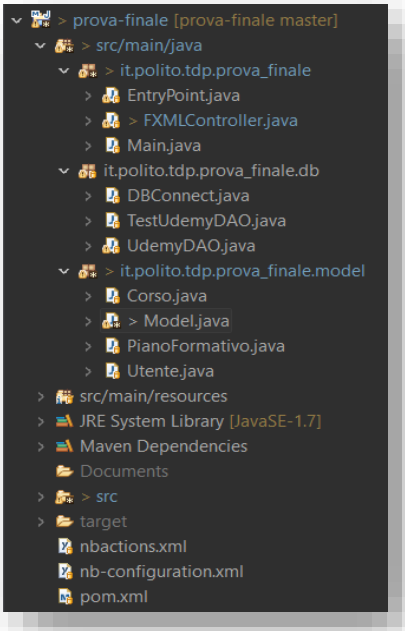


***Figura 1** - Pattern MVC e DAO*

Il software rispetta i patterns di programmazione MVC e DAO.

- Il package *it.polito.tdp.prova-finale.db* contiene la classe principale del progetto, Main, la classe EntryPoint, utilizzata per l'avvio del programma e la classe FXMLController, che collega il Model all'interfaccia grafica.
- Il package *it.polito.tdp.prova-finale.db* contiene le classi necessarie per l'accesso ai dati presenti nel database: DBConnect e UdemyDAO.
- Il package *it.polito.tdp.prova-finale.model* contiene la classe logica principale dell'applicazione (Model) e le classi di oggetti **Utente**, **Corso**, **PianoFormativo** create per il funzionamento dell'applicazione.

Il Model viene inizializzato popolando tre *mappe* principali: mappaUtenti, mappaCorsi, mappaLivelli (che associa ad ogni stringa di livello un numero compreso tra 0 e 3, dove 0=All Levels e 3=Expert Level, per facilitare l'ordinamento)

La funzione **getCatalogo**( ) è progettata per recuperare dati da un database di corsi, costruendo dinamicamente una query SQL basata sui parametri forniti dall'utente (che possono essere omessi). La funzione accetta quattro parametri: `livello`, `materia`, `costo` e `search`, e aggiunge condizioni aggiuntive alla query SQL originale se i parametri `livello` e `materia` sono forniti, utilizzando un **StringBuilder** per costruire la query SQL e un ArrayList per tenere traccia dei valori dei parametri che devono essere inseriti nella query preparata.
e restituisce una mappa dei corsi trovati. La costruzione dinamica della query consente di generare interrogazioni specifiche in base alle necessità dell'utente, rendendo la funzione flessibile e riutilizzabile per diversi tipi di ricerche all'interno del database dei corsi.

Il metodo **genera**( ) dà avvio all'algoritmo di ricorsione, inizializzando le liste necessarie ai suoi controlli compresa la lista di variabili enum Parametro che sarà utilizzata per l'elezione del miglior piano. Una serie di metodi accessori di tipo **calcolaX**( ) popolano gli attributi dell'oggetto PianoFormativo.

La funzione **ricorsione**( ) sfrutta un algoritmo di ricerca ricorsivo che trova il piano formativo ottimale esaminando tutte le possibili combinazioni di corsi in un grafo di opzioni formative, rispettando i vincoli di budget e tempo. L'algoritmo utilizza un approccio di ottimizzazione multi-parametro per confrontare i piani formativi, dando priorità ai parametri secondo una

gerarchia definita dall'utente (ad esempio, 1-numero di corsi, 2-rating medio, 3-popolarità). In ogni passo ricorsivo, valuta se l'aggiunta di un corso porti a un piano migliore attraverso la funzione **isBetter**( ); se lo fa, aggiorna il miglior piano corrente e procede all'esplorazione di nuovi percorsi, altrimenti esclude tale corso e ritorna indietro (*backtracking*) per testare alternative. La natura ricorsiva e l'esplorazione esaustiva dello spazio di ricerca rendono l'algoritmo potenzialmente esponenziale nel peggiore dei casi, poiché valuta sistematicamente ogni combinazione possibile entro i vincoli imposti.

```
private void ricorsione(Integer livello, List<Corso> adiacenti, double tempo,
        double budgetMax, PianoFormativo parziale) {
    double durataParziale = calcolaTempoParziale(parziale);
    double costoParziale = calcolaCostoParziale(parziale);
    double ratingMedio = calcolaRatingMedio(parziale);
    int popolarita = calcolaPopolarita(parziale);

// se la soluzione corrente è migliore della precedente, aggiorno il piano migliore
    if (isBetter(parziale, this.bestPFP, gerarchiaParametri)
            && durataParziale <= tempo && costoParziale <= budgetMax) {
        this.bestPFP = new PianoFormativo(new ArrayList<Corso>(parziale.getCorsi()), costoParziale,
                durataParziale, ratingMedio, popolarita);
    }

 // Verifico se siamo ancora entro i vincoli di tempo e budget
    if(durataParziale < tempo && costoParziale < this.user.getBudget()) {
        for (Corso c: adiacenti) {
            if(!parziale.getCorsi().contains(c)) {
                double tempoConC = durataParziale + c.getDuration();
                double costoConC = costoParziale + c.getPrice();

                // Verifico se aggiungendo il corso si superano i limiti
                if(tempoConC <= tempo && costoConC <= this.user.getBudget()) {
                    parziale.getCorsi().add(c);
                    parziale.setDurataTotale(calcolaTempoParziale(parziale));
                    parziale.setCostoTotale(calcolaCostoParziale(parziale));
                    parziale.setRatingMedio(calcolaRatingMedio(parziale));
                    parziale.setPopolaritaTotale(calcolaPopolarita(parziale));
                    Integer nextLevel = this.mappaLivelli.get(c.getLevel()) > livello
                            ? this.mappaLivelli.get(c.getLevel()) : livello;
                    List<Corso> successivi = new ArrayList<Corso>(Graphs.successorListOf(this.grafo, c));
                    ricorsione(nextLevel, successivi, tempo, budgetMax, parziale);

                    parziale.getCorsi().remove(c); // backtracking
                }
            }
        }
    }
}
```

Figura 2 *- Metodo ricorsione( ) per la risoluzione del problema di ottimizzazione*

La funzione **isBetter**() è un metodo di confronto che determina se un **PianoFormativo** (indicato come `current`) è migliore di un altro (indicato come `best`), basandosi su una lista di parametri data (`gerarchiaParametri`). Questi parametri hanno un ordine gerarchico, il che significa che il primo parametro nella lista ha la priorità più alta nel confronto.
Il metodo itera attraverso ciascun parametro nella lista gerarchica e confronta i corrispondenti valori tra le due istanze di **PianoFormativo**. Se trova una differenza in uno dei parametri, ritorna *true* o *false* a seconda che il **current** sia migliore del **best** secondo quel

parametro. Se i valori sono uguali, continua a confrontare i successivi parametri nell'ordine dato fino a trovare una differenza o fino a esaurire i parametri.

```java
private boolean isBetter(PianoFormativo current, PianoFormativo best, List<Parametro> gerarchiaParametri) {
    for (Parametro parametro : gerarchiaParametri) {
        switch (parametro) {
            case NUMERO_CORSI:
                if (current.getCorsi().size() != best.getCorsi().size()) {
                    return current.getCorsi().size() > best.getCorsi().size();
                }
                break;
            case RATING_MEDIO:
                if (current.getRatingMedio() != best.getRatingMedio()) {
                    return current.getRatingMedio() > best.getRatingMedio();
                }
                break;
            case POPOLARITA:
                if (current.getPopolaritaTotale() != best.getPopolaritaTotale()) {
                    return current.getPopolaritaTotale() > best.getPopolaritaTotale();
                }
                break;
        }
    }
    return false;
}
```

Figura 3 *- Metodo isBetter() per il confronto tra piani formativi*

Tra i metodi più importanti per la corretta gestione dell'accesso, abbiamo **checkRegistrazione**(), che restituisce al Controller un array di variabili booleane che fungono da flag per la correttezza dei diversi campi del form. Questa struttura dati mi ha permesso di controllare in maniera più precisa l'interazione con l'utente (e la proiezione dei messaggi di errore) e impedire l'iniezione di dati errati nel database.

Capitolo 5 - Diagramma delle classi delle parti principali

Figura 4 *- Diagramma delle classi del package model generato con StarUML*

Capitolo 6 - Esempi di utilizzo dell'applicazione

Poiché l'applicazione fornisce elaborazioni diverse a seconda dei dati memorizzati in precedenza dall'utente (non include nel piano formativo o nella wishlist corsi già acquistati), affinché gli esempi presentati in questa sezione siano facilmente riproducibili, ci poniamo nella situazione di un utente appena registrato, ossia che non ha ancora effettuato acquisti. Inoltre, tralascerò di descrivere esempi relativi all'autenticazione e registrazione degli utenti perché secondari ai fini della descrizione degli algoritmi.

6.1 Esempio di acquisto di singoli corsi

Per utilizzare il tool di acquisto, non è necessario applicare filtri: questi restano a disposizione dell'utente per raffinare la sua ricerca. L'utente seleziona dalla tabella Catalogo Corsi la riga relativa al corso desiderato e, pigiando sul tasto Aggiungi alla lista +, questo verrà messo in coda alla Wishlist. L'utente può inserire qui tutti i corsi che desidera, eccetto quelli che ha già comprato.

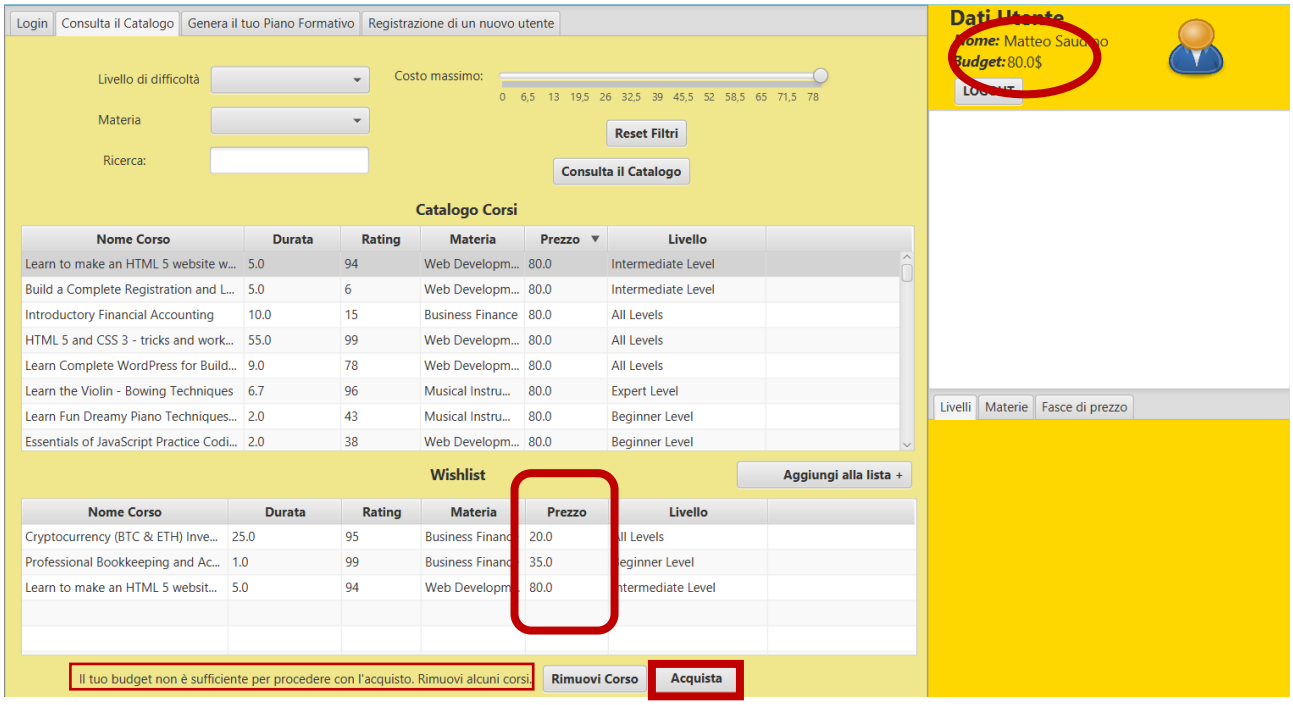


Figura 5 *- Esempio acquisto da catalogo*

Quando, dopo aver stilato la propria lista di preferiti, cliccherà sul tasto Acquista, se il costo di tutti i corsi presenti sarà superiore al budget dell'utente, l'acquisto non andrà a buon fine e l'interfaccia segnalerà all'utente che dovrà rinunciare ad alcuni dei corsi.

Viceversa, se il budget è sufficiente, la Wishlist verrà aggiunta alla lista di corsi acquistati e il budget decrementato.

6.2 Esempio di acquisto di un piano formativo

I corsi acquistati dall'utente nella sezione Catalogo Corsi sono sempre sincronizzati affinché il grafo generato per la ricerca ricorsiva non esplori mai corsi già comprati dall'utente.

Selezionato un livello di partenza (obbligatorio!), una materia, l'argomento, il quantitativo di ore e giorni di studio e il budget spendibile, cliccando sul tasto Genera sarà creato un grafo semplice e orientato che ha per nodi i corsi che rispettano i filtri e un arco per ogni corso di livello n verso tutti gli altri corsi di livello n e di livello n+1. Come si può immaginare, questo genera un grafo molto denso: è per questo che è necessario essere specifici con le proprie preferenze per permettere all'algoritmo di risolversi in poco tempo. Anche inserendo tutti i filtri, bisogna ugualmente prestare attenzione agli argomenti troppo vasti e generici che partono da un livello basso: **Inserendo Materia = Musical Instruments, Livello = All Levels e Ricerca = "guitar", con un budget di 200$, 3 x 8 ore, il grafo conta ben 223 nodi e 30400 archi!!!.**

Anche solo decrementando il budget, si può ottenere con gli stessi dati un piano molto più rapidamente ed effettuare l'acquisto col quale saranno aggiornati i dati utente.

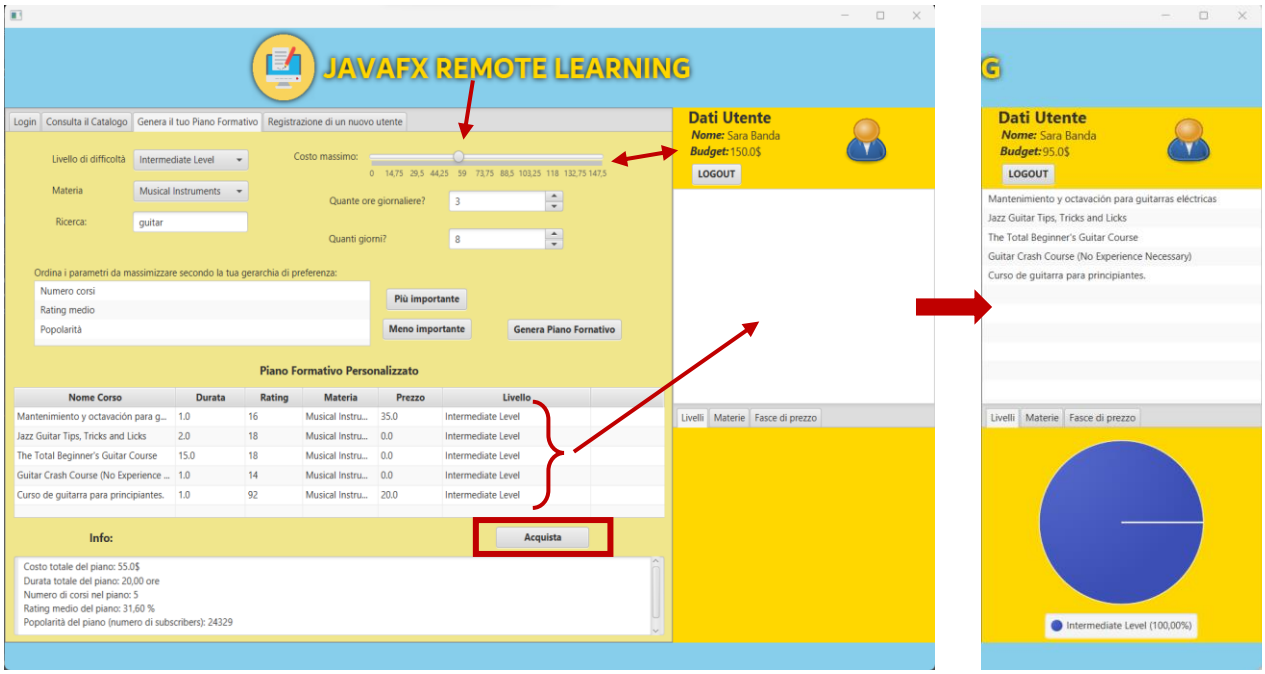
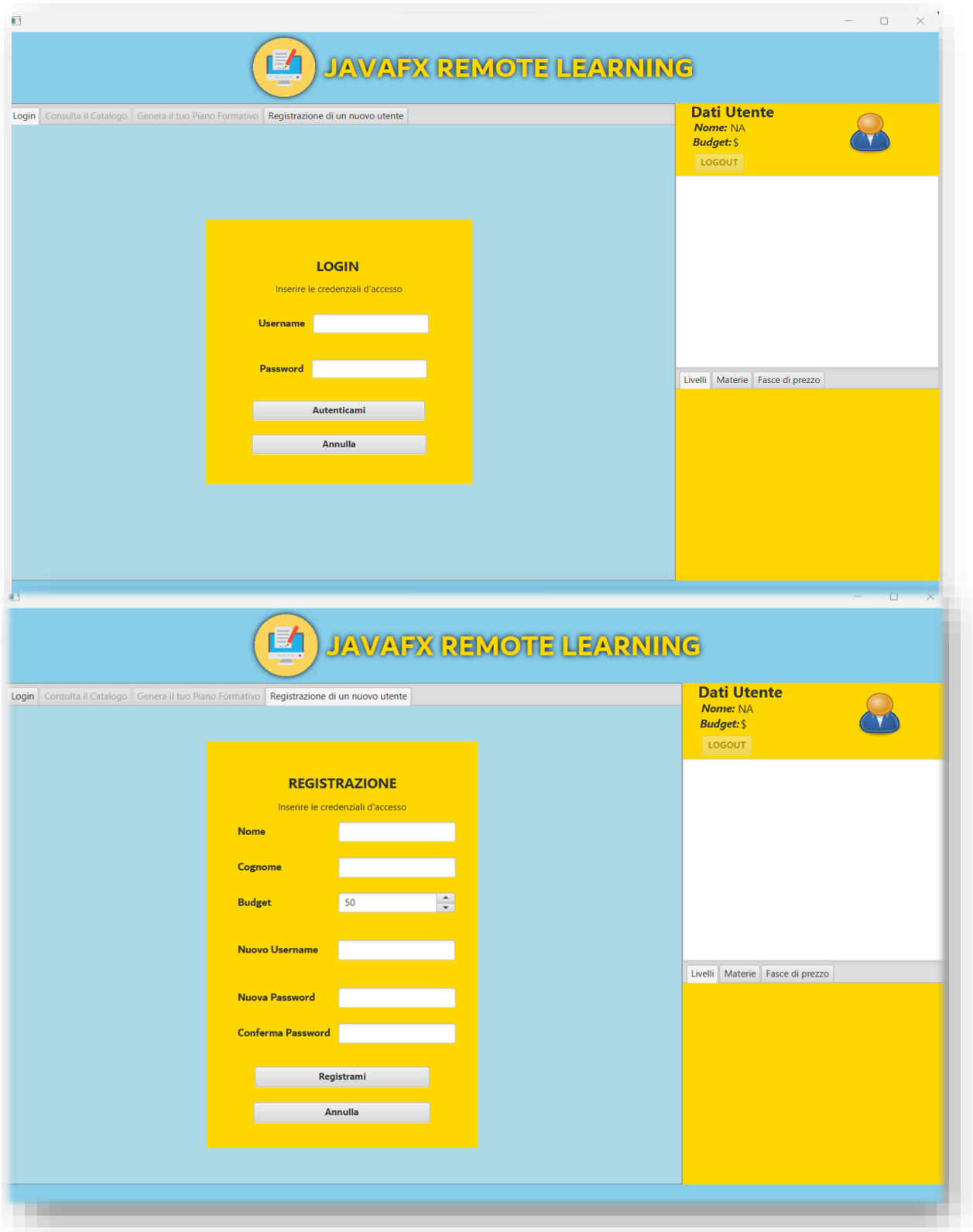


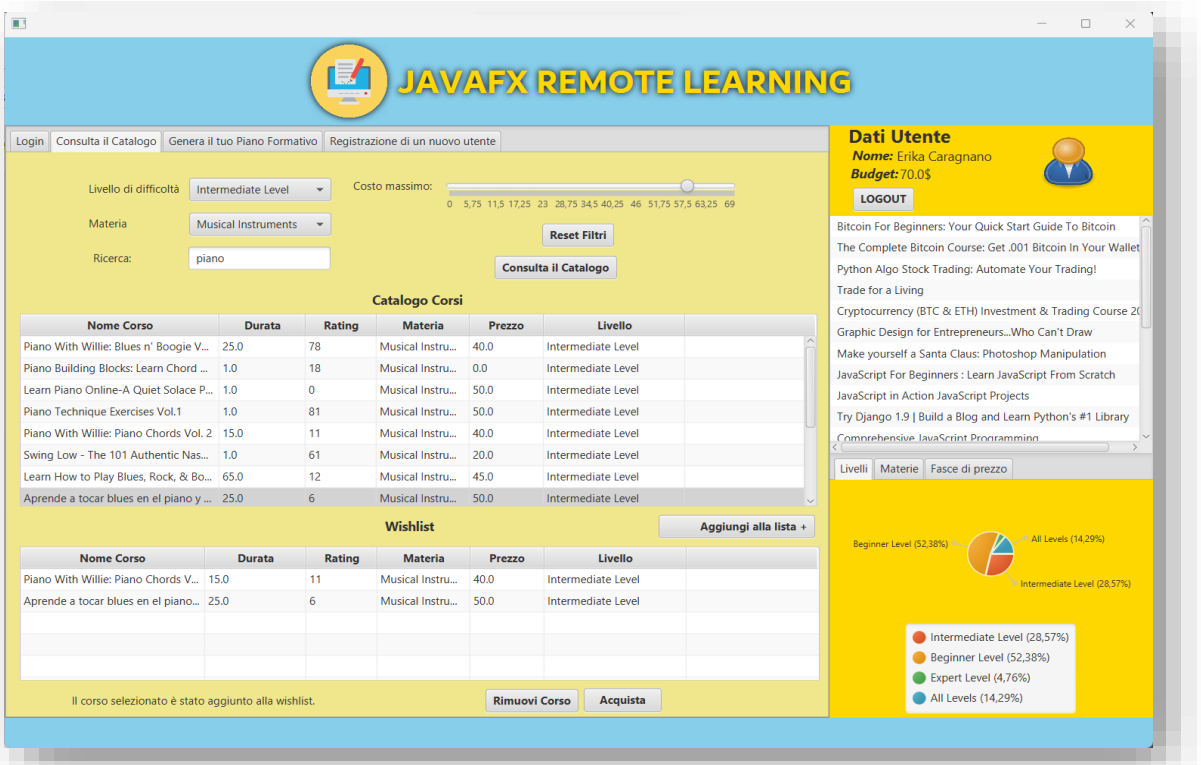
Figura 6 *- Esempio acquisto da generatore*

Capitolo 7 – Videata dell'applicazione

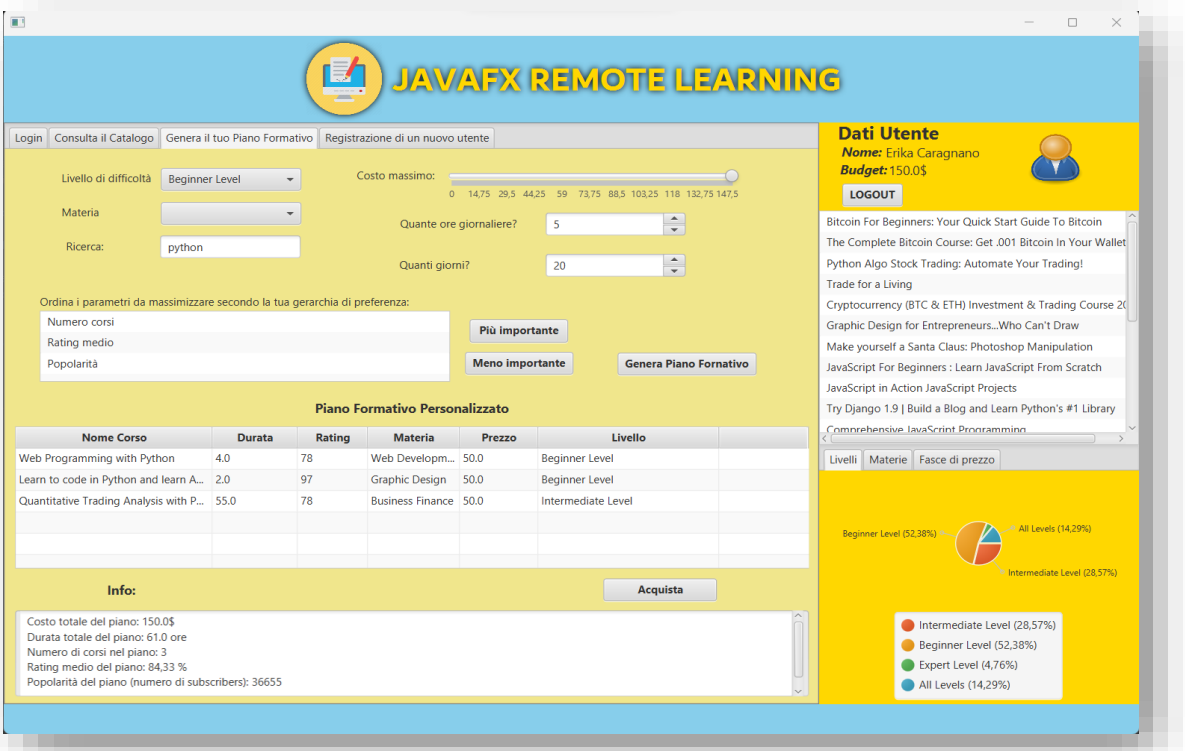
7.1 Tab *Login* e Tab *Registrazione di un nuovo utente*



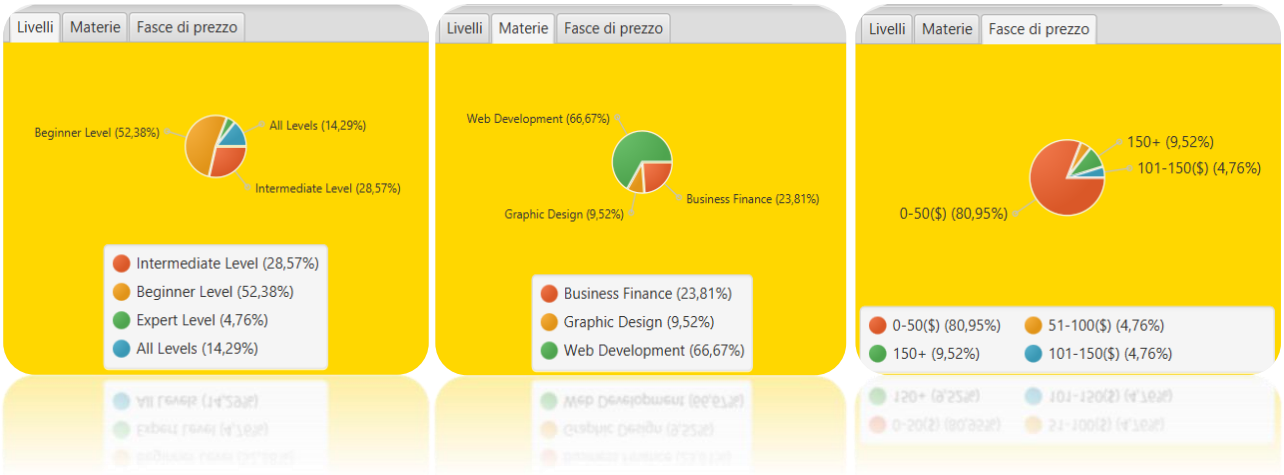
7.2 Tab *Consulta il Catalogo* e colonna Dati Utente



7.3 Tab *Genera il tuo Piano Formativo*



7.4 Multidisciplinarietà, progressi e fasce di prezzo



7.5 Video di presentazione del funzionamento del tool

Video dimostrativo sull'uso dell'applicazione disponibile al link: https://youtu.be/VgKWqBVGU00?si=k41sm-iyntvPFycR

Capitolo 8 – Risultati sperimentali ottenuti

I piani formativi generati dal software hanno, come previsto, una logica di difficoltà non decrementale. Questa caratteristica deriva direttamente dalla conformazione del nostro grafo: gli archi orientati da corsi di livello inferiore a corsi di livello equivalente o superiore e il controllo che il primo corso della lista sia di livello pari al livello minimo impostato dall'utente sono la chiave per il mantenimento del trend.

La configurazione del piano cambia significativamente con diverse combinazioni di filtri, mostrando una sensibilità particolare all'ordine in cui vengono inseriti i parametri come 'Popolarità', 'Rating medio' e 'Numero corsi'**. In un problema di ottimizzazione multi-parametro, la massimizzazione di un parametro può entrare in conflitto con altri.** Ad esempio, se l'utente prioritizza il 'Rating', un solo corso con la valutazione più alta è sufficiente per massimizzarlo. L'aggiunta di più corsi potrebbe abbassare la valutazione media. Pertanto, ottenere un piano con più di un corso è raro se il 'Rating' è la massima priorità. Se si selezionano filtri più specifici, il piano rispetta sempre il budget dell'utente, garantendo la possibilità di acquisto.

Può essere interessante per l'utente confrontare piani diversi al variare della gerarchia di massimizzazione dei parametri: se all'esempio presentato nel paragrafo 6.2 precedente applicassimo un livello di partenza Intermediate e tenessimo tutti i filtri uguali **facendo variare soltanto lo slider del budget**, man mano che quest'ultimo aumenta, **aumentano anche le prestazioni dei singoli parametri in ordine di gerarchia (Numero corsi -> Rating -> Popolarità):**

Se Budget == 30$

Info:

Costo totale del piano: 20.0$
Durata totale del piano: 4,00 ore
Numero di corsi nel piano: 3
Rating medio del piano: 41,33 %
Popolarità del piano (numero di subscribers): 15124

Se budget == 60$

Info:

Costo totale del piano: 55.0$
Durata totale del piano: 5,00 ore
Numero di corsi nel piano: 4
Rating medio del piano: 35,00 %
Popolarità del piano (numero di subscribers): 15136

Se Budget == 90$

Info:

Costo totale del piano: 70.0$
Durata totale del piano: 9,00 ore
Numero di corsi nel piano: 4
Rating medio del piano: 54,50 %
Popolarità del piano (numero di subscribers): 25993

Se Budget == 120$

Info:

Costo totale del piano: 115.0$
Durata totale del piano: 11,00 ore
Numero di corsi nel piano: 5
Rating medio del piano: 53,40 %
Popolarità del piano (numero di subscribers): 26171

Figura 7 *- Andamento delle caratteristiche del piano al variare degli input*

Capitolo 9 – Valutazioni sui risultati ottenuti e considerazioni conclusive

I piani formativi generati da JavaFX Remote Learning non hanno la pretesa di garantire propedeuticità rispetto ai contenuti dei corsi, né entrano del merito dello spettro di nozioni fornite nel complesso. Questa forte limitazione alle potenzialità dei piani generati è da imputare all'assenza di dati adeguati: l'unico dato che permetta una ricerca testuale per argomento è il titolo.

Al netto di ciò, le potenzialità dell'algoritmo restano evidenti: non ci sono limiti alla tipologia e al numero di parametri sulla base dei quali poter confrontare la bontà di due piani formativi, rendendo questo uno strumento utile e personalizzabile. Risulta essere una criticità, al contrario, la complessità computazionale delle operazioni svolte, che espone l'utente al rischio di aspettare molti minuti prima che una soluzione venga proiettata, se non ha una adeguata comprensione del catalogo e dei suoi filtri di ricerca.

Nell'ottica, dunque, di un potenziamento dell'applicazione, mi piacerebbe affiancare ad esso un dataset più corposo e un algoritmo di text mining che possa indirizzare la ricerca verso piani contenutisticamente più coerenti.

Riferimenti alle figure

Sitografia

Immagine di copertina del video:
https://www.google.com/url?sa=i&url=https%3A%2F%2Fwww.edweek.org%2Ftechnology%2Fopinion-why-are-we-turning-our-backs-on-remote-learning%2F2021%2F06&psig=AOvVaw3gX5d_liONEqLjQVTmYHmr&ust=1699696941710000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCICD9NWWuYIDFQAAAAAdAAAAABAH

Licenza