

*Università degli studi di Salerno*

*Dipartimento di Informatica*

*Corso di Laurea in Informatica*

**FONDAMENTI DI INTELLIGENZA  
ARTIFICIALE**

*Documentazione progetto*

***“UniDates”***

**Docenti:**

*Fabio Palomba*

*Fabio Narducci*

**Studenti:**

Nome

Matricola

**Alessandro Pio D'Antuono**

**05700**

**Raimondo Rapacciuolo**

**05778**

**Antonio Fasulo**

**05904**

*Anno accademico 2020/2021*

# *Indice:*

- 1. Obiettivo del sistema*
- 2. Specifica dell'ambiente*
  - 2.1. Performance Measure*
  - 2.2. Enviroment*
  - 2.3. Actuators*
  - 2.4. Sensors*
- 3. Proprietà dell'ambiente*
- 4. Dati a disposizione*
  - 4.1. Dataset originale*
  - 4.2. Rimozione degli attributi ridondanti*
  - 4.3. Separazione dell'attributo "Quali sono i tuoi interessi?"*
  - 4.4. Specifica dell'età e dell'altezza*
  - 4.5. Dataset Finale*
- 5. Contestualizzazione del problema*
- 6. Scelta del software*
- 7. Scelte implementative*
  - 7.1. Scelta dell'algoritmo*
  - 7.2. Scelta di K*
  - 7.3. Scelta degli utenti da suggerire*
  - 7.4. Suggerimenti agli utenti*
- 8. Panoramica del sistema*
- 9. Esempio di utilizzo*

# 1. Obiettivo del Sistema

L'obiettivo del nostro sistema è quello di fornire la possibilità a studenti universitari di conoscersi nonostante l'emergenza COVID-19. Attraverso la nostra piattaforma questi potranno creare un proprio profilo con alcune informazioni personali e scegliendo una lista di "topic" di interesse (Arte, Musica, Sport, etc..).

Il nostro sistema garantirà la possibilità di visualizzare la foto profilo di altri studenti ed un loro numero ridotto di informazioni personali.

Quando due studenti avranno messo "Mi piace" reciprocamente alla foto profilo dell'altro, tra di loro avverrà un "Match" e questi ultimi potranno vedere completamente l'uno il profilo dell'altro, contenente tutte le informazioni, i contatti personali e tutte le foto inserite. In futuro sarà permesso a questi ultimi di comunicare attraverso un sistema di Chat.

Nello specifico, il modulo di Intelligenza Artificiale consiste nell'implementazione di un sistema di profilazione degli studenti, che andrebbe a consigliare ad uno studente, una serie di altri studenti che potrebbero interessargli in base alcune informazioni del proprio profilo (Età, Sesso, Altezza, Interessi, Topic di interesse).

## 2. Specifica dell'ambiente

Questa sezione fornisce una descrizione della rappresentazione schematica dell'ambiente in esame e nel dettaglio dei 4 indicatori **PEAS**: *Performance Measure, Environment, Actuators, Sensors*.

### 2.1 Performance Measure

L'obiettivo del sistema è quello di suggerire ad uno studente, in base alle caratteristiche del proprio profilo, i profili degli studenti che più potrebbero interessargli.

Pertanto la misura di performance si basa sull'accuratezza delle informazioni personali inserite dall'utente (Età, Sesso, Altezza, Interessi) e su quanto la lista di topic rappresenta lo studente. Quanto più il raggruppamento degli studenti risulta preciso, tanto più sarà probabile che gli studenti suggeriti interessino allo studente finale.

### 2.2 Environment

Gli oggetti di cui l'ambiente è costituito sono gli studenti che fanno uso del sistema.

Ad ogni studente è collegato un profilo sul quale è possibile inserire le proprie informazioni e topic di interesse.

### 2.3 Actuators

Gli attuatori del sistema sono rappresentati dalle operazioni che vengono svolte lato server dal programma che, elaborando le scelte effettuate dallo studente, calcola gli studenti suggeriti e li restituisce in output.

### 2.4 Sensors

I sensori attraverso i quali il sistema riceve gli input percettivi sono le operazioni che lo studente effettua sull'interfaccia grafica della nostra applicazione, mediante le quali lo studente può inserire i propri topic e le informazioni personali.

### 3. Proprietà dell'ambiente

- **Completamente Osservabile:** i sensori dell'ambiente danno accesso in ogni momento, allo stato completo dell'ambiente. Tutti gli studenti sono accessibili con i loro profili (contenenti le informazioni personali).
- **Deterministico:** lo stato successivo dell'ambiente è determinato dallo stato corrente e dall'azione eseguita dall'agente. I profili suggeriti variano infatti sulla base di quelle che sono le informazioni personali inserite dall'utente e la sua lista dei topic.
- **Episodico:** Le azioni dell'agente vengono eseguite tenendo in considerazione solo le percezioni istantanee del singolo episodio (quali le informazioni personali dello studente, presenti in quel momento sul profilo).
- **Statico:** L'ambiente risulta statico in quanto l'agente opera su un ambiente che durante l'esecuzione non subisce alcun cambiamento.
- **Discreto:** l'ambiente prevede un numero limitato di percezioni e azioni distinte e definite, che corrispondono alle modifiche che questo può effettuare alle informazioni del proprio profilo.
- **Agente Singolo:** l'ambiente prevede un singolo agente che opera su di esso. Non sono previste situazioni in cui due o più agenti cooperano o competono tra di loro.

## 4. Dati a disposizione

Per il training del sistema è stato previsto un sondaggio mediante la piattaforma “Moduli” offerta da Google. Questo sondaggio è stato distribuito attraverso diversi canali di comunicazioni tra i quali Instagram. È stato ricavato un dataset con 441 elementi, contenente per ciascuno di essi caratteristiche e dati più o meno pertinenti al contesto in esame. Sono state aggiunte alcune domande ridondanti al questionario affinché questo risultasse più piacevole nella compilazione. I dati riportati da quest’ultimo tipo di domande verranno successivamente eliminati dal dataset, in quanto non adatti alla risoluzione del problema.

Nonostante il dataset non risulti particolarmente grande, trattandosi di dati strettamente personali e quindi di difficile reperimento online per ovvi problemi logistici, è stato ritenuto comunque di dimensioni adeguate, assicurando un funzionamento discreto del sistema.

Sono state inoltre effettuate alcune operazioni sul dataset originale affinché questo risulti più adatto al problema in questione e favorisca il corretto funzionamento dell’algoritmo di Clustering.

### 1. Dataset originale:

Il dataset originale è composto da 12 attributi:

- “Sei..”
- “Sei interessato a.. (possibile scelta multipla)”
- “Quanti anni hai?”
- “Qual è il colore dei tuoi occhi?”
- “Qual è il colore dei tuoi capelli?”
- “Quanto sei alto?”
- “Studi?”
- “Se sì, frequenti..”
- “Quali sono i tuoi interessi? (possibile scelta multipla)”
- “Cosa è importante per te in una persona? (possibile scelta multipla)”
- “Quanto reputi utile un sito dating?”
- “Per cosa useresti il nostro sito di dating?”

## 2. Rimozione degli attributi ridondanti:

Come si può vedere, alcuni attributi risultano non direttamente coerenti con il problema in esame, quali:

- "Cosa è importante per te in una persona? (possibile scelta multipla)"
- "Quanto reputi utile un sito dating?"
- "Per cosa useresti il nostro sito di dating?"
- "Qual è il colore dei tuoi occhi?"
- "Qual è il colore dei tuoi capelli?"
- "Studi?"
- "Se sì, frequenti.."

Questi verranno rimossi direttamente dal dataset per favore un funzionamento coerente dell'algoritmo di Cluster.

Gli attributi "Sei.." e "Sei interessato a.." (possibile scelta multipla)" nonostante risultino inerenti al problema, verranno comunque rimossi dato che le preferenze dell'utente in termini sessuali, verranno utilizzate soltanto per scegliere gli utenti risultanti dal clustering avvenuto.

## 3. Separazione dell'attributo ***"Quali sono i tuoi interessi? (possibile scelta multipla)"***:

Questo attributo nella sua forma originale si presenta come una stringa contenente una lista dei topic di interesse dello studente. Esistono diversi modi affinché una lista venga correttamente integrata in un dataset, uno di questi consiste nel trasformare ogni elemento possibile della lista in un attributo del dataset, e ove quell'attributo sia presente nella lista dello studente in questione, verrà messo il valore della cella ad 1, altrimenti a 0.

Quindi l'attributo

- "Quali sono i tuoi interessi? (possibile scelta multipla)" con possibili valori:  
Musica, Cinema, Sport, Calcio, Anime, Manga/Fumetti, Serie Tv, Arte, Teatro, Politica, Videogiochi, Tecnologia, Viaggi, Storia, Informatica, Libri, Cucina, Natura, Fotografia, Disegno, Motori, Moda, Altro.

Verrà trasformato nelle seguenti colonne:

- |                 |               |
|-----------------|---------------|
| ● Musica        | ● Tecnologia  |
| ● Cinema        | ● Viaggi      |
| ● Sport         | ● Storia      |
| ● Calcio        | ● Informatica |
| ● Anime         | ● Libri       |
| ● Manga/Fumetti | ● Cucina      |
| ● Serie Tv      | ● Natura      |
| ● Tv            | ● Fotografia  |
| ● Arte          | ● Disegno     |
| ● Teatro        | ● Motori      |
| ● Politica      | ● Moda        |
| ● Videogiochi   | ● Altro       |

Con possibili valori : 0 (non presente), 1 (presente).

#### 4. Specifica dell'età e dell'altezza:

Nel dataset originale gli attributi:

- "Quanto sei alto?"
- "Quanti anni hai?"

Presentano come possibili scelte delle fasce di età e di altezze.

*Es. "18 - 25" oppure "1,41 - 1,60m"*

Affinché il dati rappresentino il più possibile un dataset reale questi sono stati convertiti in età e altezze, corrispondenti alla scelta effettuata.

*Es. "18 - 25" convertito in 21, "1,41 - 1,60m" convertito in 155cm*



## 5. Dataset Finale:

Il dataset finale si presenta nella forma seguente:

- "Quanti anni hai?"
- "Quanto sei alto?"
- Musica
- Cinema
- Sport
- Calcio
- Anime
- Manga/Fumetti
- Serie Tv
- Tv
- Arte
- Teatro
- Politica
- Videogiochi
- Tecnologia
- Viaggi
- Storia
- Informatica
- Libri
- Cucina
- Natura
- Fotografia
- Disegno
- Motori
- Moda
- Altro

## 5. Contestualizzazione del problema

La successiva scelta da effettuare, consiste nell'associare il problema alla specifica categoria di apprendimento a cui esso appartiene. I dati a disposizione analizzati precedentemente, non presentano un'etichetta ben definita, quindi non ci permettono di definire un'interpretazione precisa di ciò che rappresentano. Il nostro obiettivo è quello di ottenere delle informazioni sui dati in modo tale da raggrupparli in cluster senza però assegnare etichette.

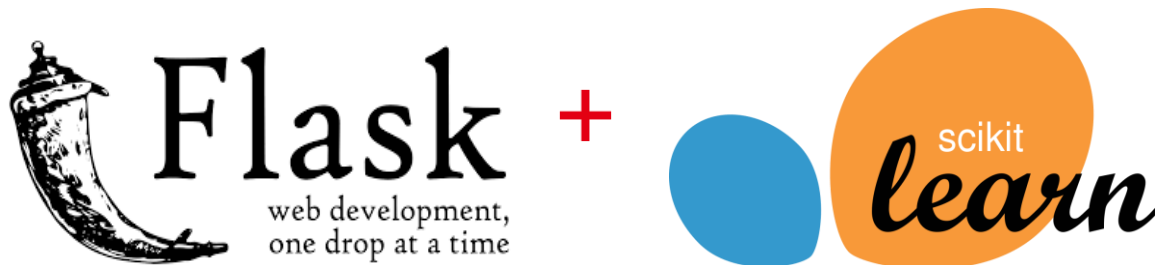
Quindi siamo in un caso di apprendimento non supervisionato, le cui tecniche ci consentono di sfruttare i dati affinché la macchina li possa elaborare e ne possa estrarre informazioni rilevanti.

Il Clustering è una tecnica esplorativa che cerca regolarità nei dati disponibili, e che consente di aggregare all'interno di gruppi (detti cluster), dati dei quali non abbiamo precedente conoscenza di appartenenza a gruppi, e che presentano caratteristiche simili tra loro.

Si tratta della tecnica più comunemente utilizzata nell'apprendimento non supervisionato, e quella che nel contesto del problema corrente risulta essere la più adeguata.

## 6. Scelta del software

Scelta la tecnica per analizzare i dati, bisogna scegliere il software con il quale viene realizzata l'implementazione. Dato che il linguaggio di programmazione più comunemente utilizzato per i problemi di machine learning, è Python, abbiamo deciso di utilizzare **scikit-learn** per implementare il modulo che analizzerà i dati e **flask** per ci permetterà di integrare questo modulo come WebService, affinché comunichi con il server Java.



*“Scikit-learn è una libreria open source di apprendimento automatico per il linguaggio di programmazione Python. Contiene algoritmi di classificazione, regressione e clustering (raggruppamento) e macchine a vettori di supporto, regressione logistica, classificatore bayesiano, k-mean e DBSCAN, ed è progettato per operare con le librerie NumPy e SciPy. scikit-learn è attualmente sponsorizzato da INRIA e talvolta da Google.”*

*“Flask è un micro-framework per Python facile da installare e da usare ed è utilizzato da compagnie come: Pinterest, LinkedIn, ed anche dal sito della campagna elettorale di Obama. Se non sapete cosa sia un framework per la creazione di applicazioni web, che ha lo scopo di facilitare la creazione di servizi web come web server, API ed altro.”*

*-Fonte: Wikipedia*

## 7. Scelte implementative

Dopo aver descritto i software utilizzati, si passa a descrivere e motivare le scelte implementative del sistema:

### 1. Scelta dell'algoritmo:

La scelta dell'algoritmo di clustering più adatto al problema è ricaduta su uno tra K-Means e DBScan. Tra i due, attraverso anche diverse osservazioni fatte sugli output di diverse esecuzioni, quello che è sembrato più adatto al problema corrente, si è rivelato il K-Means.

Il motivo principale, è sicuramente il fatto che l'algoritmo DBScan non è in grado di gestire in modo corretto dataset multidimensionali come nel caso corrente, di 26 variabili, cosa che invece riesce bene al K-Means. Sicuramente uno degli svantaggi maggiori del K-Means, risiede nella scelta dei K, ossia il numero dei cluster che dovranno essere forniti a partire dai dati a disposizione; questo numero di cluster non deve essere né troppo piccolo, altrimenti l'utilizzo dell'algoritmo perde di senso, né troppo grande altrimenti si verificherebbe *overfitting* dei dati.

### 2. Scelta di K:

Come detto in precedenza, K consiste nel numero di cluster che dovranno essere forniti a partire dai dati a disposizione.

L'algoritmo K-Means dipende fortemente dal K dato in input. Per scegliere il valore di K che potesse risultare più adeguato, abbiamo valutato iterativamente l'esecuzione del K-Means in termini di Silhouette Score (coefficiente di forma). Questa iterazione parte da K=2 fino a K=40.

Essendo inoltre, il K-Means un algoritmo di tipo *non deterministico*, la precedente valutazione eseguita più volte potrebbe riportare un numero diverso di cluster suggeriti, quindi, è stato effettuato un ulteriore processo iterativo di 5 iterazioni dove è stato scelto il K più ricorrente.

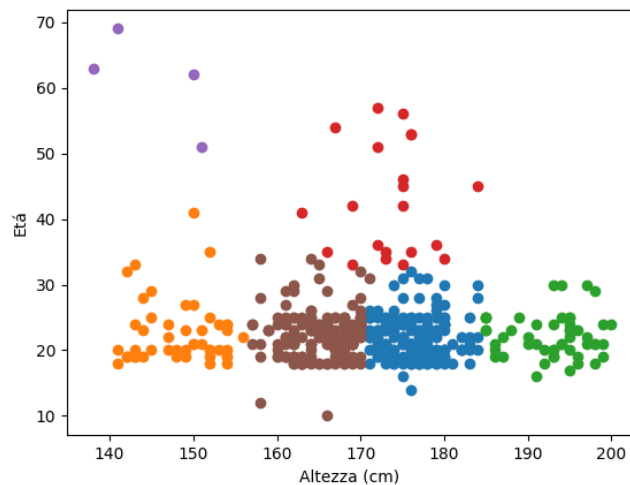
```

for i in range(5):
    silhouette_scores = []
    inertia = []

    for k in k_to_test:
        model_kmeans_k = KMeans(n_clusters=k)
        model_kmeans_k.fit(toTest.drop("cluster", axis=1))
        labels_k = model_kmeans_k.labels_
        score_k = metrics.silhouette_score(toTest.drop("cluster", axis=1), labels_k)
        silhouette_scores.append(score_k)
        inertia.append(model_kmeans_k.inertia_)
    if score_k > best_score:
        best_score = score_k
        best_number = k

```

L'esecuzione migliore avviene con K=6.



### 3. Scelta degli utenti da suggerire:

In fase di esecuzione dell'algoritmo K-Means abbiamo trascurato il sesso dello studente ed i suoi *interessi* affinché questo potesse essere eseguito con maggior coerenza:

*ad esempio l'algoritmo tenderà potenzialmente ad inserire utenti con lo stesso sesso, nel medesimo cluster, cosa che potrebbe portare a suggerimenti non efficienti.*

Verranno considerati sesso ed *interessi* nella fase di scelta della *pool* di studenti su cui effettuare l'operazione di clustering.

```
# 15 interesse, 21 sesso
# 1 = DONNE
# 0 = UOMINI
# 2 = ENTRAMBI // ALTRO

# Aggiungo gli studenti al dataset in base agli interessi dello scelto
for studente in altri_studenti:
    if (studente_scelto['studente'][15] != 2 and studente['studente'][15] != 2 and studente_scelto['studente'][15] ==
        studente['studente'][21] and studente_scelto['studente'][21] == studente['studente'][15]) or \
        (studente_scelto['studente'][15] == 2 and studente['studente'][15] != 2 and studente_scelto['studente'][21] ==
        studente['studente'][15]) or \
        (studente_scelto['studente'][15] != 2 and studente['studente'][15] == 2 and studente_scelto['studente'][15] ==
        studente['studente'][21]) or \
        (studente_scelto['studente'][15] == 2 and studente['studente'][15] == 2):
        listaId.append(studente["studente"][9])
        listaAltezze.append(studente["studente"][10])
        listaDate.append(str(studente["studente"][14])[0:4])
        listHobby.append(studente["hobbies"])
```

#### 4. Suggerimenti agli utenti:

Per rendere la *pool* di suggerimenti più variegata affinché lo studente possa interagire con un gruppo più eterogeneo di studenti, viene effettuata:

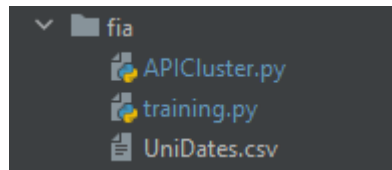
- una scelta di 15 studenti che appartengono allo stesso cluster dello studente che sta interagendo con l'agente.
- una scelta di 5 studenti che appartengono ad i due cluster più vicini a quest'ultimo.

```
# suggerisce un massimo di 15 utenti dello stesso cluster
for utente in listaUtenti:
    if utente['cluster'] == scelto['cluster'] and len(toReturn) < 15:
        toReturn.append(utente['id_profilo'])
        listaUtenti.remove(utente)
    elif len(toReturn) >= 15:
        break

# suggerisce 5 utenti di un diverso cluster
for utente in listaUtenti:
    if (utente['cluster'] == scelto['cluster'] + 1 or utente['cluster'] == scelto['cluster'] - 1) and len(toReturn) < 20:
        toReturn.append(utente['id_profilo'])
        listaUtenti.remove(utente)
    elif len(toReturn) >= 20:
        break
```

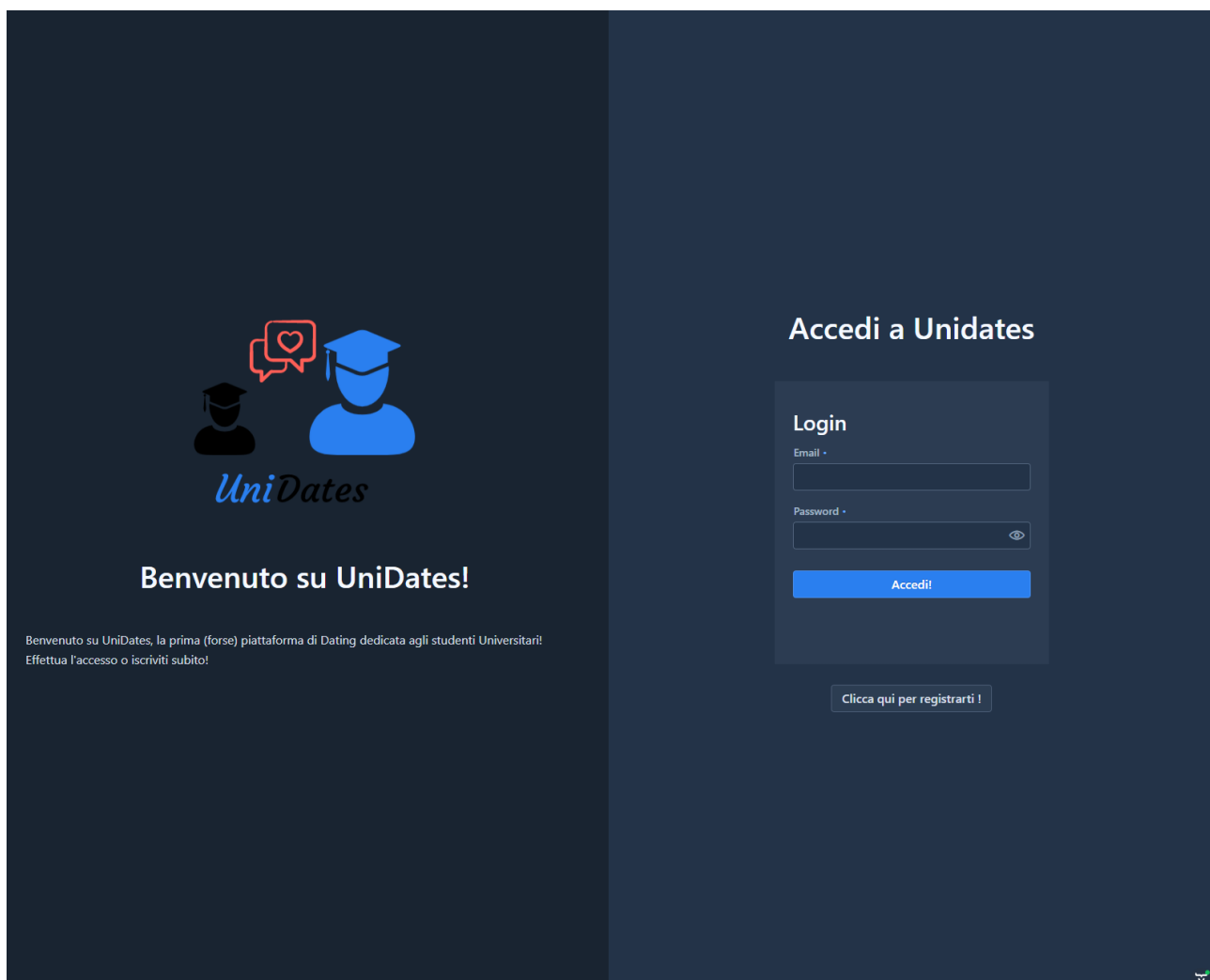
*In totale la pool di studenti suggeriti avrà dimensione 20.*


## 8. Panoramica del sistema



- *APICluster.py*: rappresenta l'implementazione del modulo attraverso APIRest (**flask**), questo effettua delle query sul database (*attraverso psycopg2*) andando a selezionare tutti gli utenti presenti sulla piattaforma. Fatto ciò eseguirà l'algoritmo K-Means su un dataset specificato nel punto 7.3 di questo documento, successivamente, restituirà una serie di *id* corrispondenti ai profili degli utenti suggeriti.
- *training.py*: rappresenta lo script che eseguirà la fase di training suggerendo quale valore di K risulta migliore in termini di coefficiente di forma.
- *UniDates.csv*: rappresenta il training-set.


## 9. Esempio di utilizzo





Inserisci email dell'utente da cercare

Marcella




Report

Interessi:  
♂

Topic:  
musica serie\_tv tv videogiochi viaggi  
storia cucina natura fotografia

Claudia




Inserisci email dell'utente da cercare

Ciao Marcella  
Profilo Personale  
Logout

♀

Topic:  
musica serie\_tv tv videogiochi viaggi  
storia cucina natura fotografia

Alessandro



Report

Interessi:  
♀

Topic:  
musica cinema anime manga serie\_tv  
tv cucina