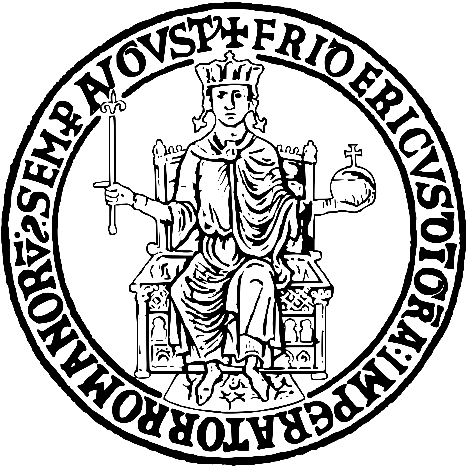
UNIVERSITÀ DEGLI STUDI DI NAPOLI “FEDERICO II”

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TRECNOLOGIE DELL’INFORMAZIONE

****

CORSO DI LAUREA IN INFORMATICA

INSEGNAMENTO DI LABORATORIO DI PROGRAMMAZIONE

ANNO ACCADEMICO 2020/2021

Sviluppo di un social network minimalista

*Autore:**Docenti:*

Crescenzo Lucio CICATIELLO Prof. Bastien MAUBERT

N86003457 Prof. Aniello MURANO

cr.cicatiello@studenti.unina.it

Sommario

[Progetto 2](#_Toc73381426)

[Approccio alla soluzione 2](#_Toc73381427)

[Traccia in breve 2](#_Toc73381428)

[Soluzione 2](#_Toc73381429)

[Funzionalità e descrizione 3](#_Toc73381430)

[Strutture dati utilizzate 4](#_Toc73381431)

[Struct data 4](#_Toc73381432)

[Struct utente 4](#_Toc73381433)

[Struct seguiti 4](#_Toc73381434)

[Esempi di esecuzione 5](#_Toc73381435)

[Seguo un utente e stampo il suo ultimo post 5](#_Toc73381436)

[Creo un post e lo mostro 5](#_Toc73381437)

[Eliminazione di un’utenza e nelle relative liste in cui è seguito 6](#_Toc73381439)

Progetto

Approccio alla soluzione

### **Traccia in breve**

L’obbiettivo del nostro progetto è la realizzazione di un social network minimalista con varie e semplici funzionalità iniziali tra cui:

* **Sign up** L’utente può creare un nuovo account.
* **Log in** : L’utente sceglie di fare il log in.
* **Delete account** : L’utente sceglie di eliminare il suo account.

Al seguito dell’avvenuto login l’utente ha accesso alle seguenti funzionalità:

* Visualizzare il nome di tutti gli utenti.
* Cercare se esiste un utente.
* Decidere di seguire un utente.
* Fare un post
* Vedere tutti i post di un utente.
* Vedere l’ultimo post di ogni utente seguito.
* Fare il log out.

Le uniche limitazioni applicare all’intero social sono:

* Post di solo testo.
* Massimo numero di post per account 10.

### **Soluzione**

#### **Conservazione dei dati**

L’approccio alla soluzione è stato diviso in: conservazione dei dati dell’utente,dei post e dei seguiti di un’utenza.

* **Dati utente**: i dati dell’utente sono stati unificati in un’unica struct di nome “data” al cui all’interno ho inserito username, password, post e seguiti. Quest’ultima è usata in una struct “utente” dove si conoscono i dati e la posizione del prossimo utente.
* **Post**: i post sono stati salvati in un array bidimensionale con massimo dieci righe ed una lunghezza massima del post di 400 caratteri.
* **Seguiti**: I seguiti sono stati salvati in una coda dove ogni seguito conosce l’utente(si punta direttamente alla lista di utente per evitare la ridondanza delle informazioni) quindi i suoi dati ed il prossimo seguito.

#### **Funzionalità in breve**

Ho separato le funzionalità del social in due menù separati prima e dopo il login dell’utente.Prima del login (*Figura 1*) l’utente ha a disposizione le seguenti funzionalità:

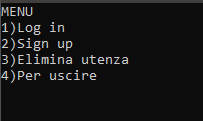
* Login
* Sign up
* Eliminazione dell’utenza
* Chiusura del programma

Figura 1 – Menu prima del log in

Mentre dopo aver effettuato il log in con username esistente e password si entra in un secondo menù

(*Figura 2*) dove abbiamo a disposizione le seguenti funzionalità richieste dal social:

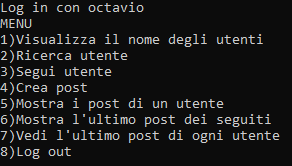
* Visualizzazione del nome degli utenti.
* Ricerca di un utente.
* Aggiunta di un utente ai seguiti.
* Creazione di un nuovo post.
* Mostra gli ultimi dieci post di un utente.
* Mostra l’ultimo post dei seguiti.
* Mostra l’ultimo post di ogni utente.
* Log out torna al menu precedente.

Figura 2 - Menu dopo il log in

### **Funzionalità e descrizione**

|  |  |
| --- | --- |
| **Funzioni** | **Descrizione** |
| Menu\_login | Viene visualizzato un menu dove l’utente ha possibilità di scelta sulle azioni da fare dopo il login. |
| Sign up | Viene creata una nuova utenza dove viene richiesto username (che viene verificato se già esistente), la password e la conferma password. |
| Aggiungi utente | Prende in input un’utenza che aggiunge alla lista già esistente. |
| Stampa utenti | Stampa il nome di tutti gli utenti scorrendo la lista. |
| Controllo utenza | Verifica l’esistenza di un’utenza in lista e la restituisce. |
| Controllo password | Verifica che la password inserita corrisponda con quella associata all’utenza. |
| Log in | Verifica che l’utenza esista e che la password sia corretta in caso affermativo |
| Ricerca utente | Verifica se l’utenza sia corretta. |
| Crea post | Creo un nuovo post per l’utente che ha effettuato il log in. |
| Stampa ultimo post utenti | Stampo l’ultimo post di tutti gli utenti |
| Mostra post | Mostra gli ultimi dieci post dell’utente inserito. |
| Segui utente | Inserisco un nuovo utente ai seguiti dell’utente che ha effettuato il log in. |
| Vedi post seguiti | Mostra l’ultimo post di tutti gli utenti seguiti. |
| Init | Inizializza il social con cinque utenti con dieci post ciascuno. |
| Elimina utente | Elimina l’utenza ed i dati associati. |
| Elimina seguito | Elimina l’utenza che si sta per eliminare dai seguiti degli altri utenti. |
| Controllo seguiti | Verifica che un’utenza non sia già seguita. |

Strutture dati utilizzate

Le strutture utilizzate per la conservazione delle informazioni dell’utente come precedentemente detto sono state divise in tre parti differenti che elencherò in forma tabellare con le rispettive descrizioni. Come lunghezze massime per i vari campi ho usato **16 come lunghezza fissa di password e username** **e 400 come lunghezza massima di un post. Per username e password ho utilizzato lo scanf quindi non sono ammessi spazi.**

### **Struct data**

Lo struct data è stato implementato per contenere le informazioni dell’utente come vediamo da tabella.

|  |  |
| --- | --- |
| Nome | Definizione |
| char username[USER\_LENGTH] | Contiene l’username dell’utente max 16 char. |
| char password[USER\_LENGTH] | contiene la password dell’utente max 16 char. |
| char post[10][POST\_LENGTH] | array bidimensionale contente massimo 10 post con una lunghezza di 400 char. |
| seguiti \* seguiti | contiene il puntatore alla testa della lista dei seguiti. |

### **Struct utente**

Lo struct utente è stato implementato per poter creare una lista di puntata a variabili di tipo struct utente.

|  |  |
| --- | --- |
| Nome | Definizione |
| data infoutente | Contiene username, password, post e seguiti di un utente. |
| utente \* next | Contiene l’indirizzo del prossimo utente della lista. |

### **Struct seguiti**

Lo struct seguiti è stato implementato per poter creare una lista di puntatori a variabili di tipo struct seguiti. Inoltre, ho evitato di aggiungere il nome per evitare la ridondanza dei dati ed ho direttamente puntato all’utenza seguita.

|  |  |
| --- | --- |
| Nome | Definizione |
| utente \* seguito | Contiene l’indirizzo dell’utente seguito di tipo utente. |
| seguiti \* next | Contiene l’indirizzo del prossimo utente della lista dei seguiti. |

Esempi di esecuzione

Mostrerò qui di seguito con alcune GIF delle azioni possibili all’interno del social minimalista. Le utenze presenti sono gli account di: luca, arturo, gibbo, octavio e roberto. Mentre le loro password sono tutte settate a “baubau” come si potrà vedere anche nelle varie immagini. I post iniziali sono uguali per le cinque utenze.

### Seguo un utente e stampo il suo ultimo post

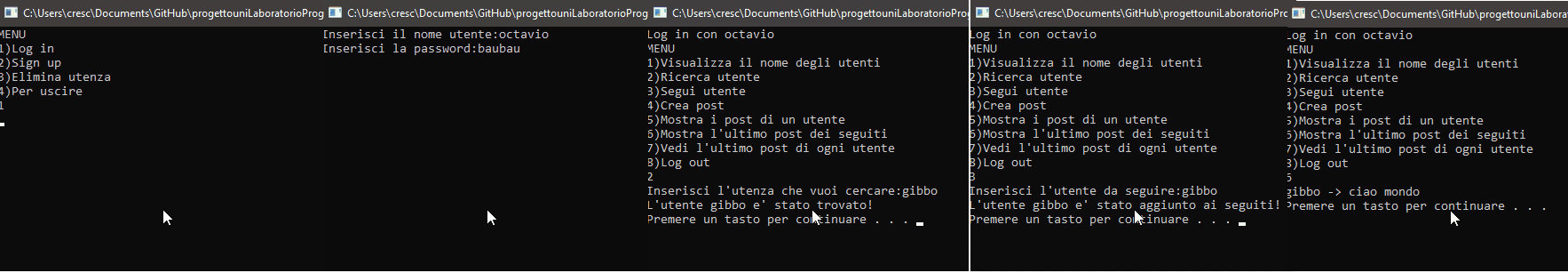
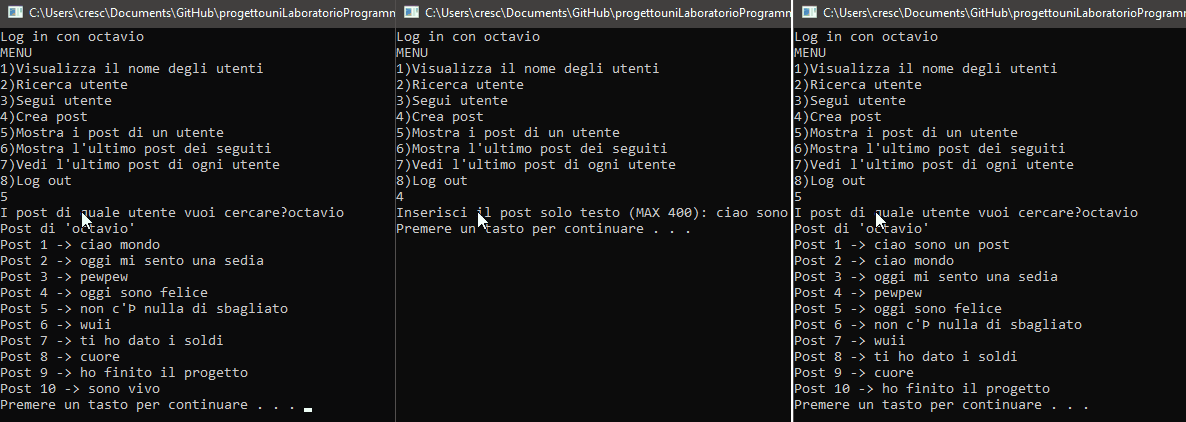


Figura 3 – Seguo un utente e stampo il suo ultimo post

Come possiamo notare in sequenza viene:

* Effettuato il log in.
* Si cerca se esiste un’utenza.
* La si segue.
* Si stampa l’ultimo post del seguito.

### Creo un post e lo mostro



### 

Figura - Creo un post e lo mostro

Come possiamo notare in sequenza viene:

* Mostro i post dell’utente.
* Creo un nuovo post.
* Mostro i post dell’utente con il nuovo post appena creato.

### Eliminazione di un’utenza e nelle relative liste in cui è seguito

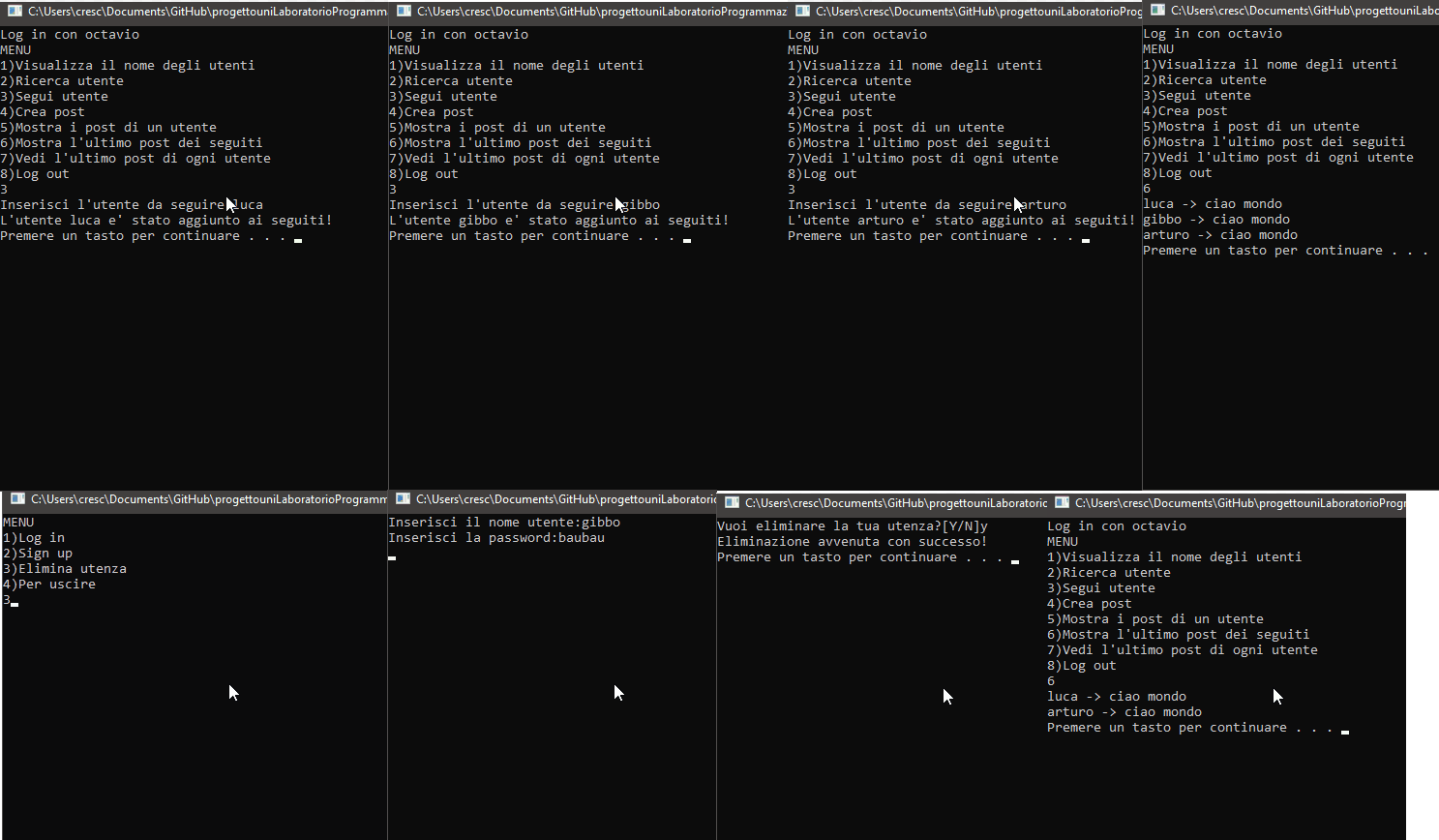


Figura 5 - Eliminazione di un’utenza e nelle relative liste in cui è seguito

Come possiamo notare in sequenza viene:

* Seguo tre utenti differenti.
* Stampo l’ultimo post dei seguiti.
* Elimino una delle utenze seguite.
* Verifico con username e password l’utenza.
* Confermo l’eliminazione.
* Log in ancora con l’utenza precedente e stampo ancora una volta gli ultimi post dei seguiti.