CashCommBot

Video Demo: CashCommBot - My Portable Finance Manager

Languages:

- English
- Italiano

English

This project was written to satisfy a need of mine: to keep track of all my expenses and to be able to see the trend of it over time using a tool that can be used inside and outside the home. For external use I chose to use a bot for the Telegram application, in which you need to be registered, also having a username associated with the account.

The name "CashCommBot" comes from an abbreviation of "Cash Command Bot" and, whose sound, resembles "Cash Control". In fact, these commands can be used both from the command line and from Telegram bots.

In general, the program allows you to do the following actions:

- Enter credentials for a new user;
- Change the password;
- Insert transactions into the database;
- Delete transactions from the database;
- Request the balance in a certain period of time;
- Extract your data, in chronological order of date, based on a specified period of time;
- Request the list of all transaction categories;
- Request the graph to see the trend based on the given specifications.

For admins, there are additional commands:

- Appoint as admin or remove the admin role from another user;
- Reset a user's password;
- Change a user's username;
- Delete a user from the database.

For each action other than inserting a new user, a username and password are required, in order to make the insertion, extraction or deletion of data more secure.

How the project works

The operation between the command line version and the bot version is almost identical. The main difference between the two versions (bot and command line) is the management for entering passwords to access the program's features. Since reporting the password multiple times may not be very secure, for the bot there is the possibility of being able to save the encrypted password in a session variable, so as not to have to write it every time (as you have to do from the command line). The fact that it is already encrypted also allows not to

understand the actual password, increasing security. From the command line I could not use the same system, so I used the getpass function of the Python getpass library to allow the password to be entered without the possibility of showing it visually. The same mechanism was also adopted for changing the password, in which, for each password specified, the same method was used.

Furthermore, whether it is the bot or the command line, there are several very precise checks, such as for the syntax of the specified data (e.g. dates or the amount entered for the transaction), for the presence or absence of the user in the database, or the correctness of the passwords. Consequently, in case of error, the incorrect data will be reported precisely, except for the username check in the database (for the command line) since it is not very safe to let people know which users use the program since there could be an interest in understanding which data has been entered when the actual user does not want to disclose it. In fact, to ensure better user privacy, I made sure that the password and username were encrypted, using different algorithms, so as not to be able to associate the user ID with a transaction if access to the database were to be gained.

Furthermore, to ensure even better privacy, I decided to also encrypt the description of the transactions, so that the detailed reason cannot be understood.

Telegram Bot

First of all, with the bot, using the **/start** command, you can enter, if the username has not already been entered, only one user (yours), automatically setting the password identical to the username. The password can be changed later using the **/password** command, in addition to which we must specify the current password, the new one and the confirmation of the new one, all separated by a space. Using the same command, in the bot, you can save the password, in case it is not or in case an incorrect password has been saved, specifying, after the **/password** command, the password you want to save, also separated by a space.

Subsequently, we can insert new transactions into the database using the **/transaction** command. For insertion, we must specify, separated by a comma, a description (useful to describe the reason for that transaction), the amount that was spent or earned, the date (in the format ***YYYY-MM-DD***), the personal category to which it belongs (e.g. "food" for expenses or "monthly" for earnings) and the type of transaction (expense or income).

In case of error or for other reasons, there is the opportunity to delete one or more transactions, using the **/delete** command. There are two ways to use this command:

- The first is to cancel the last operation performed;
- The second is to delete one or more transactions based on an input entered.

For the first case, we do not have to enter anything because, by typing only the command, the bot takes the last transaction and asks if you want to delete it via a clickable button. For the second case, specifying, separated by commas, the transaction details (keeping the same format as the entry), multiple transactions containing those details are sent and the bot also shows them here in the form of buttons. For the first case, in addition to the button to delete, it also shows a button to cancel the operation, while, for the second, it also shows a button to simultaneously delete each transaction in the list.

Depending on the button pressed, a different action is performed:

- If a button containing a transaction is pressed (including the button to delete all those present)

- In the first case, the message of deletion will be sent;
- In the second case, the text and the value of the button are replaced with "Deleted" and "del" respectively for a visual confirmation of the actual deletion;
- If the button to cancel or complete the operation is pressed, the operation is concluded and a message is sent
- In case of cancellation, the message contains the confirmation of the conclusion of the operation;
- In case of completion, the message contains the list of deleted transactions.

To have a clearer idea of the balance between incomes and expenses, the **/balance** command comes in handy, which returns the balance of a certain period of time. If executed individually, it will return the total balance, otherwise, if a time period is specified (***YYYY-MM-DD***/***YYYY-MM***/***YYYY*** or ***day***/***month***/***year***), the balance of that period only is returned.

If you want to extract your data, you can do so using the **/export** command, which, like the **/balance** command, if nothing is specified, refers to each transaction, otherwise, to all the data present in the specified time frame. However, unlike the balance, it is possible to specify one or two dates:

1. Extract all the data in that time range (e.g. in that day/month/year);
2. The extraction will take place between the first date and the day before the second date (e.g. if I request the extraction between YYYY-01 and YYYY-10-15 it will be done from YYYY-01 to YYYY-10-14).

If you want to extract data from a more precise date to a less precise one, you can specify two different types of dates, such as ***YYYY-MM*** and ***YYYY-MM-DD***, etc. Furthermore, if two dates are specified, the extraction will be managed automatically so that it is done between the smallest and the largest date, regardless of the order in which they are specified in the message.

Finally, you can request a graph to view the trend over a specific period of time. This is possible using the **/graph** command, which requires the following parameters separated by colons ("😊

- Title
  - Any text string is fine, as long as it does not contain some characters (**" ' \ / : * ? < > |**);
- Type
  - Indicates the type of transaction (***expense***/***income*** or ***tot*** to indicate the difference between income and expense);
- Time
  - Indicates the period of time from which to take the data (***day***/***month***/***year*** or ***tot*** to indicate each data in each year);
- Category
  - Specify (***cat?*** followed by the category) only if you do not choose "tot" for the transaction type, otherwise leave only ***cat?***.

If you do not remember the name of a transaction category, you can use the **/categories** command, which returns the list of all categories entered by the user.

Finally, admins have the possibility to send a message to multiple users, using the following syntax:

```
_SEND_ | _SEND_
TO | To: * | users, ...
MSG: ...
```

Command line

From the command line, the operation is very similar to the previous one, except for the commands to use and the syntax for inserting data.

Starting from the insertion of data in the table, it is possible to do so using the **--newuser** command, which does not require arguments, but, if inserted, the program will ask the user to insert a username, password and confirmation of it. If these latter match, the program will insert the username and password into the database, otherwise it returns an error message.

If you want to change the password later, you can do so using the **--password** command, which does not require arguments but the program will ask for the current password and, if it matches, also the new password and confirmation. In this case, as previously mentioned, it is not possible to save the password, but you will have to enter it every time you want to run the program.

As for inserting transactions, you need to use the **--transaction** command, which requires the same arguments as the bot command, with some differences:

- As a separator, you need to use a space between the arguments;
- If you want to insert arguments with spaces inside, they must be inserted between **"**.

If you want to delete one or more transactions, you must use the **--delete** command. This command can be used alone or together with another. If it is used alone, the last transaction is returned. If you want to understand which transactions have certain characteristics, you can use the **--delete** command combined with **--details**, specifying the transaction details separated by a semicolon (";"). This combination of commands will return a list of transactions that will have those certain characteristics in those certain fields. If we want to permanently delete the data, we must use the **--delete** command with the **--todelete** command, specifying "**all**" or a list of ids, separated by a space, followed, in the case of "**all**", by the category details (separated by a semicolon as for **--details**), while, in the case of the list of ids, even just an empty string (as it will not affect the process).

As for the balance request, you need to use the **--balance** command, in addition to which we can specify nothing, returning the total balance, or we can specify a generic period of time using the **--time** command, preceded by **--balance** and followed by the period in the format ***YYYY-MM-DD*/*YYYY-MM*/*YYYY***, which will return the balance on that day/month/year, or ***day*/*month*/*year***, which will return the balance on this day/month/year.

For data extraction, we use the **--export** command. If we do not specify anything, the file containing each user extraction will be returned, otherwise, if we want to extract the data in a certain day/month/year, we use the command **--export** followed by **--date**, in addition to which we must specify the date, based on the period in which we want the extraction, according to the structures ***YYYY-MM-DD*/*YYYY-MM*/*YYYY***. If we want to have a wider range, we should use, in addition to **--export**, the command **--range**, which requires 2 dates, separated by a space, according to the structure ***YYYY-MM-DD*/*YYYY-MM*/*YYYY*** (also combined as YYYY-MM and YYYY-MM-DD), and which will return the transactions in the specified range, that is, from the lower date to the day before the higher date.

Finally, if we wanted to request the graph, we should use the **--graph** command, in addition to which we should specify the same parameters for the bot command, that is, a title, the type of graph (***income***/***expense*** or ***tot*** for the difference between the previous two), the time period (***day***/***month***/***year*** or ***tot*** to indicate every transaction up to now) and the category, which, unlike the bot syntax, does not require ***cat?*** before the category. In fact, if you do not want to specify one, you can leave an empty string, between two quotes, separated by a space (**" "**).

Again, if you do not remember the name of a category, you can request a list of them using the **--categories** command.

Unlike the bot, additional commands can be executed for admins.

If you want to appoint another user as admin, you can do so using the **--admin** command which requires, as its only argument, a username or a list of usernames (between "" and separated by a comma). This role is not permanent as it can be removed by adding the **--roleremove** command after the list of usernames.

If we want to reset a password, we must use the **--reset** command followed by one or more usernames (with the same logic as for **--admin**). The program will ask for the password and then a confirmation and, if they match, it will assign the same password to each existing user in the list.

If we want to change the username of one or more users, we must use the **--username** command followed by two parameters: the old username(s) and the ones we want to change them with. It is possible to insert multiple usernames with the same logic as above, and for each existing username, the corresponding new username is associated.

Finally, the admin has the possibility to remove the user's username via the **--remove** command, which requires a list of usernames (as reported in the previous commands). If it exists, the user is removed from the database (as is all of his data).

Tests carried out

Since the secondary functions are used within the main ones, I decided to test only the latter.

As for the functions that create the files, I chose to also test the presence of the latter and their elimination, using the Python os library, in order to check the presence or absence from the respective folders.

Future extensions

For this project, in addition to an initial use from the command line and Telegram bot, I had also thought of creating a graphical part, for a more user-friendly use. For this reason I intend to add this logic to the final project for the CS50 W course.

In this second part of the project, I would use the same logic used in this project, with the addition of a more secure login page, some web applications for the same project, in order to separate the earnings part from the expenses part, without changing the database structure so as not to upset the functions already written.

In addition, the general graphs will already be available visually and there will be a section where you can filter the data to have more detailed graphs based on the needs of the users.

I saw that in the specifications of the final project of the CS50 W course a greater commitment is required and it is necessary to justify in case there is a lot of similarity with other projects. I would like to say that the idea of

doing such a "large" project came to me because at least my friends can also use this project in a simpler way, in addition to Telegram and command line.

Italiano

Questo progetto è stato scritto per soddisfare una mia esigenza: tenere traccia di tutte le mie spese e poter vedere l'andamento di esso nel tempo usando uno strumento che possa essere utilizzato dentro e fuori casa. Per l'utilizzo esterno ho scelto di utilizzare un bot per l'applicazione Telegram, nel quale bisogna essere registrati, avendo anche uno username associato all'account.

Il nome "CashCommBot" deriva da un abbreviazione di "Cash Command Bot" e, il cui suono, assomiglia a "Cash Control". Infatti, questi comandi possono essere utilizzati sia da riga di comando, sia da bot di Telegram.

In generale, il programma permette di fare le seguenti azioni:

- Inserimento delle credenziali per un nuovo utente;
- Cambiare la password;
- Inserire transazioni nel database;
- Eliminare transazioni dal database;
- Richiedere il bilancio in un determinato periodo di tempo;
- Estrarre i propri dati, in ordine cronologico di data, in base ad un periodo di tempo specificato;
- Richiedere la lista di tutte le categorie delle transazioni;
- Richiedere il grafico per vedere l'andamento in base alle specifiche date.

Per gli admin, ci sono ulteriori comandi:

- Nominare come admin o togliere il ruolo di admin ad un altro utente;
- Resettare la password di un utente;
- Cambiare lo username di un utente;
- Eliminare un utente dal database.

Per ogni azione diversa dall'inserimento di un nuovo utente, sono richiesti username e password, in modo da rendere più sicuri l'inserimento, l'estrazione o l'eliminazione dei dati.

Funzionamento del progetto

Il funzionamento tra la versione da riga di comando e quella da bot sono pressochè identiche. La sostanziale differenza tra le due versioni (bot e riga di comando) è la gestione per l'inserimento di password per accedere alle funzionalità del programma. Siccome riportare la password più voltre potrebbe non essere molto sicuro, per il bot c'è la possibilità di poter salvare la password cifrata in una variabile di sessione, in modo da non doverla scrivere ogni volta (come si deve fare da riga di comando). Il fatto che sia già cifrata, permette, inoltre, di non poter capire la password effettiva, aumentando la sicurezza. Da riga di comando non potevo usare lo stesso sistema, quindi ho usato la funzione getpass della libreria getpass di Python per permettere l'inserimento della password senza la possibilità di mostrarla visivamente. Stesso meccanismo è stato adottato anche per il cambio della password, nel quale, per ogni password specificata, è stato utilizzato lo stesso metodo.

Inoltre, sia che si tratti del bot, sia che si tratti di linea di comando, ci sono diversi controlli molto precisi, come per la sintassi dei dati specificati (es date o l'ammontare inserito per le transazione), per la presenza o meno dell'utente nel database, o la correttezza delle password. Di conseguenza, in caso di errore, verrà segnalato, in

modo preciso, il dato errato, eccezione fatta per il controllo dello username in database (per quanto riguarda la riga di comando) poichè non è molto sicuro far sapere quali utenti utilizzano il programma dato che ci potrebbe essere interesse nel capire quali dati sono stati inseriti quando l'utente effettivo non vuole divulgarli. Infatti, per garantire una migliore privacy dell'utente, ho fatto in modo che la password e lo username fossero cifrati, tramite algoritmi diversi, in modo da non poter associare l'id utente ad una transazione se si dovesse avere accesso al database.

Inoltre, per garantire una privacy ancora migliore, ho deciso di cifrare anche la descrizione delle transazioni, in modo che non si possa capirne il motivo dettagliato.

Bot di Telegram

Innanzitutto, con il bot, tramite il comando **/start**, è possibile inserire, se lo username non era già stato inserito, solo un utente (il proprio), impostando, automaticamente, la password identica allo username. La password è possibile cambiarla successivamente tramite il comando **/password**, oltre al quale dobbiamo specificare la password attuale, quella nuova e la conferma di quella nuova, il tutto separato sa uno spazio. Tramite lo stesso comando, nel bot, è possibile salvare la password, nel caso non lo sia o nel caso sia stata salvata una password errata, specificando, dopo il comando **/password** la password che si vuole salvare, anch'essa separata da uno spazio.

Successivamente, possiamo inserire nuove transazioni nel database tramite il comando **/transaction**. Per l'inserimento dobbiamo specificare, separato da una virgola, una descrizione (utile a descrivere il motivo di quella transazione), la quantità che è stata spesa o guadagnata, la data (nel formato ***YYYY-MM-DD***), la categoria personale alla quale appartiene (es "cibo" per le spese o "mensile" per i guadagni) ed il tipo della transazione (expense o income).

In caso di errore o per altri motivi, c'è l'opportunità di eliminare una o più transazioni, tramite il comando **/delete**. Ci sono due modi per usare questo comando:

- Il primo è per annullare l'ultima operazione eseguita;
- Il secondo è per eliminare una o più transazioni in base ad un input inserito.

Per il primo caso non dobbiamo inserire nulla perchè, digitando solo il comando, il bot prende l'ultima transazione e chiede se si vuole eliminarla tramite un bottone cliccabile. Per il secondo caso, specificando, separati da virgole, i dettagli della transazione (tenendo lo stesso formato dell'inserimento), vengono inviate più transazioni contenenti quei dettagli ed il bot li mostra anche qui sottoforma di bottoni. Per il primo caso, oltre al bottone per eliminare, mostra anche un bottone per annullare l'operazione, mentre, per il secondo, mostra anche un bottone per eliminare contemporaneamente ogni transazione della lista.

In base al bottone premuto, viene eseguita una azione diversa:

- Se viene premuto un bottone contenente una transazione (compreso il bottone per l'eliminazione di tutte quelle presenti)
    - Nel primo caso, verrà inviato il messaggio di avvenuta eliminazione;
    - Nel secondo caso, viene sostituito il testo ed il valore del bottone rispettivamente con "Deleted" e "del" per una conferma visiva dell'effettiva eliminazione;
- Se viene premuto il bottone per l'annullamento o il completamento dell'operazione, viene conclusa l'operazione ed inviato un messaggio
    - In caso di annullamento, il messaggio contiene la conferma di conclusione dell'operazione;

    - In caso di completamento, il messaggio contiene l'elenco delle transazioni eliminate.

Per avere un'idea più chiara del bilancio tra incomes e expenses, viene in aiuto il comando **/balance** che restituisce il bilancio di un determinato periodo di tempo. Se viene eseguito singolarmente, restituirà il bilancio totale, altrimenti, se viene specificato un periodo di tempo (***YYYY-MM-DD***/***YYYY-MM***/***YYYY*** oppure ***day***/***month***/***year***), viene restituito il bilancio solo di quel periodo.

Nel caso si volesse estrarre i propri dati, è possibile tramite il comando **/export**, nel quale, come il comando **/balance**, se non viene specificato niente, fa riferimento ad ogni transazione, altrimenti, a tutti i dati presenti nell'arco di tempo specificato. Però, a differenza del bilancio, è possibile specificare una o due date:

1. Estrazione di tutti i dati in quel range di tempo (es in quel giorno/mese/anno);
2. L'estrazione avverrà tra la prima data e il giorno prima della seconda data (es se richiedo l'estrazione tra il YYYY-01 ed il YYYY-10-15 verrà fatta dal YYYY-01 al YYYY-10-14).

In caso si vogliano estrarre dati da una data più precisa ad una meno, si possono specificare due tipi di date diversi, come ***YYYY-MM*** e ***YYYY-MM-DD***, etc. Inoltre, in caso vengano specificate due date, l'estrazione verrà gestita in automatico in modo che venga fatta tra la data più piccola e quella più grande, a prescindere dall'ordine in cui esse vengono specificate nel messaggio.

Infine è possibile richiedere un grafico per visualizzare l'andamento in un determinato periodo di tempo. Ciò è possibile utilizzando il comando **/graph**, il quale richiede i seguenti parametri separati dai due punti (":blush:

- Titolo
    - Va bene qualsiasi stringa di testo, purchè non contenga alcuni caratteri (**" ' \ / : * ? < > |**);
- Tipo
    - Indica il tipo della transazione (***expense***/***income*** oppure ***tot*** per indicare la differenza tra income ed expense);
- Tempo
    - Indica il periodo di tempo dal quale prendere i dati (***day***/***month***/***year*** oppure ***tot*** per indicare ogni dato in ogni anno);
- Categoria
    - Da specificare (***cat?*** seguito dalla categoria) solo in caso non si scelga "tot" per il tipo di transazione, altrimenti lasciare solo ***cat?***.

In caso non si ricordi il nome di una categoria, si può ricorrere all'uso del comando **/categories**, il quale restituisce l'elenco di tutte le categorie inserite dall'utente.

Infine, gli admin, hanno la possibilità di inviare un messaggio a più utenti, tramite la seguente sintassi:

```
\_SEND\_ | \_INVIA\_
TO | A: \* | users, ...
MSG: ...
```

Riga di comando

Da riga di comando, il funzionamento è molto simile al precedente, se non per i comandi da utilizzare e la sintassi per l'inserimento dei dati.

Partendo dall'inserimento dei dati in tabella, è possibile farlo utilizzando il comando **--newuser**, il quale non richiede argomenti, ma, se inserito, il programma farà inserire all'utente uno username, la password e la conferma di esso. In caso queste ultime combacino, il programma inserirà il nome utente e la password all'interno del database, altrimenti restituisce un messaggio di errore.

In caso si voglia cambiare successivamente la password è possibile farlo tramite il comando **--password**, il quale non richiede argomenti ma il programma richiederà la password attuale e, se corrisponde, anche la nuova password e la conferma. In questo caso, come detto precedentemente, non è possibile salvare la password, ma bisognerà inserirla ogni volta che si vuole eseguire il programma.

Per quanto riguarda l'inserimento di transazioni, bisogna usare il comando **--transaction**, il quale richiede gli stessi argomenti per il comando del bot, con alcune differenze:

- Come separatore, è necessario utilizzare lo spazio tra i vari argomenti;
- In caso si vogliono inserire argomenti con degli spazi all'interno, devono essere inseriti tra **"**.

In caso si vogliano eliminare una o più transazioni, bisogna utilizzare il comando **--delete**. Questo comando può essere usato da solo o insieme ad un altro. In caso venga utilizzato da solo, viene restituita l'ultima transazione. In caso si voglia capire quali sono le transazioni che abbiano determinate caratteristiche, è possibile utilizzare il comando **--delete** combinato a **--details**, specificando i dettagli della transazione separati dal punto e virgola (";"). Questa combinazione di comandi restituirà un elenco di transazioni che avranno quelle determinate caratteristiche in quei determinati campi. Se vogliamo eliminare definitivamente i dati, dobbiamo usare il comando **--delete** con il comando **--todelete**, specificando "**all**" oppure una lista di id, separati da uno spazio, seguito, in caso di "**all**", dai dettagli delle categorie (separati dal punto e virgola come per **--details**), mentre, in caso della lista di id, anche solo una stringa vuota (poichè non influirà sul processo).

Per quanto riguarda la richiesta del bilancio, bisogna utilizzare il comando **--balance**, oltre al quale possiamo non specificare nulla, restituendoci il bilancio totale, oppure possiamo specificare un lasso di tempo generico usando il comando **--time**, preceduto da **--balance** e seguito dal periodo nel formato ***YYYY-MM-DD*/*YYYY-MM*/*YYYY***, che restituirà il bilancio in quel giorno/mese/anno, oppure ***day*/*month*/*year***, il quale ci restituirà il bilancio in questo giorno/mese/anno.

Per l'estrazione dei dati, utiliziamo il comando **--export**. Se non specifichiamo nulla, verrà restituito il file contenente ogni estrazione dell'utente, altrimenti, se vogliamo estrarre i dati in un determinato giorno/mese/anno, utiliziamo il comando **--export** seguito da **--date**, oltre al quale dobbiamo specificare la data, in base al periodo in cui vogliamo l'estrazione, secondo le strutture ***YYYY-MM-DD*/*YYYY-MM*/*YYYY***. In caso volessimo avere un range più ampio, dovremmo utilizzare, in aggiunta ad **--export**, il comando **--range**, il quale richiede 2 date, separate da uno spazio, secondo la struttura ***YYYY-MM-DD*/*YYYY-MM*/*YYYY*** (anche combinata come YYYY-MM e YYYY-MM-DD), e che restituirà le transazioni nel range specificato, ovvero dalla data inferiore al giorno prima della data superiore.

Infine, se volessimo richiedere il grafico, dovremmo utilizzare il comando **--graph**, oltre al quale dovremmo specificare gli stessi parametri per il comando del bot, ovvero un titolo, il tipo di grafico (***income*/*expense*** o ***tot*** per la differenza dei precedenti due), il periodo di tempo (***day*/*month*/*year*** o ***tot*** per indicare ogni transazione fino ad ora) e la categoria, la quale, a differenza per la sintassi del bot, non richiede ***cat?*** prima della categoria. Infatti, se non si vuole specificarne una, è possibile lasciare una stringa vuota, tra due apici o virgolette, separate da uno spazio (**" "**).

Anche in questo caso, se non si ricorda il nome di una categoria, è possibile richiederne la lista utilizzando il comando **--categories**.

A differenza del bot, è possibile eseguire ulteriori comandi per gli admin.

Se si vuole nominare un altro utente come admin, è possibile farlo tramite il comando **--admin** il quale richiede, come unico argomento, uno username o una lista di username (tra "" e separati da una virgola). Questo ruolo non è definitivo poichè può essere rimosso, aggiungendo il comando **--roleremove** dopo l'elenco di username.

Se si vuole resettare una password, dobbiamo usare il comando **--reset** seguito da uno o più username (con la stessa logica riportata per **--admin**). Il programma chiederà la password e successivamente una conferma e, se corrispondono, assegnerà, ad ogni utente esistente della lista, la stessa password.

Se si vuole cambiare lo username di uno o più utenti, dobbiamo usare il comando **--username** seguito da due parametri: lo/gli username vecchi e quelli con cui vogliamo cambiarli. E' possibile inserire più username con la stessa logica riportata sopra, e, per ogni username esistente, viene associato il corrispettivo nuovo username.

Infine, l'admin ha la possibilità di rimuovere lo username dell'utente tramite il comando **--remove**, il quale richiede una lista di username (come riportato nei precedenti comandi). Se esiste, l'utente viene rimosso dal database (come ogni suo dato).

Test effettuati

Siccome le funzioni secondarie sono utilizzate all'interno di quelle principali, ho deciso di testare solo queste ultime.

Per quanto riguarda le funzioni che creano i file, ho scelto di testare anche la presenza di questi ultimi e la loro eliminazione, sfruttando la libreria os di Python, in modo da controllare la presenza o assenza dalle rispettive cartelle.

Estensioni future

Per questo progetto, oltre ad un iniziale utilizzo da riga di comando e bot di Telegram, avevo pensato di crearne anche una parte grafica, per un utilizzo più user-friendly. Per questo motivo ho intenzione di aggiungere questa logica, al progetto finale per il corso di CS50 W.

In questa seconda parte del progetto, utilizzerei la stessa logica utilizzata in questo progetto, con l'aggiunta di una pagina login più sicura, alcune applicazioni web per lo stesso progetto, in modo da separare la parte dei guadagni da quella delle spese, senza però cambiare la struttura del database per non stravolgere le funzioni già scritte.

Inoltre, i grafici generali saranno già disponibili visivamente e ci sarà una sezione in cui si potranno filtrare i dati per avere grafici più dettagliati in base alle esigenze degli utenti.

Ho visto che nelle specifiche del progetto finale del corso CS50 W è richiesta un impegno maggiore e bisogna giustificare in caso ci sia molta somiglianza con altri progetti. Io vorrei già dire che l'idea di fare un progetto così "ampio" mi è venuta perchè almeno anche i miei amici possono utilizzare questo progetto in modo più semplice, oltre che da Telegram e riga di comando.