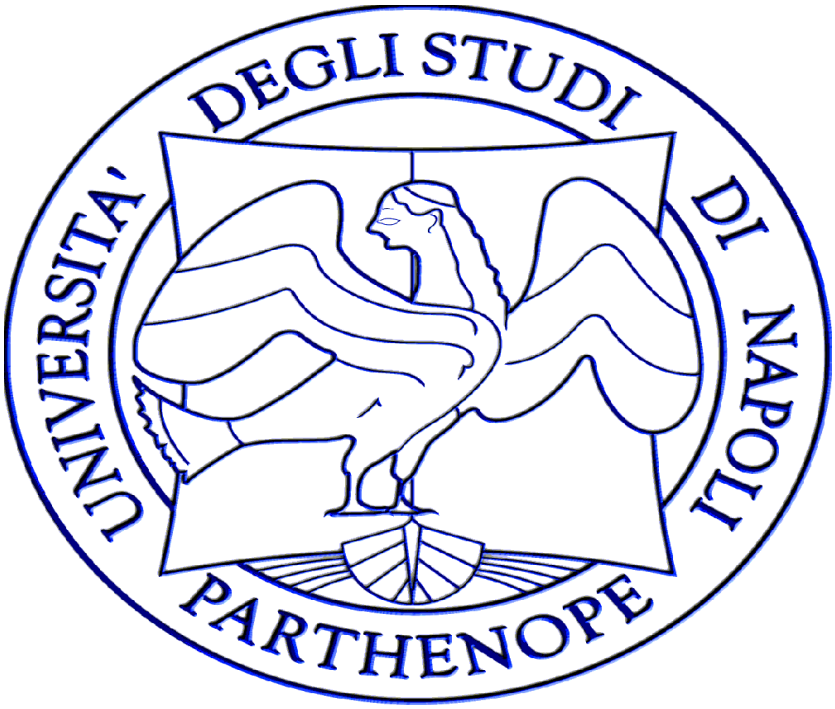


Laboratorio di Reti di Calcolatori

Green Pass

Ilardo Gianluca -0124/2368

Prof Alessio Ferone

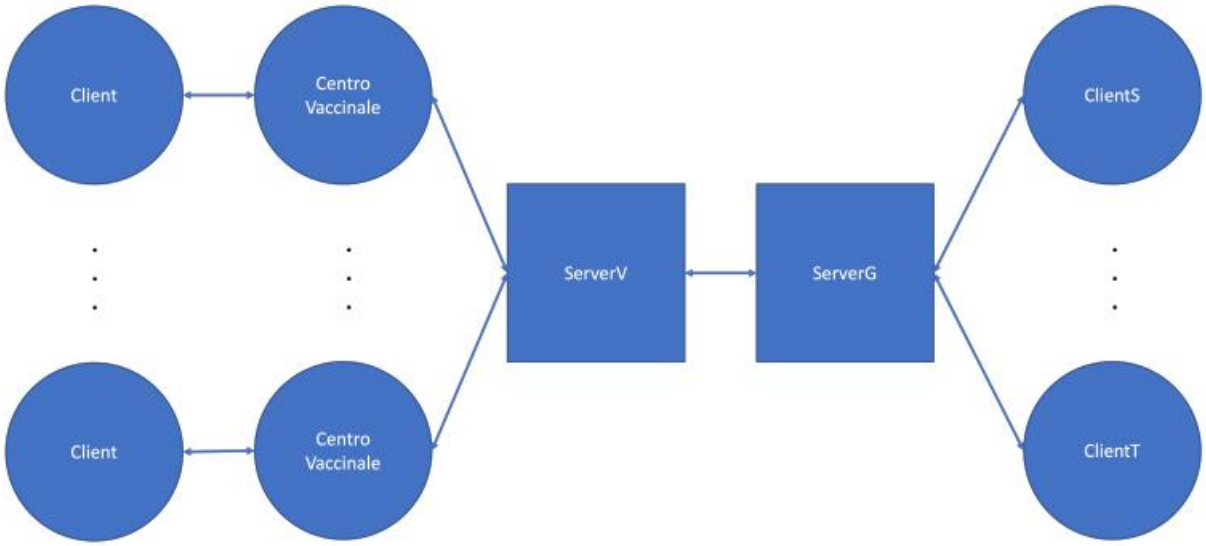
11/10/2022

Sommario

Descrizione del progetto

L'applicazione realizzata è un'applicazione client server che implementa un servizio di gestione dei green pass secondo le seguenti specifiche: un utente, una volta effettuata la vaccinazione, tramite un client si collega ad un centro vaccinale e comunica il codice della propria tessera sanitaria. Il centro vaccinale, una volta inserito in un apposito pacchetto applicazione il codice ricevuto, il periodo di validità del green pass e altri dati (li illustreremo più avanti nel paragrafo del pacchetto applicazione), invia questo pacchetto al serverV che lo salva in database attraverso un'apposita query. Un clientS, per verificare se è un green pass è valido, invia il codice di una tessera sanitaria al serverG il quale richiede al serverV il controllo della validità: il serverV controllerà tra i green pass salvati se è presente quello del clientS e, in caso sia presente, invierà al serverG la risposta con la validità del green pass e il serverG la inoltrerà al clientS. Un clientT, infine, può invalidare o ripristinare la validità di un green pass comunicando al serverG il contagio o la guarigione di una persona attraverso il codice di tessera sanitaria. Il serverG inoltrerà anche in questo caso la richesta al serverV che, una volta aggiornata la validità del green pass, invierà l'esito al serverG che a sua volta lo inoltrerà al clientT.

Descrizione e schemi dell'architettura



Lo schema qui presente mostra l'architettura dell'applicazione. Osservandolo possiamo notare che i client non hanno un diretto accesso al serverV, infatti:

-Il client a cui si connette l'utente per comunicare la propria vaccinazione deve prima inviare il codice della propria tessera sanitaria al centro vaccinale che a sua volta lo inoltrerà al ServerV.

-Il clientS, per verificare se il proprio green pass è ancora valido, dovrà passare per il serverG, inviandogli i propri dati, e aspettare che il serverG chieda informazioni sulla validità al serverV. Quest'ultimo verificherà se il green pass dell'utente è valido e inoltrerà la risposta a clientS passando per il serverG.

-Il clientT comunica soltanto con serverG, inviandogli la propria tessera sanitaria e la richiesta di ripristinare o annullare la validità del proprio green pass. Il serverG invia la richiesta al serverV che cambierà la validità del green pass e invierà l'esito dell'operazione.

Descrizione e schemi del protocollo applicazione

In questa applicazione è presente un pacchetto applicazione, chiamato pacchettoGreenPass, che viene scambiato tra tutte le entità tranne quella del client utente. Il pacchetto è una struct costituita da quattro variabili:

-codiceFiscale: una stringa contenente il codice fiscale dell'utente.

-mesiGreenPass: un intero che indica quanti mesi è ancora valido il green pass.

-validita: un intero che indica se il green pass è valido o no.

-ID: un intero che indica quale client ha inviato il pacchetto.

codiceFiscale	**mesiGreenPass**
validita	**ID**

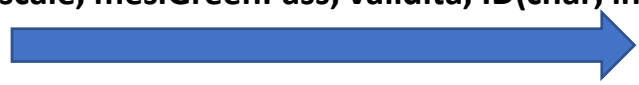
Seguono adesso degli schemi del protocollo applicazione che mostrano come vengono scambiati i dati tra le varie entità.

1)Gestione della parte relativa al client utente:

ClientUtente **CentroVaccinale** **ServerV**

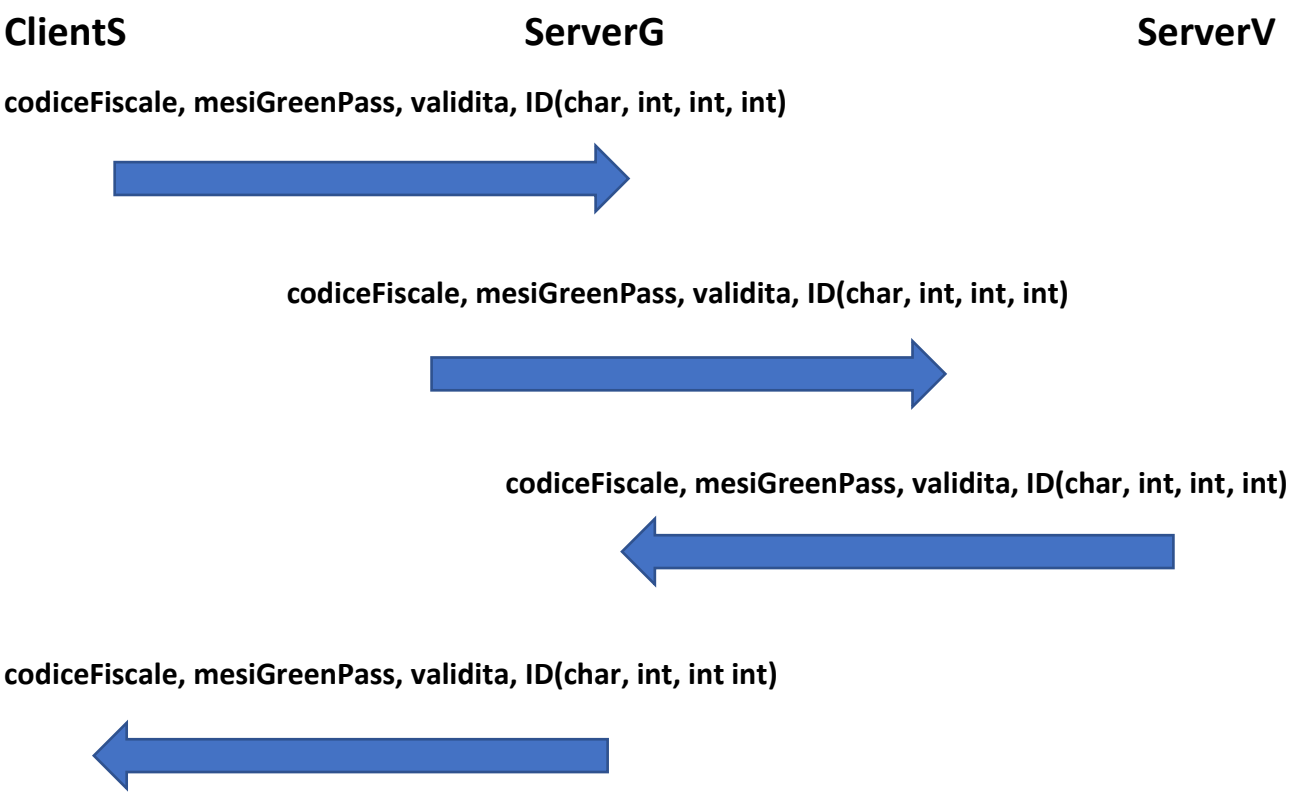
codiceFiscale(char)

codiceFiscale, mesiGreenPass, validita, ID(char, int, int int)



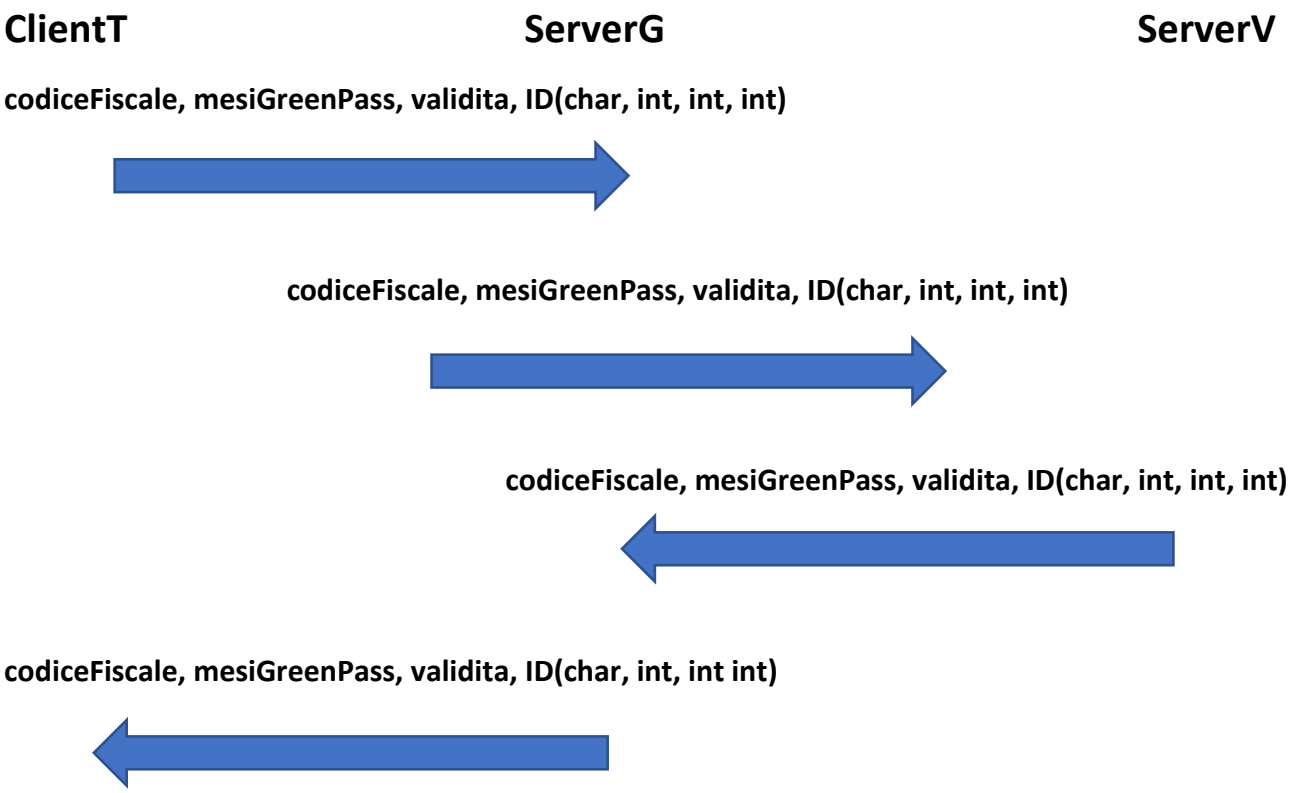
Il client utente invia il proprio codice fiscale al centro vaccinale che lo inserisce nel pacchetto applicazione definendo anche le altre variabili e lo invia al serverV, che salva le informazioni nel database attraverso una query.

2)Gestione della parte relativa al clientS:



Il clientS, per verificare se il proprio green pass è ancora valido, invia un pacchetto applicazione al serverG, che lo inoltra a sua volta al serverV. Il serverV verifica nel database, attraverso una query, se è presente un green pass con lo stesso codice fiscale e, in caso affermativo, spedisce al serverG il proprio pacchetto applicazione con all'interno anche la validità del green pass. Il serverG inoltra al clientS il pacchetto ricevuto e, il clientS, grazie alla variabile "validita" contenuta nel pacchetto, verifica se il suo green pass è valido o no.

3)Gestione della parte relativa al clientT:



La comunicazione che coinvolge il clientT segue gli stessi passaggi di quella che coinvolge clientS: Il clientT, per aggiornare la validità del proprio green pass, invia un pacchetto applicazione al serverG, che lo inoltra a sua volta al serverV. Il serverV verifica che nel database sia presente il green pass ricevuto e, in caso affermativo, ne cambia la validità per poi spedire al serverG il proprio pacchetto applicazione con all'interno la validità aggiornata del green pass di clientT. Il serverG inoltra al clientT il pacchetto ricevuto e il clientT viene quindi aggiornato dell'esito dell'operazione.

Dettagli implementativi dei client

L'applicazione è composta da tre client: clientUtente, clientS e clientT. Tutti e tre sono caratterizzati dallo stesso schema: nel main abbiamo la dichiarazione delle variabili, seguita dalla configurazione e la connessione con il server. Dopo che il client si è connesso al server attraverso la connect, viene richiamata una funzione che andrà a gestire le comunicazioni con le altre entità attraverso le FullRead e le FullWrite.

Nel caso di clientUtente non si fa altro che inviare attraverso la FullWrite il codice fiscale dell'utente (inserito attraverso riga di comando) al centro vaccinale: quest'ultimo lo inserirà in un pacchetto applicazione (descritto nel paragrafo precedente) per inviarlo al serverV.

Il clientS invece, oltre ad inviare con la FullWrite un pacchetto applicazione, la cui variabile codiceFiscale sarà inserita da riga di comando, aspetterà anche di ricevere un pacchetto di risposta dal ServerG (che avrà comunicato con serverV) con una FullRead per verificare la validità del proprio green pass. Alla fine con un semplice controllo sulla variabile "validita" del pacchetto ricevuto il clientS capisce se il green pass è valido oppure no.

Il clientT eseguirà le stesse comunicazioni del clientS, ma dato che il suo scopo è quello di cambiare la validità del proprio green pass, riceverà con l'ultima FullRead un pacchetto contenente la nuova validtà.

Dettagli implementativi dei server

I due server di questa applicazione, ServerV e serverG, seguono inizialmente lo stesso schema implementativo: per prima cosa si dichiarano le variabili, poi c'è la configurazione dell'indirizzo del server con la bind e della lista d'attesa dei client con la listen. Successivamente in un ciclo while viene effettuata la connessione ad un client con la accept, usando il descrittore "conn_fd". Successivamente i due server saranno gestiti in modo diverso, proseguirò quindi a descriverli separatamente:

ServerV

Il serverV, per gestire le richieste dei client, usufruirà dei thread (scelta fatta per una migliore gestione e sicurezza durante l'aggiornamento dei dati): attraverso la funzione pthread_create infatti genererà un nuovo thread per la gestione delle richieste dei client. Il processo padre continuerà a gestire solo la lista d'attesa per accettare altri client, mentre il thread generato gestirà le richieste dei client. Esso, per capire da quale client è partita la richiesta da gestire, farà un controllo sulla variabile ID del pacchetto applicazione:

se riceve una richiesta dal centro vaccinale (ID = 1) esegue una copia dei dati nel pacchetto applicazione all'interno di un database attraverso una query.

Se invece riceve una richiesta da serverG con la variabile ID del pacchetto uguale a 2 (richiesta inviata da clientS) il ServerV controllerà se il codice fiscale nel pacchetto ricevuto coincide con uno dei codici fiscali all'interno del database: in caso affermativo invia un pacchetto di risposta contenete la validità del green pass

attraverso una FullWrite a ServerG che lo rispedirà a clients. Quest'ultimo controllerà così la validità del proprio green pass.

Infine, se la richiesta del serverG avrà nel pacchetto applicazione la variabile ID uguale a 3 (richiesta inviata da clientT) il serverV controllerà se il codice fiscale nel pacchetto ricevuto coincide con uno dei codici fiscali all'interno del database: in caso affermativo cambia la validità del green pass con una apposita query e invia un pacchetto con la nuova validità attraverso una FullWrite a serverG che lo rispendirà a clientT. Quest'ultimo riceverà l'esito dell'operazione.

ServerG

Il serverG, attraverso la funzione fork, genererà un processo figlio: il processo padre continuerà solo a gestire la lista d'attesa e ad accettare altri client, mentre il processo figlio (pid = 0) gestirà le richieste. Il processo figlio controllerà anch'esso la variabile ID per capire da chi arriva la richiesta:

Se serverG riceve una richiesta da clientS (ID = 2) invierà il pacchetto ricevuto a serverV che, dopo l'opportuno controllo descritto in precedenza, invierà un pacchetto di risposta a serverG. Con la funzione FullWrite il pacchetto sarà spedito a clientS che controllerà la validità del green pass.

Se riceve una richiesta da clientT (ID = 3) serverG eseguirà operazioni analoghe: inoltrerà la richiesta al serverV che dopo aver cambiato la validità del green pass rispedirà indietro un pacchetto applicazione con la validità aggiornata.

Come possiamo notare, il serverG ha sia delle funzioni da client che da server: esso accetta le richieste da clientS e clientT come un server ma si collega al serverV con delle funzioni da client.

Centro vaccinale

Abbiamo infine un'entità, il centro vaccinale, che come il serverG ha delle funzioni sia da client che da server: esso, infatti, è connesso al client utente, riceve da esso il codice fiscale dell'utente, e si connette al serverV per inoltrargli un pacchetto contenente questo codice. Esegue quindi sia la bind per essere rintracciato dal client che una connect per connettersi al server, ed ha anch'esso un ciclo while con la creazione di un processo figlio che gestirà la richiesta del client.

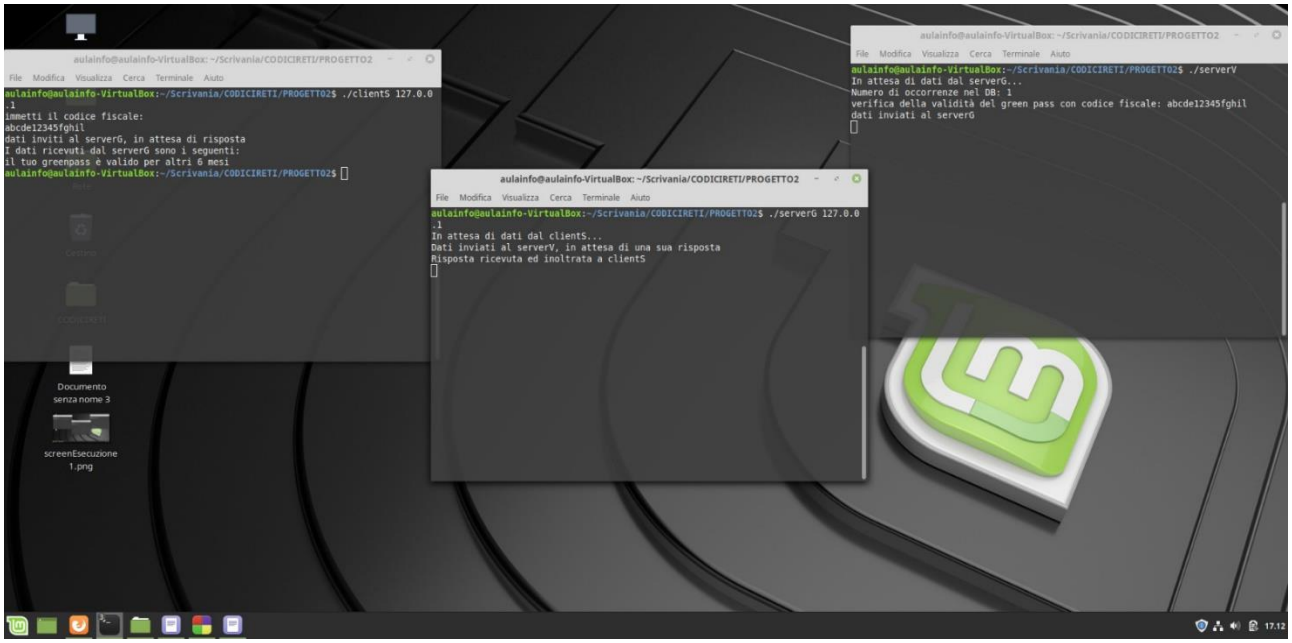
Manuale utente

Per eseguire l'applicazione è necessario eseguire prima di tutto il ServerV da riga di comando. Successivamente si potranno eseguire il centro vaccinale e il client utente: una volta avviato il client utente si potrà inserire da riga di comando il codice fiscale (massimo 15 caratteri) ed esso sarà inviato al centro vaccinale che lo inoltrerà al serverV. I dati salvati in ServerV riguardo il codice fiscale e i mesi di validità del green pass saranno visualizzati a video.

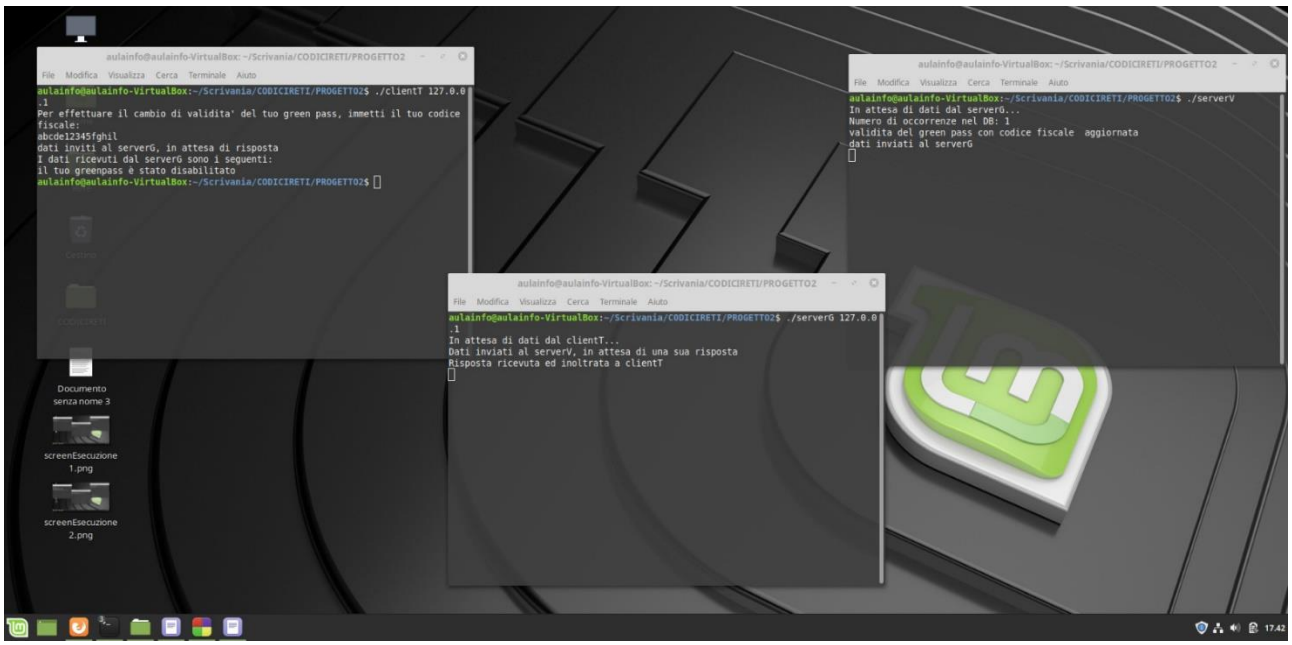
Successivamente, per testare le altre entità, va avviato prima il serverG, poi clientS o clientT: per entrambi i client va inserito anche questa volta il codice fiscale da riga di comando. Nel primo caso serverG richiederà la validità del green pass al serverV e la invierà al clientS che la visualizzerà. Nel secondo caso il serverG cambierà la validità del clientT come richiesto. Seguono alcuni screen dell'esecuzione del programma:

Screen della gestione della richiesta del client utente

Screen della gestione della richiesta del clientS:



Screen della gestione della richiesta del clientT:



NOTE

Il database utilizzato per salvare ed aggiornare le informazioni è mysql. All'interno della cartella del codice con i file C è presente un file attraverso il quale viene creato il database del green pass e dove viene spiegato in un commento la creazione della tabella effettuata invece da linea di comando.

Il serverV, escluso il file per la creazione del database, è l'unico ad interfacciarsi con quest'ultimo, pertanto la compilazione del file C da linea di comando deve essere eseguita in questo modo:

gcc -o creazionedb creazionedb.c `mysql_config --cflags --libs`