Progetto Reti dei Calcolatori

Traccia 2

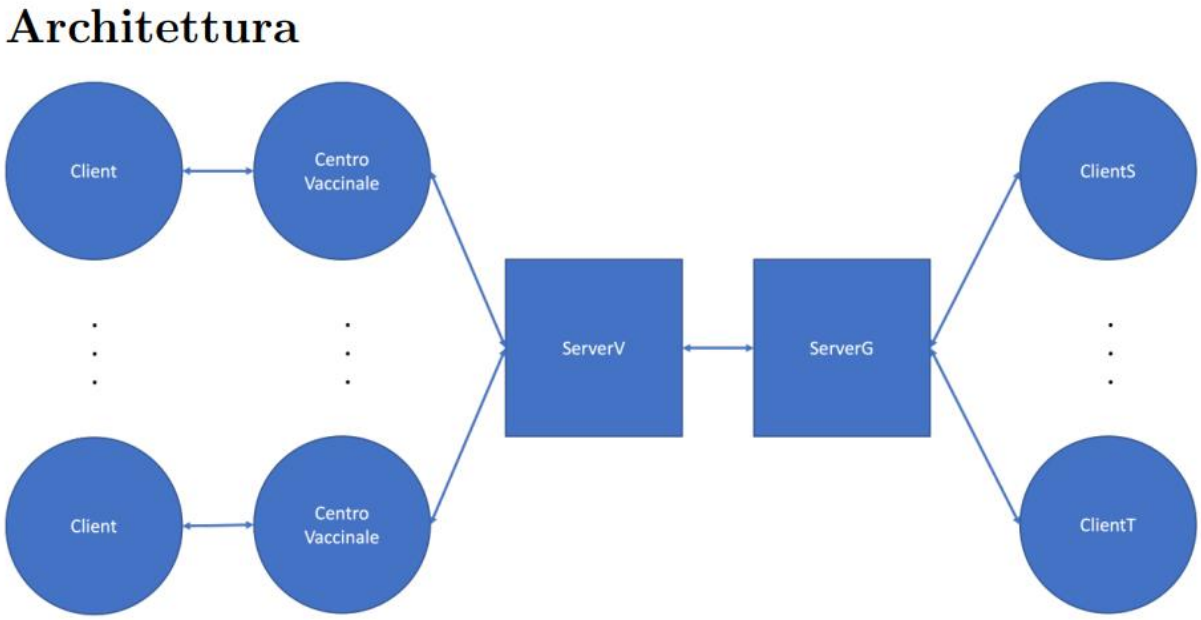
Gianfranco Terrazzano

0124002052

Indice

Descrizione del progetto

Progettare e implementare 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 comunica al ServerV il codice ricevuto dal client ed il periodo di validità del green pass. Un ClientS, per verificare se un green pass `e valido, invia il codice di una tessera sanitaria al ServerG il quale richiede al ServerV il controllo della validità. Un ClientT, inoltre, può invalidare o ripristinare la validità di un green pass comunicando al ServerG il contagio o la guarigione di una persona attraverso il codice della tessera sanitaria.



Il progetto è ispirato al sistema per la gestione della certificazione verde, più comunemente detta green pass. Questo sistema Client-Server permetterà ad un primo Client di comunicare le proprie generalità, comprensive di codice della tessera sanitaria, a un Centro Vaccinale il quale provvederà a generare il periodo di validità della certificazione e comunicarlo al ServerV. Quest'ultimo funzionerà da DataBase/Verificatore di certificazioni verdi.

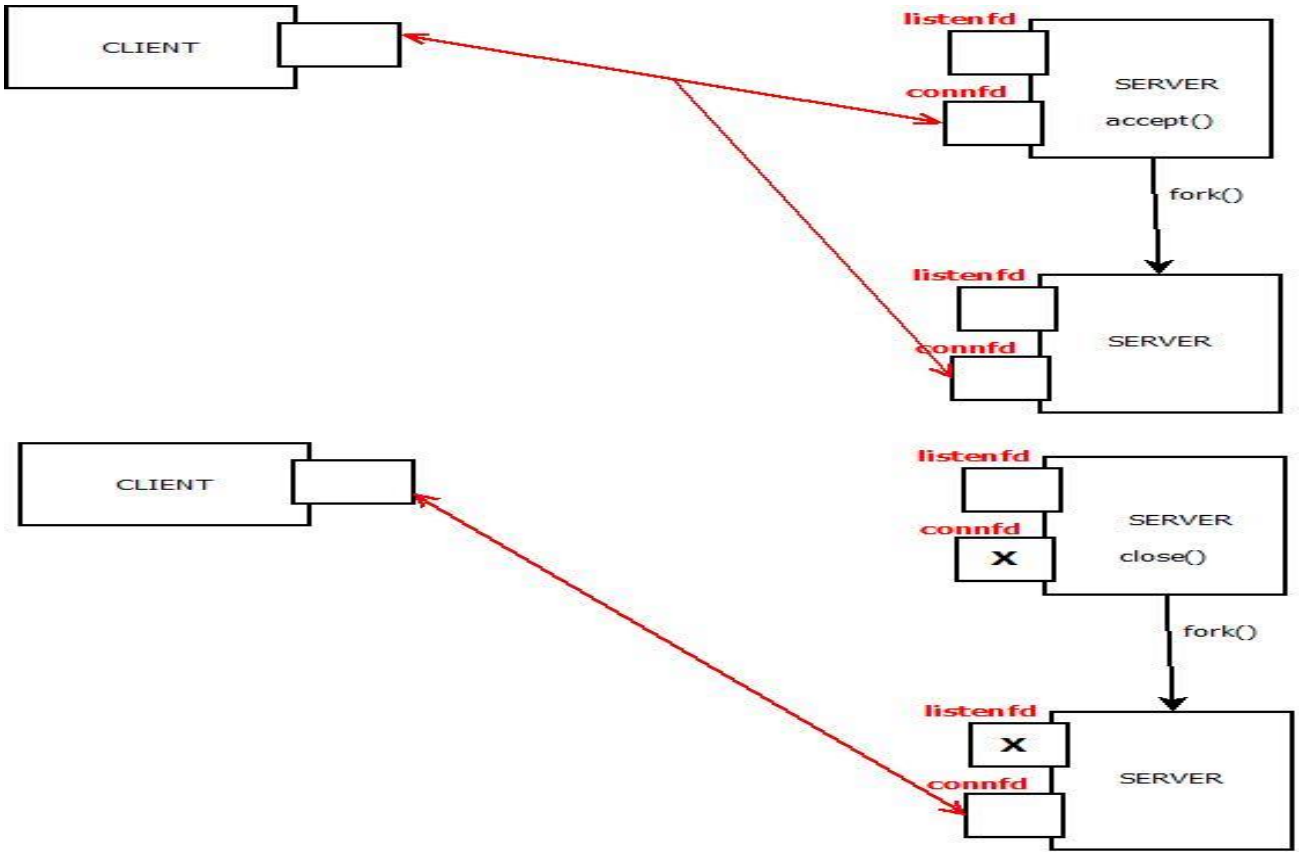
Inoltre in comunicazione con il ServerV avremo il ServerG, il quale è collegato a sua volta con due client:

- Il ClientS -> che chiede la verifica di un codice sanitario, quindi vuole sapere se esso sia valido o meno, al ServerG che comunicando con il ServerV si farà spedire la certificazione in questione e provvederà alla verifica della validità controllando data e check.
- Il ClientT -> Esso avrà la funzionalità di poter invalidare o ripristinare un Green Pass, quindi in input darà un determinato codice sanitario e il nuovo check da impostare, esso sarà passato al ServerG che in comunicazione con il ServerV gli passerà il pacchetto dati, il quale verrà riscritto in memoria. Inoltre verranno ovviamente restituiti messaggi di conferma e avvenuta operazione dal ServerV tramite opportuni controlli.

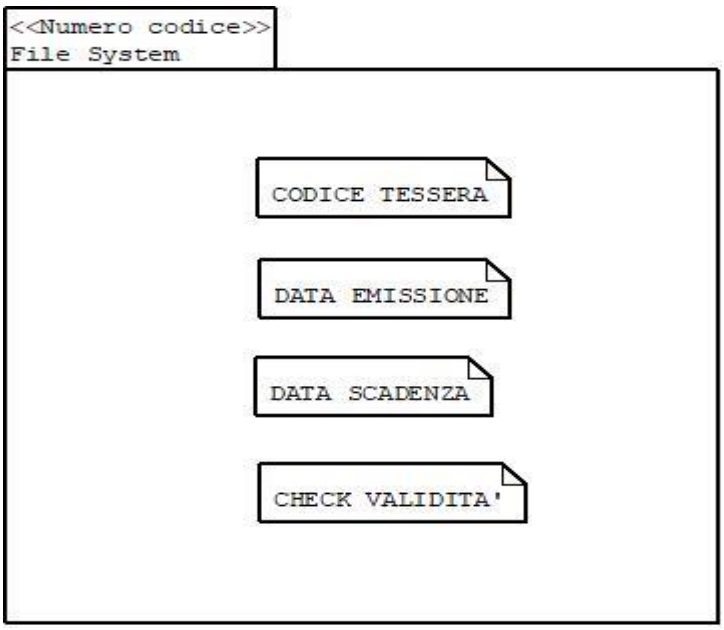
Descrizione e schemi dell'architettura

L'architettura generale del progetto si rifà ovviamente a quella mostrata in traccia.

E' stato utilizzato un **modello** Client-Server, con l'utilizzo di **server concorrenti** in grado di servire più client contemporaneamente tramite l'utilizzo della systemcall() fork, essa permette di creare tanti processi figli quanti sono i client, ai quali viene delegata la gestione di quest'ultimi , mentre il processo padre rimane in attesa di nuove connessioni. Pertanto vengono utilizzati due descrittori diversi uno di listen e uno di connect.



Il ServerV in particolare funzionerà come DB, il quale è stato simulato con un filesystem:



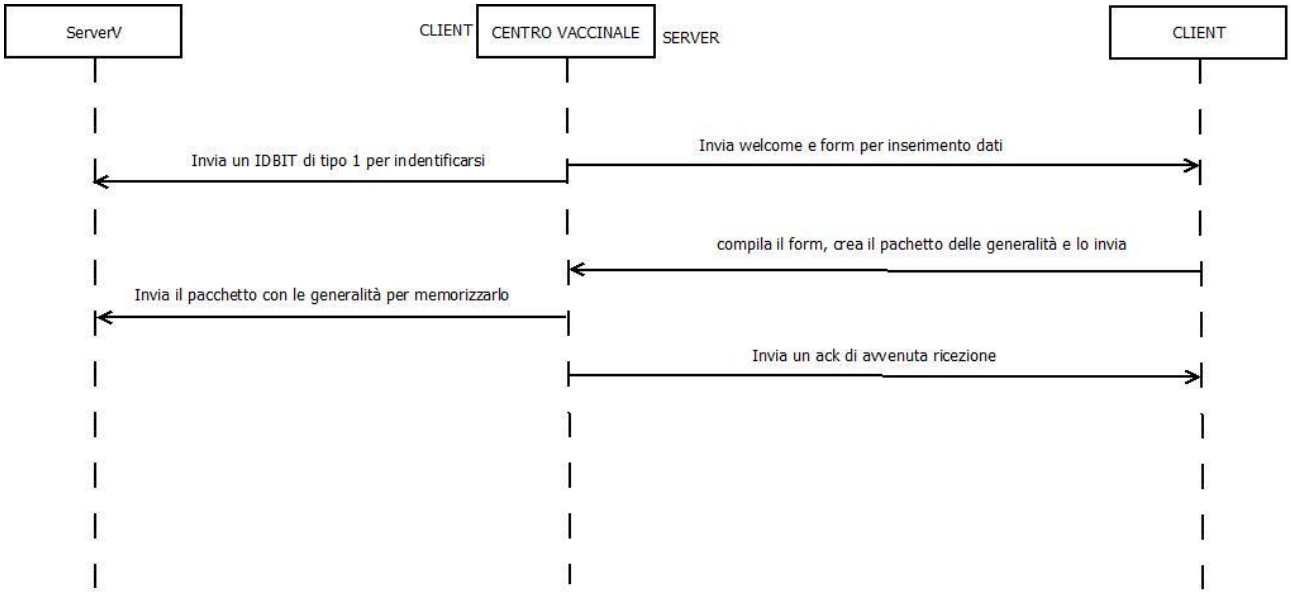
Descrizione e schemi del protocollo applicazione

E' stato utilizzato un protocollo di tipo TCP socket, ovvero ogni socket Client, identificata con l'identificativo della connessione TCP a cui appartiene, è quindi accoppiata con una Socket Server.

Sono stati utilizzati dei pacchetti applicazione, rifacendomi all'esercizio della battaglia navale, il vantaggio di questo approccio è che client e server sanno dove prendere le informazioni e sanno esattamente la dimensione dei dati che si stanno scambiando: il client invierà pacchetto richiesta e il server risponderà con pacchetto risposta. A questo punto, il client e il server non dovranno fare la doppia fullRead e doppia Fullwrite perché conoscono la DIM del pacchetto.

Inoltre i pacchetti sono utili a esplicitare le regole del protocollo applicazione.

Schema ServerV-CentroVaccinale-Client



Pacchetto dati creato dal client ed inviato al CentroVaccinale

```c
//Definiamo il pacchetto dati con
typedef struct{
    char nome[SIZE];
    char cognome[SIZE];
    char COD[COD_SIZE];
} GEN;
```

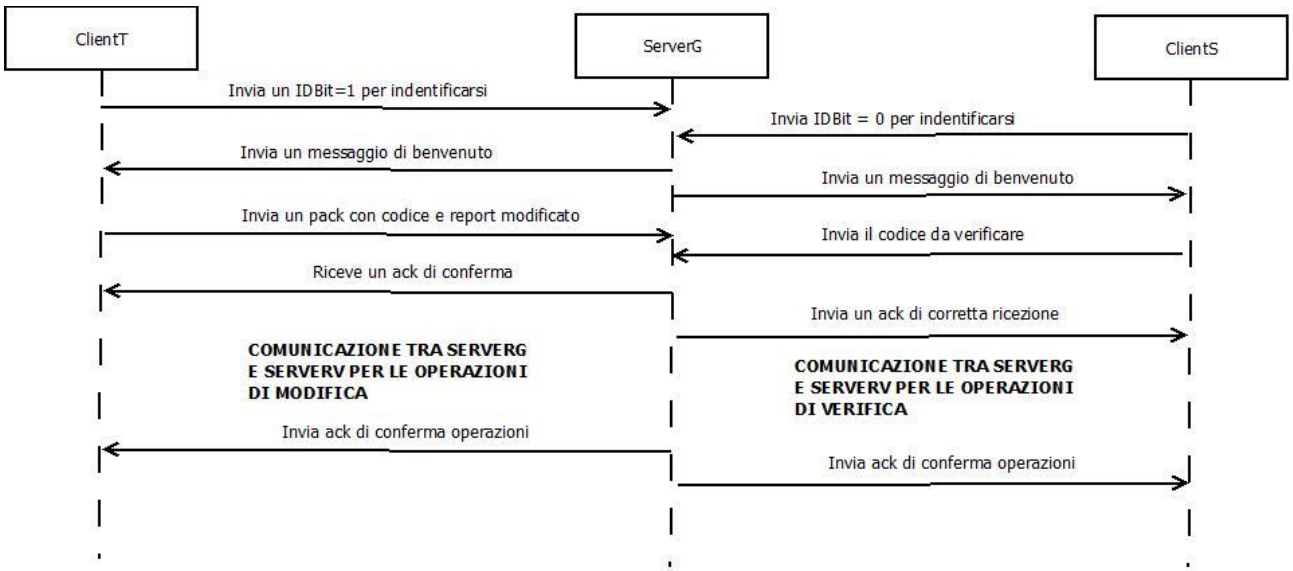
Pacchetti dati utilizzati dal CentroVaccinale per generare il pacchetto al ServerV

```c
//pacchetto dati dell'utente
typedef struct{
    char nome[SIZE];
    char cognome[SIZE];
    char COD[COD_SIZE];
} GEN;

//Struct che serve a formare un tipo data
typedef struct {
    int day;
    int month;
    int year;
} DATE;

//pacchetto da inviare al ServerV
typedef struct{
    char COD[COD_SIZE];
    DATE data_scadenza;
    DATE data_emissione;
}PACKGP;
```

Il pacchetto dati finale arrivato al serverV che verrà memorizzato nel file system

```
typedef struct{
    char COD[COD_SIZE];
    DATE data_scadenza;
    DATE data_emissione;
    char check; //Aggiungiamo un campo che indica la validità del nostro GP
}PACKGP;
```

Schema ClientT-ServerG-ClientS

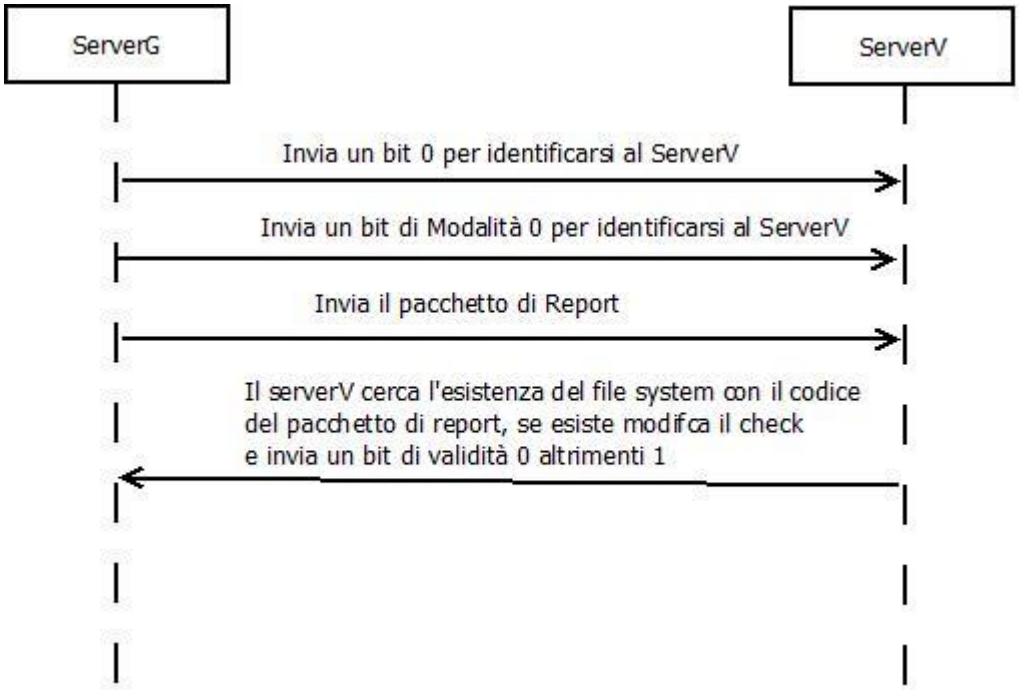


In quanto pacchetti dati per queste operazioni vi è "solo" il pacchetto del ClienT scambiato con il serverG, il quale contiene codice e check di validità.

```
typedef struct  {
    char COD[COD_SIZE];
    char check;
} MODIFYGP;
```

Ovviamente al paragrafo successivo verranno descritte più dettagliatamente le operazioni tra client e server.

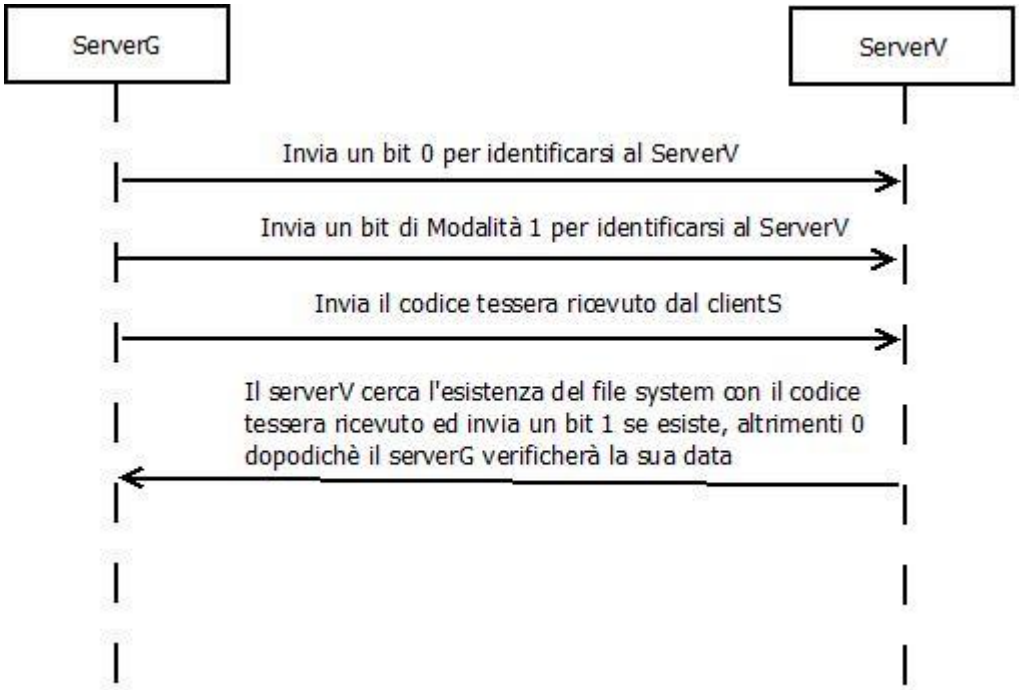
Operazioni ServerG e ServerV per conto del ClientT



Di seguito ci sono i pacchetti utilizzati, il ServerV riceve un modifyGP dal ServerG, controlla l'esistenza del file di nome COD e modifica il pacchetto PACKGP aggiornando il check di validità.

```
typedef struct{
    char COD[COD_SIZE];
    DATE data_scadenza;
    DATE data_emissione;
    char check; //Aggiungiamo un campo che indica la validità del nostro G
}PACKGP;

typedef struct  {
    char COD[COD_SIZE];
    char check;
} MODIFYGP;
```

Operazioni ServerG e ServerS per il ClientS



Dettagli implementativi dei Client

Client

Questo client è un programma scritto in C che utilizza la libreria socket per comunicare con un server, ovvero il CentroVaccinale. Il client invia informazioni generali sull'utente, come il nome, il cognome e il numero di tessera sanitaria, al server. Il client utilizza la funzione gethostbyname per ottenere l'indirizzo IP del server a partire dal nome del server passato come argomento. Quindi, il client crea una struttura GEN che contiene i dati sull'utente e la invia al server tramite la funzione FullWrite. Il client riceve quindi una conferma dal server, chiamata ACK, che indica che i dati sono stati ricevuti correttamente. Il client stampa quindi questa conferma sullo schermo. Successivamente il CentroVaccinale comunicherà con il ServerV.

ClientS

Questo client comunica con il serverG per verificare la validità di un codice rappresentante una tessera sanitaria.

Il client accetta come argomento la tessera sanitaria da verificare. Inizialmente, invia un bit di valore 0 al serverG per identificarsi come client.

Successivamente, il client riceve un messaggio di benvenuto dal serverG e invia il codice della tessera sanitaria. In seguito, attende la conferma dal server che la ricezione è avvenuta correttamente.

Infine, il client riceve l'esito della verifica del codice della tessera sanitaria e lo stampa a schermo.

Le funzioni Socket(), Connect(), FullWrite(), FullRead() sono funzioni personalizzate che gestiscono la creazione del socket, la connessione al server, la scrittura e la lettura sulla connessione socket. La funzione sleep() viene utilizzata per simulare un tempo di attesa durante il caricamento.

ClientT

Questo è un client che si connette al serverG sulla porta 1027 e invia informazioni sul green pass di un individuo. Il codice di tessera sanitaria dell'individuo è passato come argomento da riga di comando.

Innanzitutto, il client imposta la connessione al server specificando l'indirizzo IP "127.0.0.1" e la porta 1027. Quindi, invia un carattere '1' al serverG per identificarsi come ClientT.

Successivamente, il client riceve un messaggio di benvenuto dal serverG e lo stampa a schermo. In seguito, chiede all'utente se il green pass è valido o meno. Se l'utente inserisce un valore valido ('0' o '1'), questo valore viene inviato al server come parte della struttura dati "MODIFYGP".

Infine, il client riceve un messaggio di ack e un messaggio di “report” dal server, simula un caricamento e stampa il messaggio di “report” a schermo. Il client quindi chiude la connessione al server e termina l'esecuzione.

CentroVaccinale(Parte Client)

Descriviamo la funzione che si chiama "InvioDati", essa è un client che invia i dati a un server. La funzione utilizza la struttura di dati "PACKGP" come input. La funzione crea una socket con la famiglia di indirizzi AF_INET(IPV4) e il tipo di socket SOCK_STREAM(TCP), quindi imposta l'indirizzo del server come "127.0.0.1" e la porta 1026 per comunicare con il ServerV.

Successivamente, imposta una variabile "IDbit" a 1 per identificare se stesso al serverV. Utilizza la funzione "FullWrite" per inviare "IDbit" e la struttura di dati "PACKGP" al server. Infine, la funzione chiude la socket.

Dettagli implementativi dei Server

ServerV

Questo è un server che gestisce informazioni su Green Pass.

Il server utilizza una struttura di dati chiamata PACKGP per archiviare informazioni sul Green Pass, come il codice identificativo, la data di scadenza e la data di emissione. La validità del Green Pass è rappresentata da un campo check che può essere impostato su '1' o '0'.

Il server ascolta per le connessioni sulla porta 1026 e accetta nuove connessioni utilizzando la funzione Accept. Quando una connessione è stabilita, viene creato un processo figlio che gestisce la connessione. Il figlio legge un carattere che identifica il tipo di richiesta effettuata dal client.

Se il carattere è '1', il server esegue la funzione CV_recv, che riceve i dati del Green Pass dal client e salva i dati in un file con il nome specificato dal campo COD della struttura PACKGP.

Se il carattere è '0', il server legge un altro carattere per determinare se la richiesta sia di inviare un Green Pass o di modificare la validità di un Green Pass. Se il carattere è '1', il server esegue la funzione InvioGP, che invia il Green Pass richiesto al ServerG. Se il carattere è '0', il server esegue la funzione Mod_GP, che modifica la validità di un Green Pass sulla base della richiesta del ServerG.

Il codice utilizza anche la struttura MODIFYGP per rappresentare una richiesta di modifica della validità del Green Pass e la struttura DATE per rappresentare una data.

ServerG

Questo codice rappresenta un server che gestisce le connessioni sia con il client "ClientT" che con il client "ClientS". Utilizza la libreria wrapper.h, che fornisce alcune funzioni di supporto per la gestione delle socket.

La funzione main() crea un socket, assegna un indirizzo e una porta alla socket e mette in ascolto il socket. Quando viene accettata una connessione, il processo principale crea un processo figlio per gestire la connessione. Il figlio legge un carattere, identificato come "IDbit", dalla connessione per determinare se la connessione proviene dal client "ClientT" o dal client "ClientS". Se viene letto '1', il figlio esegue la funzione "GetDataClientT", altrimenti se viene letto '0', esegue

la funzione "clientS_comm". Se l'ID non è riconosciuto, stampa un messaggio di errore.

La funzione "get_date()" restituisce la data corrente. Questa funzione viene utilizzata per verificare la validità del Green Pass.

La funzione "date_check" prende come input un puntatore a una struttura "PACKGP" che rappresenta il Green Pass. La funzione calcola la data corrente e confronta questa data con la data di scadenza del Green Pass. Se la data corrente è successiva alla data di scadenza, restituisce un carattere che indica che il Green Pass è scaduto. Altrimenti, restituisce un carattere che indica che il Green Pass è ancora valido.

La funzione "VerifyCOD" prende come input una stringa rappresentante il codice del Green Pass e controlla se è valido.

La funzione "clientS_comm" gestisce la comunicazione con il client "ClientS". Questa funzione riceve il codice del Green Pass dal client, controlla se è valido e invia al client un messaggio di conferma o di errore.

La funzione "GetDataClientT" gestisce la comunicazione con il client "ClientT". Questa funzione riceve i dati sul Green Pass dal client, verifica la validità del Green Pass e invia al client un messaggio di conferma o di errore.

La funzione "Mod_GP" prende come input una struttura "MODIFYGP" che rappresenta i dati sul Green Pass da modificare. La funzione effettua la modifica dei dati sul Green Pass e restituisce un carattere che indica se la modifica è stata effettuata con successo o con errore.

CentroVaccinale(Parte Server)

Questo server è configurato per ascoltare su una specifica porta (1024) in attesa di connessioni da parte di utenti, ovvero il Client. Quando un utente si connette, viene generato un processo figlio che gestisce la comunicazione con quell'utente.

La funzione int main() rappresenta il punto di partenza del codice, essendo la funzione principale che avvia il server. Questa funzione fa uso della chiamata a socket() per creare un socket di ascolto. La funzione bind() viene poi utilizzata per associare il socket ad un indirizzo IP e una porta specifici. La funzione listen() viene utilizzata per far sapere che il socket è pronto ad accettare connessioni. Il codice entra in un ciclo infinito di attesa per connessioni, e quando viene accettata una connessione, viene generato un processo figlio che gestisce la comunicazione con l'utente tramite la funzione communication().

Le funzione get_dataE() e get_dataS() calcolano la data di emissione e scadenza facendo uso della libreria time.h e della funzione localtime. Entrambe utilizzano un puntatore alla struct tm e poi valorizzano i campi del pacchetto DATA. La data di emissione è semplicemente il localtime aggiustato con degli spiazzamenti per arrivare al 2023, mentre la data di Scadenza è la localtime più 3 mesi, se essi capitano in due anni diversi, allora si calcola il modulo e si incrementa l'anno.

La funzione communication() gestisce la comunicazione tra il server e il Client. Questa funzione fa uso di un buffer di ricezione per ricevere i dati dall'utente e di un buffer di invio per inviare i dati all'utente. La funzione rand() viene utilizzata per generare un numero casuale da utilizzare come identificativo del centro vaccinale, mentre la funzione InvioDati() è descritta nella parte "client".

Manuale utente

Istruzioni per la compilazione

Libreria: gcc -c -g wrapper.c

Client: gcc -c- g Client.c -> gcc -o client Client.o wrapper.o

ClientT: gcc -c- g ClientT.c -> gcc -o clientT ClientT.o wrapper.o

ClientS: gcc -c- g ClientS.c -> gcc -o client ClientS.o wrapper.o

CentroVaccinale: gcc -c -g CentroVaccinale.c -> gcc -o CV CentroVaccinale.o wrapper.o

ServerV: gcc -c -g ServerV.c -> gcc -o SV ServerV.o wrapper.o

ServerG: gcc -c -g ServerG.c -> gcc -o SG ServerG.o wrapper.o

Istruzioni per la compilazione

Per eseguire digitare ./<nome eseguibile> ovvero il primo parametro dopo gcc -o.

Per un corretto utilizzo si consiglia di seguire questo ordine di esecuzione: ServerV, CentroVaccinale, ServerG, Client, ClientS, ClientT.

Ora seguiranno dei **Test Run**:

Eseguiamo prima i test sul Client con eventuali verifiche di errori:

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./Client localhost
Benvenuto nel centro vaccinale N° : 2!
Digita le tue generalità per inserirle sulla piattaforma.
Inserisci nome: Gianfranco
Inserisci cognome: Terrazzano
Inserisci codice: 01334466889
Il codice è lungo 10 caratteri, riprova!

Inserisci codice:
```

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./Client localhost
Benvenuto nel centro vaccinale N° : 2!
Digita le tue generalità per inserirle sulla piattaforma.
Inserisci nome: Gianfranco
Inserisci cognome: Terrazzano
Inserisci codice: 01334466889
Il codice è lungo 10 caratteri, riprova!

Inserisci codice: 0133446688
I tuoi dati sono stati correttamente inseriti in piattaforma
```

Nel frattempo, il CentroVaccinale

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./CV
In attesa...
In attesa...

Dati ricevuti
Nome: Gianfranco
Cognome: Terrazzano
Numero Tessera Sanitaria: 0133446688

La data di emissione del green pass e': 13:02:2023
La data di scadenza del green pass e': 13/05/2023
```

Ora proviamo a verificare questa tessera sanitaria con il clientS:

Se inseriamo un codice non memorizzato

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientS 0133446689
***Benvenuto nel ServerG! Testeremo la validità del tuo codice della tessera sanitaria

**Il codice è stato ricevuto!
Verifica in corso...

Numero tessera sanitaria non esistente
```

Se inseriamo più di dieci caratteri

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientS 01334466889
Tessera Sanitaria non valida
```

Il Green Pass risulterà valido perché appena aggiunto

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientS 0133446688
***Benvenuto nel ServerG! Testeremo la validità del tuo codice della tessera sanitaria

**Il codice è stato ricevuto!
Verifica in corso...

Green Pass valido, operazione conclusa
```

Nel frattempo, i server si rimetteranno in attesa dopo ogni richiesta:

```
franco@LAPTOP-DGUJMJM4: /mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano
ranco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./SV
n attesa...
n attesa...
n attesa...
n attesa...
```

```
franco@LAPTOP-DGUJMJM4: /mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano
ranco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./SG
n attesa...
@^@^@^@In attesa...
@^@In attesa...
```

(Mi scuso per le chiocciole ed apici, sono a causa degli swift dei desktop su Windows11)

Proviamo attraverso il clientT ad invalidare il Green Pass e a gestire alcuni errori:

```
franco@LAPTOP-DGUJMJM4: /mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientT 0133446689
**Benvenuto al ServerG modificheremo questo green pass
Inserire 0 se il green pass non è valido
Inserire 1 se il gren pass è valido
 Esito: 0
il codice non è associato ad alcun Green Pass!
```

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientT 013344668899
Tessera Sanitaria non valida
```

Con un codice giusto

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientT 0133446688
**Benvenuto al ServerG modificheremo questo green pass
Inserire 0 se il green pass non è valido
Inserire 1 se il gren pass è valido
 Esito: 0
Modifica avvenuta con successo!
```

Proviamo a riverificare il codice con il clientS

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientS 0133446688
***Benvenuto nel ServerG! Testeremo la validità del tuo codice della tessera sanitaria

**Il codice è stato ricevuto!
Verifica in corso...

Green Pass non valido, uscita in corso
```

Ovviamente risulterà non valido, proviamo a rivalidarlo e successivamente a riverificare con il clientT:

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientT 0133446688
**Benvenuto al ServerG modificheremo questo green pass
Inserire 0 se il green pass non è valido
Inserire 1 se il gren pass è valido
 Esito: 1
Modifica avvenuta con successo!
```

```
franco@LAPTOP-DGUJMJM4:/mnt/c/Users/39333/Desktop/Progetto_Gianfranco_Terrazzano$ ./ClientS 0133446688
***Benvenuto nel ServerG! Testeremo la validità del tuo codice della tessera sanitaria

**Il codice è stato ricevuto!
Verifica in corso...

Green Pass valido, operazione conclusa
```

***N.B** La gestione e la persistenza dei dati è stata implementata attraverso il FileSystem per comodità progettuale. Sono consapevole che un database avrebbe garantito una migliore sicurezza e persistenza dei dati oltre a gestire autonomamente la mutua esclusione, ma dato che non era richiesto nei vincoli progettuali ho preferito utilizzare un FileSystem per semplicità. Per implementare la concorrenza ho utilizzato la systemcall (flock()).