

# **PROGETTO RETI DI CALCOLATORI**

**A.A 2022\23**



**Proponente**

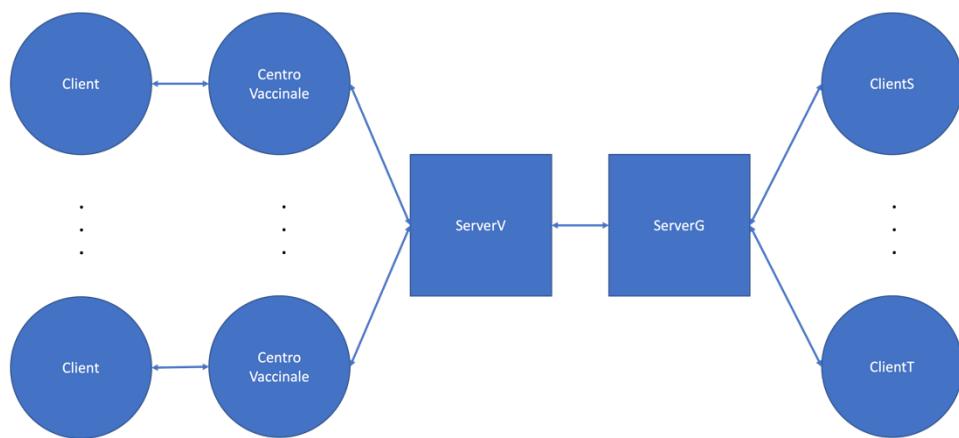
Vincenzo Salzano

# SOMMARIO

Informazioni generali.....	1
Informazioni client e server .....	2
Utente .....	2
CentroVaccinale .....	4
ServerV .....	8
ServerG.....	11
ClientS .....	15
ClientT .....	15
Descrizione Modello e Protocollo.....	16
Manuale utente.....	16
Istruzioni Compilazione.....	17
Istruzioni Esecuzione .....	18
Esempio D'Uso .....	19
Conclusione.....	22

# INFORMAZIONI GENERALI

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 è 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.



Quest'ultimo viene realizzato mediante linguaggio C ed è pronto ad essere eseguito su qualsiasi piattaforma Unix. La realizzazione di questo progetto inoltre è avvenuta su un MacBook Air m1.

# INFORMAZIONI CLIENT E SERVER

Di seguito verranno riportate le informazioni specifiche sui Client e i Server del progetto, esperendo nel miglior modo possibile scopo e collegamenti tra quest'ultimi.

## UTENTE

In questo paragrafo andrò ad illustrare le azioni che vengono effettuate dall'utente (in quanto persona fisica) e dall'utente (in quanto Client).

Nel primo caso l'Utente può svolgere molteplici azioni all'interno della piattaforma. Esso inizialmente riceve un messaggio di benvenuto, in cui viene mostrato il Centro Vaccinale nel quale è stato indirizzato; successivamente gli verranno chieste tutte le generalità, per poi poter completare la registrazione.

In quanto Client, inizialmente vado a creare un pacchetto, che denomino VAX, in cui vado ad inserire le generalità che devono essere richieste affinché si possa completare la vaccinazione. Quest'ultimo ho preferito spostarlo in un Header file, in cui si trovano: librerie, costanti, struct e funzioni riutilizzate da più, o tutti, i file.c del progetto.

Di seguito la struct VAX:

```
//Struct del pacchetto da mandare al CentroVaccinale con: nome, cognome ed identificativo della tessera sanitaria
typedef struct {
    char name[MAX_SIZE];
    char surname[MAX_SIZE];
    char ID[ID_SIZE];
} VAX;
```

Successivamente si crea una function per la creazione del pacchetto da inviare al Centro Vaccinale, in cui l'Utente da tastiera inserisce le generalità:

```
//Function per la creazione del pacchetto da inviare al CentroVaccinale
//Si utilizza un buffer nel quale sono contenute le stringhe, contenenti le generalità, da far leggere poi al CentroVaccinale
VAX create_package() {
    char buffer[MAX_SIZE];
    VAX create_pack;

    //Utente inserisce il nome
    printf("Nome Utente: ");
    if (fgets(create_pack.name, MAX_SIZE, stdin) == NULL) {
        perror("fgets() error");
    }

    //Si va ad inserire il terminatore al posto dell'invio inserito dalla fgets, poichè questo veniva contato ed inserito come carattere nella stringa
    create_pack.name[strlen(create_pack.name) - 1] = 0;

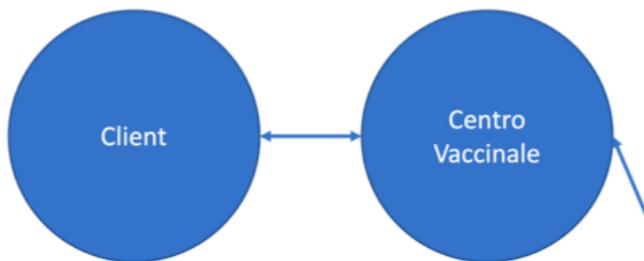
    //Utente inserisce il cognome
    printf("Cognome Utente: ");
    if (fgets(create_pack.surname, MAX_SIZE, stdin) == NULL) {
        perror("fgets() error");
    }

    //Si va ad inserire il terminatore al posto dell'invio inserito dalla fgets, poichè questo veniva contato ed inserito come carattere nella stringa
    create_pack.surname[strlen(create_pack.surname) - 1] = 0;

    //Si inserisce l'identificativo(codice) della tessera sanitaria
    while (1) {
        printf("Inserisci l'identificativo della tessera sanitaria(MAX 10 caratteri): ");
        if (fgets(create_pack.ID, MAX_SIZE, stdin) == NULL) {
            perror("fgets() error");
            exit(1);
        }

        //Nel caso in cui il numero di caratteri inseriti da tastiera fosse diverso dai 10 prestabiliti, si stampi il messaggio
        if (strlen(create_pack.ID) != ID_SIZE) printf("Il numero di caratteri della tessera sanitaria è errato, si prega di inserire esattamente 10 cifre!\n\n");
        else {
            //Si va ad inserire il terminatore al posto dell'invio inserito dalla fgets, poichè questo veniva contato ed inserito come carattere nella stringa
            create_pack.ID[ID_SIZE - 1] = 0;
            break;
        }
    }
    return create_pack;
}
```

Come si può notare sia dal punto di vista implementativo che dal punto di vista progettistico, il client (Utente) comunica direttamente solo con il Centro Vaccinale.



# CENTRO VACCINALE

Il Centro Vaccinale successivamente alla compilazione delle generalità fornite dall'utente, comunica ad esso la corretta ricezione del pacchetto VAX (che contiene nome, cognome e id dell'utente).

Una volta ottenuti i dati, il Centro Vaccinale ha il compito di generare il GreenPass, attribuendogli una data di emissione e una data di scadenza. Per fare ciò vado a creare due struct: una per poter generare una Data (con giorno, mese e anno; successivamente mostrerò come convertire il formato data in intero) e un struct riguardante il GreenPass. Quest'ultime contenute sempre all'interno di un Header file:

```
//Struct data(giorno,mese,anno)
typedef struct {
    int day;
    int month;
    int year;
} DATE;

//Struct del pacchetto inviato dal CentroVaccinale con id utente, data inizio gp, data fine gp
typedef struct {
    char ID[ID_SIZE];
    char report;
    DATE start_date;
    DATE end_date;
} GREEN_PASS;
```

Si creano quindi due function: la prima per impostarere la data d'inizio del GreenPass e una per la data di scadenza.

Per quanto riguarda la data di emissione ci serviamo della data odierna, ricavandola dall'orario della macchina, per poi convertirla in intero:

```
//Function che calcola la data di inizio di validità del GreenPass, cioè quando viene emesso il certificato
void create_start_date(DATE *start_date) {
    time_t time_tick;
    time_tick = time(NULL);

    //Function di localtime che permette di convertire un formato data a intero
    struct tm *s_date = localtime(&time_tick);
    //Si parte con i mesi di tm da 0
    s_date->tm_mon += 1;
    //Si somma 1900 perchè si contano 123 anni, di conseguenza per arrivare al 2023 impostiamo 1900
    s_date->tm_year += 1900;

    printf("Il GreenPass è stato emesso in data : %02d:%02d:%02d\n", s_date->tm_mday, s_date->tm_mon, s_date->tm_year);

    //Si assegnano i valori ai parametri di ritorno
    start_date->day = s_date->tm_mday ;
    start_date->month = s_date->tm_mon;
    start_date->year = s_date->tm_year;
}
```

Utilizziamo il medesimo procedimento per la data di scadenza. Tuttavia, in questo caso è necessario gestire la variazione dell'anno. La durata del GreenPass è stata impostata a 6 mesi, ciò comporta che i mesi che vanno da luglio in poi devono essere gestiti in modo differente. Ad esempio: se l'emissione del mio GreenPass risale al 3 luglio 2023, il GreenPass scadrà il 3 Gennaio 2023; di conseguenza è in un mese identificabile, ragionando per assurdo e in modo del tutto teorico, al tredicesimo mese dell'anno (di conseguenza il primo dell'anno nuovo). Tale procedimento viene poi eseguito per tutti i mesi che vanno da luglio in poi:

```
//Function che definisce la scadenza del GreenPass
void create_end_date(DATE *end_date) {
    time_t time_tick; //struttura per la gestione della data
    time_tick = time(NULL); //Utilizziamo l'ora attuale per poi assegnarla alla variabile

    //Funzione di localtime che permette di convertire un formato data a intero
    struct tm *t_date = localtime(&time_tick);
    //Si somma 7 perchè il conteggio dei mesi va da 0 ad 11 (quindi impostiamo 6 mesi alla scadenza)
    t_date->tm_mon += 7;
    //Si somma 1900 perchè si contano 123 anni, di conseguenza per arrivare al 2023 impostiamo 1900
    t_date->tm_year += 1900;

    //Si effettua un controllo nel caso in cui il vaccino sia stato fatto nei mesi in questione, provocando un aumento dell'anno
    //luglio
    if (t_date -> tm_mon == 13){ //13 perchè si riferisce ad un improbabile 13simo mese, attribuibile
        t_date -> tm_mon = 1; //al primo(1) mese dell'anno successivo
        t_date -> tm_year++;
    }

    //agosto
    if (t_date -> tm_mon == 14){ //Si esegue la stessa azione vista per il primo if, anche negl'if successivi
        t_date -> tm_mon = 2;
        t_date -> tm_year++;
    }

    //settembre
    if (t_date -> tm_mon == 15){
        t_date -> tm_mon = 3;
        t_date -> tm_year++;
    }

    //ottobre
    if (t_date -> tm_mon == 16){
        t_date -> tm_mon = 4;
        t_date -> tm_year++;
    }

    //novembre
    if (t_date -> tm_mon == 17){
        t_date -> tm_mon = 5;
        t_date -> tm_year++;
    }

    //dicembre
    if (t_date -> tm_mon == 18){
        t_date -> tm_mon = 6;
        t_date -> tm_year++;
    }

    printf("Il GreenPass scadrà in data : %02d:%02d:%02d\n", t_date->tm_mday, t_date->tm_mon, t_date->tm_year);

    //Si assegnano i valori definiti da tale funzione con i vari if nella struct:
    end_date->day = t_date->tm_mday ;
    end_date->month = t_date->tm_mon;
    end_date->year = t_date->tm_year;
}
```

Una volta create Data di emissione e di scadenza, il Centro Vaccinale si connette al ServerV, spedendo i dati ottenuti ed inviando 1 bit per far capire che la comunicazione sta avvenendo con il Centro Vaccinale:

```
//Function per inviare il GreenPass al ServerV
void gp_dispatch(GREEN_PASS gp) {
    int socket_fd;
    struct sockaddr_in server_addr;
    char bit_communication, buffer[MAX_SIZE];

    //inizializziamo il bit a 1 da inviare al ServerVaccinale
    bit_communication = '1';

    //Creazione descrizione del socket
    if ((socket_fd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
        perror("socket() error");
        exit(1);
    }

    //Strutture
    server_addr.sin_family = AF_INET;
    server_addr.sin_port = htons(1036); //porta

    //Conversione formato dell'indirizzo IP
    if (inet_pton(AF_INET, "127.0.0.1", &server_addr.sin_addr) <= 0) {
        perror("inet_pton() error");
        exit(1);
    }

    //Si effettua la connessione con il server
    if (connect(socket_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) < 0) {
        perror("connect() error");
        exit(1);
    }

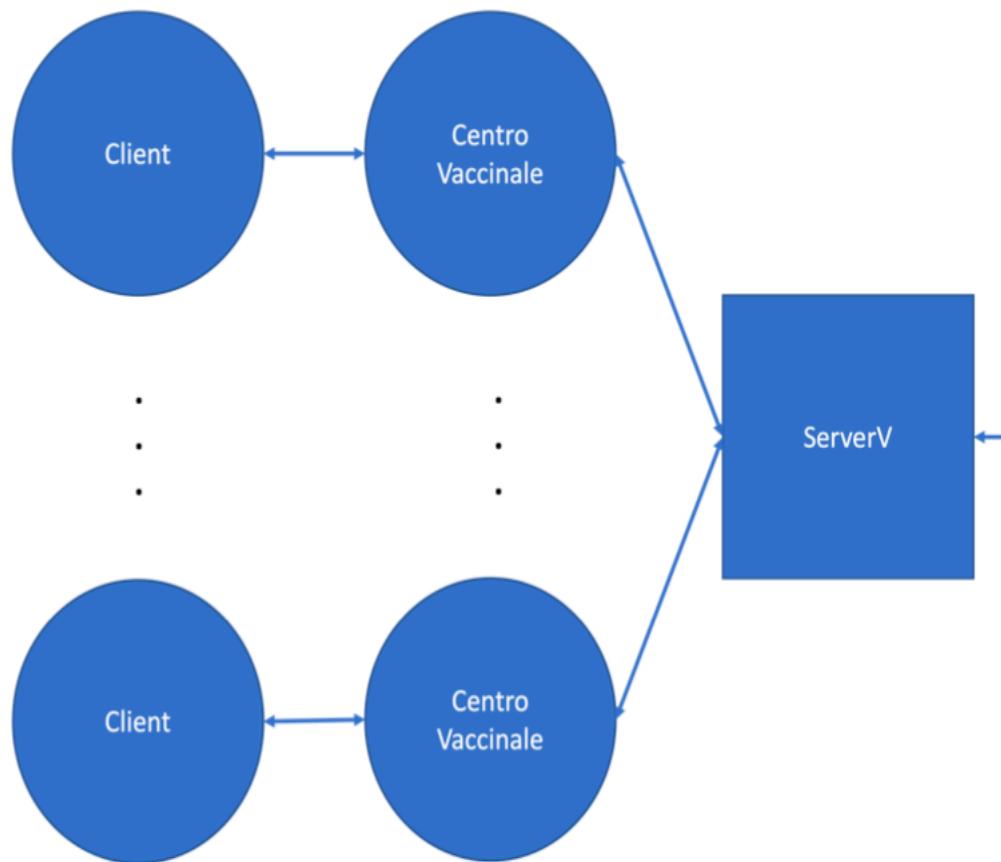
    //Invia un bit di valore 1 al ServerV per informarlo che la comunicazione deve avvenire con il CentroVaccinale
    if (full_write(socket_fd, &bit_communication, sizeof(char)) < 0) {
        perror("full_write() error");
        exit(1);
    }

    //Si invia il green pass al ServerV
    if (full_write(socket_fd, &gp, sizeof(gp)) < 0) {
        perror("full_write() error");
        exit(1);
    }

    close(socket_fd);
}
```

Successivamente si può generare il GreenPass, con i dati mandati dall'Utente, e reindirizzare randomicamente quest'ultimo ad un Centro Vaccinale.

Come si può notare sia dal punto di vista implementativo che dal punto di vista progettistico, il Centro Vaccinale comunica direttamente solo con il Client (Utente) e il ServerV.



# SERVERT

Il ServerV occupa una funzione molto importante all'interno del progetto, siccome simula e svolge la funzione di Database. In questo caso ho optato però per un filesystem, affinché potessi tener conto di tutti i dati generati e potessi gestire con maggior semplicità gli accessi concorrenti. Difatti, in questo paragrafo, verrà mostrato come sono stati gestiti alcuni accessi concorrenti.

Inoltre, il ServerV deve gestire la comunicazione sia con il Centro Vaccinale, da cui riceve il pacchetto GreenPass, sia con il ServerG, a cui spedisce il pacchetto. Per regolare tale scambio di informazioni, si utilizza un bit di comunicazione che: se pari ad 1 gestisce la connessione con il CentroVaccinale, se pari a 0 gestisce la connessione con il ServerG.

Il ServerV, come spiegato in precedenza, riceve il pacchetto denominato GreenPass dal Centro Vaccinale. Una volta ricevuto lo apre per controllare l'Id ricevuto in quel pacchetto, e nel caso in cui quell'Id esistesse il pacchetto viene inviato al ServerG.

```
//Funzione che invia un GP richiesto dal ServerG
void gp_dispatch(int connect_fd) {
    char report;
    char ID[ID_SIZE];
    int fd;
    GREEN_PASS gp;

    //Scrive in ID l'identificativo della tessera sanitaria ricevuta dal ServerG da connect_fd
    if (full_read(connect_fd, ID, ID_SIZE) < 0) {
        perror("full_read() error");
        exit(1);
    }
    //Si apre il file rinominato ID, cioè l'identificativo inviato dal ServerG
    fd = open(ID, O_RDONLY);
```

Tuttavia, in questa parte di codice vi si deve accedere in maniera mutuamente esclusiva alla sezione critica, poiché si deve evitare che nel momento in cui un processo sta leggendo l'Id del pacchetto Green Pass, passatogli dal Centro Vaccinale, possa poi essere sovrascritto da ulteriori processi.

```
/*
Nel caso in cui l'identificativo della tessera sanitaria inviata dal ClientS non dovesse esistere, la variabile globale errno catturerà ciò
inviando un report uguale ad 1 al ServerG, il quale aggiornerà il ClientS dell'inesistenza di quest'ultimo.
Nel caso in cui l'identificativo della tessera sanitaria esistesse, errno invierà un report uguale a 0, che sta ad indicare che
l'operazione è avvenuta correttamente
*/
//errno = 2 se la open è fallita non avendo trovato alcun numero di tessera sanitaria ("No such file or directory")
if (errno == 2) {
    printf("L'identificativo della tessera sanitaria non esiste, si prega di riprovare...\\n");
    report = '2';

    //Invia tale report al ServerG
    if (full_write(connect_fd, &report, sizeof(char)) < 0) {
        perror("full_write() error");
        exit(1);
    }
} else {
    //Si accede in maniera mutuamente esclusiva, cioè LOCK_EX definisce che solo un processo alla volta può accedere al blocco
    if (flock(fd, LOCK_EX) < 0) {
        perror("flock() error");
        exit(1);
    }
    //Lettura del GreenPass del file aperto in precedenza
    if (read(fd, &gp, sizeof(GREEN_PASS)) < 0) {
        perror("read() error");
        exit(1);
    }
    //Si sblocca il lock, con LOCK_UN che rimuove la function flock con LOCK_EX definita in precedenza
    if(flock(fd, LOCK_UN) < 0) {
        perror("flock() error");
        exit(1);
    }
    //chiudiamo il file
    close(fd);
    report = '1';

    //Si invia il report al ServerG
    if (full_write(connect_fd, &report, sizeof(char)) < 0) {
        perror("full_write() error");
        exit(1);
    }
}

//Si manda il GreenPass richiesto al ServerG che controllerà la sua validità
if(full_write(connect_fd, &gp, sizeof(GREEN_PASS)) < 0) {
    perror("full_write() error");
    exit(1);
}
}
```

Menzionando precedentemente la scelta del filesystem e della gestione della concorrenza, è stata utilizzata la systemcall flock(), per gestire la concorrenza in lettura e scrittura.

Inoltre, il ServerV svolge anche azioni di modifica dei referti, avviando una comunicazione con il ClientS mentre il ServerG è in attesa. Per modificare i referti vi si deve accedere, come in precedenza, in maniera mutuamente esclusiva:

```
//Riceve il pacchetto REPORT dal ServerG proveniente dal ClientT, contenente l'identificativo della tessera sanitaria e il referto del tampone
if (full_read(connect_fd, &package, sizeof(REPORT)) < 0) {
    perror("full_read() error");
    exit(1);
}

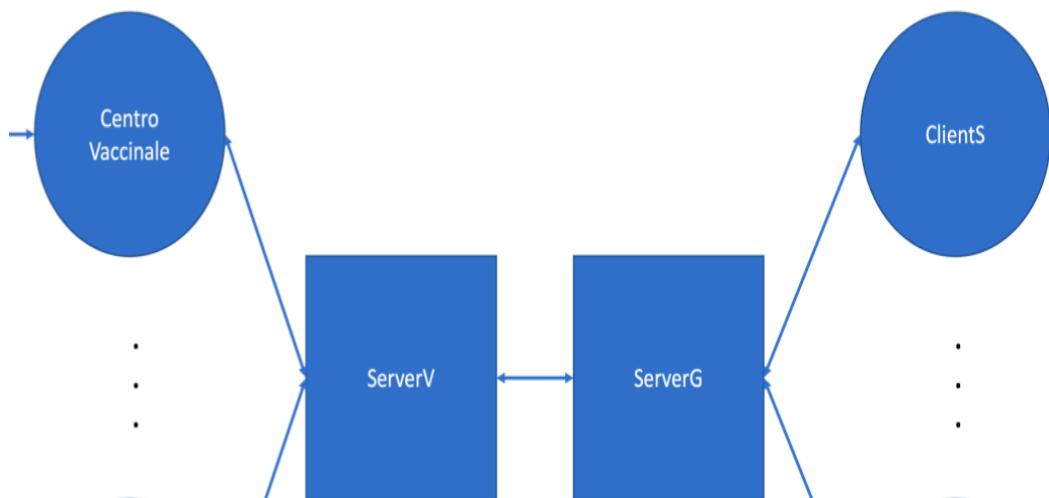
//Si apre il file contenente il GreenPass relativo all'identificativo della tessera ricevuto dal ClientT
fd = open(package.ID, O_RDWR);

/*
Nel caso in cui l'identificativo della tessera sanitaria inviata dal ClientT non dovesse esistere, la variabile globale errno catturerà ciò
invia un report uguale ad 1 al ServerG, il quale avverrà il ClientT dell'inesistenza di quest'ultimo.
Nel caso in cui l'identificativo della tessera sanitaria esistesse, errno invierà un report uguale a 0, che sta ad indicare che
l'operazione è avvenuta correttamente
*/
if (errno == 2) {
    printf("L'identificativo della tessera sanitaria non esiste, si prega di riprovare...\n");
    report = '1';
} else {
    //Si accede in maniera mutuamente esclusiva
    if(flock(fd, LOCK_EX | LOCK_NB) < 0) { //LOCK_NB ritorno immediato
        perror("flock() error");
        exit(1);
    }
    //Si legge il file aperto contenente il GreenPass relativo all'identificativo della tessera ricevuta dal ClientT
    if (read(fd, &gp, sizeof(GREEN_PASS)) < 0) {
        perror("read() error");
        exit(1);
    }

    //Si assegna al report del GreenPass associato all'ID del report che quest'ultimo vogliono modificare
    gp.report = package.report;
    //Si ritorna all'inizio dello stream del file
    lseek(fd, 0, SEEK_SET);

    //Si va a sovrascrivere i campi del GreenPass nel file binario con nome il numero di tessera sanitaria del green pass
    if (write(fd, &gp, sizeof(GREEN_PASS)) < 0) {
        perror("write() error");
        exit(1);
    }
    //Sblocciamo il lock
    if(flock(fd, LOCK_UN) < 0) {
        perror("flock() error");
        exit(1);
    }
    report = '0';
}
//Si invia il report al ServerG
if (full_write(connect_fd, &report, sizeof(char)) < 0) {
    perror("full_write() error");
    exit(1);
}
}
```

Come si può notare dal punto di vista progettistico il ServerV comunica direttamente con Centro Vaccinale e ServerG, mentre dal punto di vista implementativo comunica anche con il ClientS, nel momento in cui il ServerG risulta in attesa.



## SERVERG

Il ServerG è il server del progetto che instaura più connessioni e per il quale passano più informazioni. Inizialmente il ServerG resta in attesa. Una volta instaurata poi una connessione con il ServerV confronta la data odierna, presa dalla macchina, con quella del GreenPass. Questo confronto avviene affinché possa esserci un controllo sulla validità del GreenPass, risultando ancora valido.

Di seguito una porzione di codice che riporta quest'ultima operazione:

```
//Si riceve l'esito della verifica dal ServerV e:se 0 è non valido,se 1 è valido
if (report == '1') {
    //Si estrae il GreenPass associato alla tessera sanitaria inviata dal ClientS
    if (full_read(socket_fd, &gp, sizeof(GREEN_PASS)) < 0) {
        perror("full_read() error\n");
        exit(1);
    }

    //Si chiuda la connessione dopo aver ottenuto il GP associato
    close(socket_fd);

    //Function per ricavare la data odierna
    create_today_date(&today_date);

    /*Si veda se il GreenPass è concretamente valido dopo averlo confrontato con la data odierna
    Effettuiamo quindi il controllo anche mediante l'uso della data di fine del GreenPass
    */
    if (today_date.year > gp.end_date.year) report = '0';
    if (report == '1' && today_date.year > gp.end_date.year && today_date.month > gp.end_date.month) report = '0';
    if (report == '1' && today_date.day > gp.end_date.year && today_date.month > gp.end_date.month && today_date.day > gp.end_date.day) report = '0';
    //Si fa un controllo se è presente il GreenPass di un utente positivo(gp.report == 0)
    if (report == '1' && gp.report == '0') report = '0';
}
return report;
```

Il ServerG inoltre si occupa, come accennato nel paragrafo precedente, anche di prendere e spedire il report (il referto), con annesse modifiche o meno, facendo quindi da tramite tra il ServerV e il ClientS.

Oltre ad instaurare connessioni con il ServerV, il ServerG instaura connessioni anche con il ClientT. Per quest'ultimo, il ServerG ha il compito di controllare l'Id quando l'Utente, mediante l'utilizzo dei servizi del ClientT, inserisce da tastiera l'identificativo della propria tessera sanitaria.

In questo caso il ServerG, una volta instaurata una connessione con il ClientT, confronta gli Id e se l'Id risultasse valido, informa l'utente se il GreenPass è valido o meno.

Di seguito una porzione di codice della function che svolge tale compito:

```
//Si stampa un messaggio di benvenuto da inviare al ClientS dopo la connessione con il ServerG
sprintf(buffer, WELCOME_SIZE, "[Le diamo il benvenuto nell'App CampaniaGP]\n");
buffer[WELCOME_SIZE - 1] = 0;
//Si manda il messaggio scritto in precedenza
if(full_write(connect_fd, buffer, WELCOME_SIZE) < 0) {
    perror("full_write() error");
    exit(1);
}

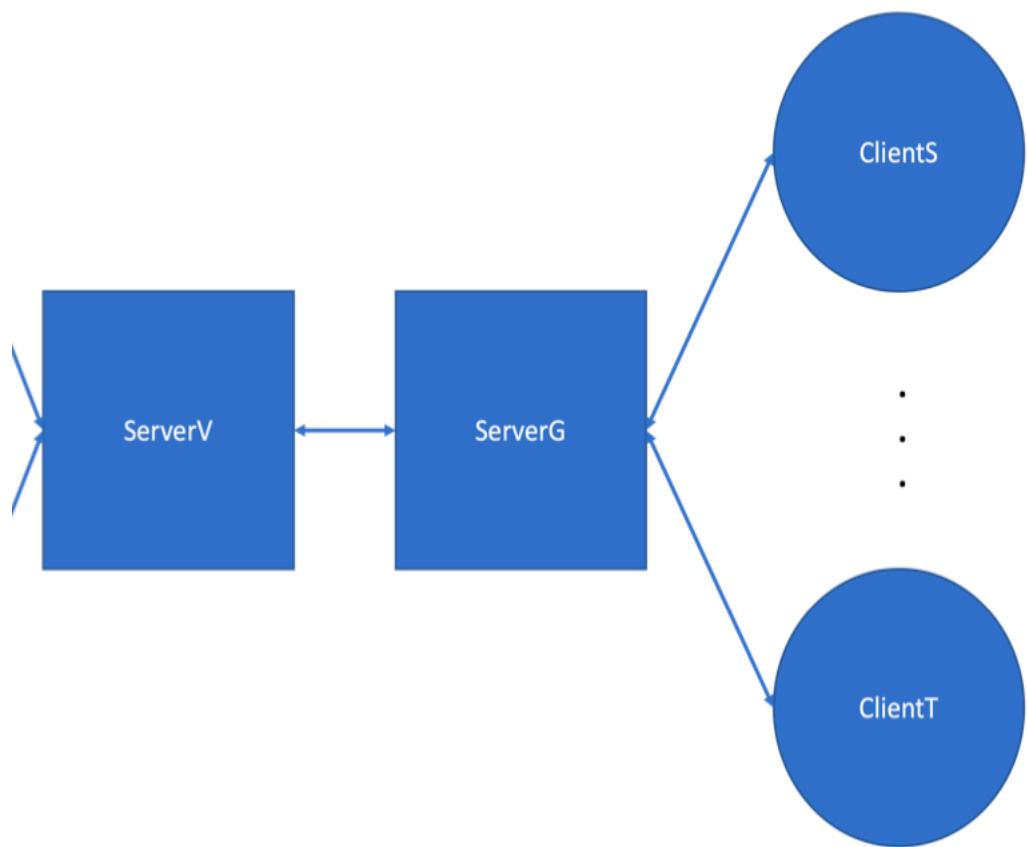
//Si legge il codice fiscale inviato dal ClientS
if(full_read(connect_fd, ID, ID_SIZE) < 0) {
    perror("full_read error");
    exit(1);
}

//Si notifica all'utente la corretta ricezione dei dati che aveva inviato
sprintf(buffer, MEX_SIZE, "L'identificativo della tessera sanitaria è corretto... \n");
buffer[MEX_SIZE - 1] = 0;
if(full_write(connect_fd, buffer, MEX_SIZE) < 0) {
    perror("full_write() error");
    exit(1);
}

//Function che invia l'identificativo della tessera sanitaria al ServerV e riceve il report dal ServerG
report = check_id(ID);

//Si invia il report di validità del green pass al ClientS, per verificarne anche l'esistenza di quest'ultimo
if (report == '1') {
    strcpy(buffer, "Il GreenPass è valido! \n");
    if(full_write(connect_fd, buffer, CT_ACK) < 0) {
        perror("full_write() error");
        exit(1);
    }
} else if (report == '0') {
    strcpy(buffer, "Il GreenPass non è valido! \n");
    if(full_write(connect_fd, buffer, CT_ACK) < 0) {
        perror("full_write() error");
        exit(1);
    }
} else {
    strcpy(buffer, "Ops, l'id non esiste! \n");
    if(full_write(connect_fd, buffer, CT_ACK) < 0) {
        perror("full_write() error");
        exit(1);
    }
}
//Si chiude la connessione dopo il check di validità del GreenPass
close(connect_fd);
```

Come si può notare sia dal punto di vista implementativo che dal punto di vista progettistico, il ServerG comunica direttamente con il ServerV, il ClientS e il ClientT.



## **CLIENTS (APP CAMPANIAGP)**

Il ClientS, nel mondo reale, rappresenta quella che potrebbe essere un'app, come per esempio "e-covid", in cui possiamo controllare i dati del GreenPass e la sua validità. Ciò avviene ovviamente immettendo i nostri dati, in questo caso l'identificativo della tessera sanitaria.

Da un punto di vista progettistico e implementativo instaura connessioni solo con il ServerG

## **CLIENTT (CAMPANIA GREENPASS)**

Il ClientT, nel mondo reale, rappresenta quella che potrebbe essere una società privata, o un ente pubblico, che gestisce la validità del GreenPass in base ai referti, che possono essere positivi o negativi.

Per il ClientT usiamo la seguente struct, anch'essa contenuta nell'Header file:

```
//Struct del pacchetto del ClientT contenente ID e referto di validità
typedef struct {
    char ID[ID_SIZE];
    char report;
} REPORT;
```

Da un punto di vista progettistico il ClientT instaura connessioni con il ServerG, da un punto di vista implementativo instaura connessioni anche con il ServerV.

## **DESCRIZIONE MODELLO E PROTOCOLLO**

Come definito dalla traccia scelta, il modello di programmazione scelto è il client-server.

I server che sono stati utilizzati sono concorrenti, cioè offrono servizi a più client contemporaneamente. Ciò avviene grazie all'utilizzo della systemcall fork(), che crea N processi figli che sono pari poi alle connessioni dei vari client con il server.

Il padre gestisce il descrittore listenfd che accetta le richieste di connessione, il figlio invece gestisce il descrittore connfd che fornisce servizi ai client connessi.

Il protocollo utilizzato invece è il TCP, siccome il progetto che è stato sviluppato richiede un servizio orientato alla connessione. Tale protocollo prima di poter trasmettere dei dati deve stabilire una connessione.

## **MANUALE UTENTE**

Di seguito verrà mostrata una guida riguardante la compilazione e l'esecuzione del progetto. Al termine di quest'ultima verrà mostrata una simulazione del progetto.

# **ISTRUZIONI COMPILAZIONE**

Per poter compilare da terminale, recarsi nella cartella "CampaniaGP\_codice" e digitare i seguenti comandi:

1. gcc -o ServerV ServerV.c
2. gcc -o CentroVaccinale CentroVaccinale.c
3. gcc -o ServerG ServerG.c
4. gcc -o Utente Utente.c
5. gcc -o ClientS ClientS.c
6. gcc -o ClientT ClientT.c

# **ISTRUZIONI ESECUZIONE**

Per eseguire da terminale digitare i seguenti comandi:

1. ./ServerV
2. ./CentroVaccinale
3. ./ServerG
4. ./Utente localhost
5. ./ClientS
6. ./ClientT

Per evitare di incorrere in inconvenienti durante l'esecuzione, si consiglia di eseguire i file nel seguente ordine:

ServerV, CentroVaccinale, ServerG, Utente, ClientS, ClientT

# ESEMPIO D'USO

In questo paragrafo verrà mostrata una simulazione, tramite delle schermate, del progetto.

Inizialmente si esegue il ServerV:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice  
[10-22-54-96:CampaniaGP_codice vinces$ ./ServerV  
In attesa di nuove connessioni...  
█
```

Poi si esegue il CentroVaccinale:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice  
[10-22-54-96:CampaniaGP_codice vinces$ ./CentroVaccinale  
In attesa di nuove richieste di vaccinazione...  
█
```

Successivamente si esegue il ServerG:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice  
[10-22-54-96:CampaniaGP_codice vinces$ ./ServerG  
Attendendo GreenPass da scansionare...  
█
```

Poi si va ad eseguire il Client (Utente):

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice  
[10-22-54-96:CampaniaGP_codice vinces$ ./Utente localhost  
[Benvenuto al centro vaccinale di Frattamaggiore]  
  
Nome Utente: Vincenzo  
Cognome Utente: Salzano  
Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456789  
I suoi dati sono stati correttamente registrati!
```

Inoltre, si veda come questi dati ci vengono poi mostrati nel Centro vaccinale con annesse date di emissione e di scadenza:

```
Dati ricevuti
Nome: Vincenzo
Cognome: Salzano
Identificativo Tessera Sanitaria: 0123456789

Il GreenPass è stato emesso in data : 19:01:2023
Il GreenPass scadrà in data : 19:07:2023
```

Andiamo ad eseguire di nuovo il Client (Utente) per mostrare come ad ogni nuovo utente venga assegnato un Centro Vaccinale in modo differente. Ciò avviene grazie all'uso della funzione srand(time (NULL)) che serve ad inizializzare la funzione per la generazione dei numeri pseudocasuali. Grazie ad essa allo stesso seed vengono estratti numeri pseudocasuali diversi:

```
[10-22-54-96:CampaniaGP_codice vinces$ ./Utente localhost
[Benvenuto al centro vaccinale di Ischia]

Nome Utente: Vince
Cognome Utente: Sal
Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456788
I suoi dati sono stati correttamente registrati!
```

Come si potrà notare, ogni volta che un Client effettuerà una connessione, il ServerV mostrerà un numero pari a quest'ultime sul proprio terminale:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice
[10-22-54-96:CampaniaGP_codice vinces$ ./ServerV
In attesa di nuove connessioni...

In attesa di nuove connessioni...

In attesa di nuove connessioni...
```

Si va ad eseguire poi il ClientS, che ci dirà se il GreenPass sia valido o meno:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice
[10-22-54-96:CampaniaGP_codice vinces$ ./ClientS
[Le diamo il benvenuto nell'App CampaniaGP]

Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456789
L'identificativo della tessera sanitaria è corretto...

Convalida in corso, attendere...
Il GreenPass è valido!
```

Verificato che il GreenPass è valido, si va ad eseguire il ClientT con l'intenzione di voler sospendere il GreenPass:

```
[10-22-54-96:Desktop vinces$ cd CampaniaGP_codice
[10-22-54-96:CampaniaGP_codice vinces$ ./ClientT
[Campania GreenPass]

Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456789
Inserire: 0 [GreenPass Non Valido]
Inserire 1 [GreenPass Valido]
[INSERISCI 0 o 1]: 0
Inviando la richiesta di sospensione del GreenPass...
L'operazione ha avuto successo!
```

Una volta modificato il referto andiamo ad eseguire di nuovo il ClientS, per vedere se il referto è stato effettivamente cambiato:

```
[10-22-54-96:CampaniaGP_codice vinces$ ./ClientS
[Le diamo il benvenuto nell'App CampaniaGP]

Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456789
L'identificativo della tessera sanitaria è corretto...

Convalida in corso, attendere...
Il GreenPass non è valido!
```

Una volta sospeso il GreenPass, se lo si vuole ripristinare, si esegua nuovamente il ClientT avviando la procedura di ripristino:

```
[10-22-54-96:CampaniaGP_codice vinces$ ./ClientT
[Campania GreenPass]

Inserisca l'identificativo della tessera sanitaria(MAX 10 caratteri): 0123456789
Inserire: 0 [GreenPass Non Valido]
Inserire 1 [GreenPass Valido]
[INSERISCI 0 o 1]: 1

Inviando la richiesta di ripristino del GreenPass...
L'operazione ha avuto successo!
```

## CONCLUSIONE

Come illustrato nel paragrafo riguardante il “Manuale Utente” nella cartella “CampaniaGP\_codice” sono presenti tutti i file.c e l’header.h, opportunamente commentati, affinché chi legge possa incorrere nel minor numero di problemi possibili riscontrabili alla lettura.