UNIVERSITÀ
DEGLI STUDI
DI NAPOLI
PARTHENOPE

PROGETTO DI
RETI DI CALCOLATORI
E
LAB. DI RETI DI CALCOLATORI

Green Pass

Proponenti

Gianmarco Donnesi	0124002146
Sergio Aprea	0124002175

2021/2022

Indice

Capitolo 1

Descrizione generale del progetto

1.1 Traccia del progetto

La seguente relazione illustra un'implementazione della traccia **GreenPass**. Questa prevede la progettazione e l'implementazione di un servizio per la gestione dei Green pass secondo delle specifiche.

Un utente, una volta effettuata la vaccinazione, tramite un *Client* si collega ad un *Centro Vaccinale*, comunicando 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 verifcare 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.

Come base di dati è stato utilizzato un file .dat denominato **"db.dat"**. Questo file viene modificato e letto dal *ServerV*.

Il sistema è organizzato in questo modo:

- **ServerV (server principale)**: accede al file **"db.dat"** e esegue su questo le funzioni richieste dal *ClientS*, *ClientT* e *Client*, tramite i *ServerV* e *Centro Vaccinale*.
- **ServerG (server intermedio)**: rappresenta un "gateway" che permette lo scambio di informazioni tra i due Client (S e T) e il ServerV.
- **Client (utente)**: consente all'utente di inviare il numero della tessera sanitaria al Centro-Vaccinale, interagendo con il ServerV.
- **CentroVaccinale (server intermedio)**: rappresenta un "gateway" che permette lo scambio di informazioni tra il Client e il ServerV.
- **ClientT (utente)**: comunica con il ServerV, attravero il ServerG, potendo controllare, ripristinare o annullare la validità di un GreenPass.
- **ClientS (utente)**: comunica con il ServerV, attravero il ServerG, potendo controllare la validità di un GreenPass.

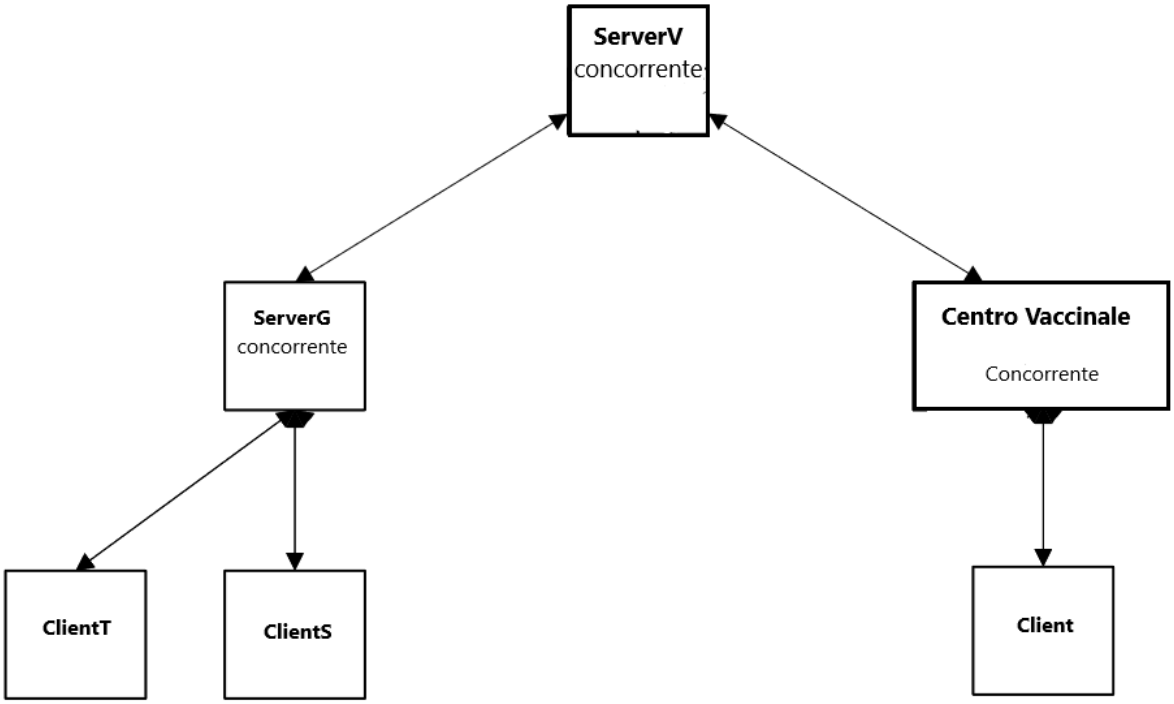
Capitolo 2

Descrizione e schemi dell'architettura di rete

La consegna prevede una struttura gerarchica delle entità. L'architettura di rete è la seguente:

- Il **ServerV**, il **ServerG** e il **CentroVaccinale** sono stati sviluppati sul modello di server concorrente e utilizzano, dunque, un descrittore in attesa di connessioni di client e un descrittore che si occupa di gestire la comunicazione con un client(ottenuto tramite la chiamata alla funzione fork()).

- Il **Client**, **ClientT** e il **ClientS** comunicano con i rispettivi Server e gestiscono il loro flusso di input/output con delle procedure bloccanti.

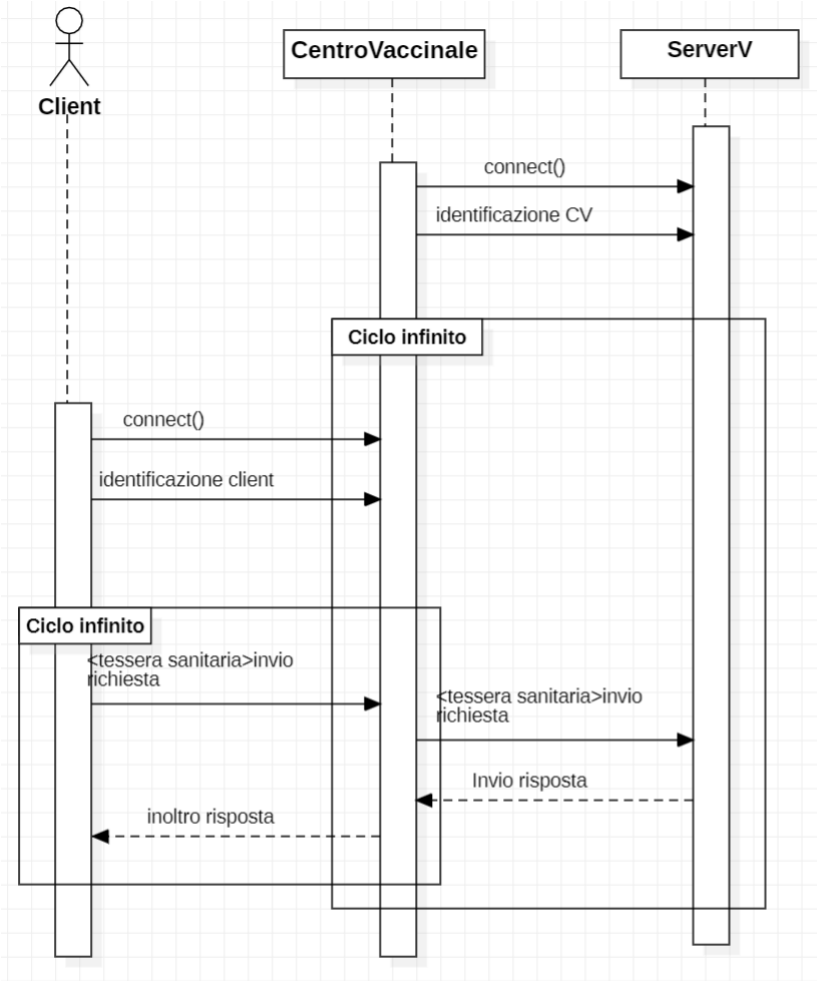


Capitolo 3

Descrizione e schemi del protocollo applicazione

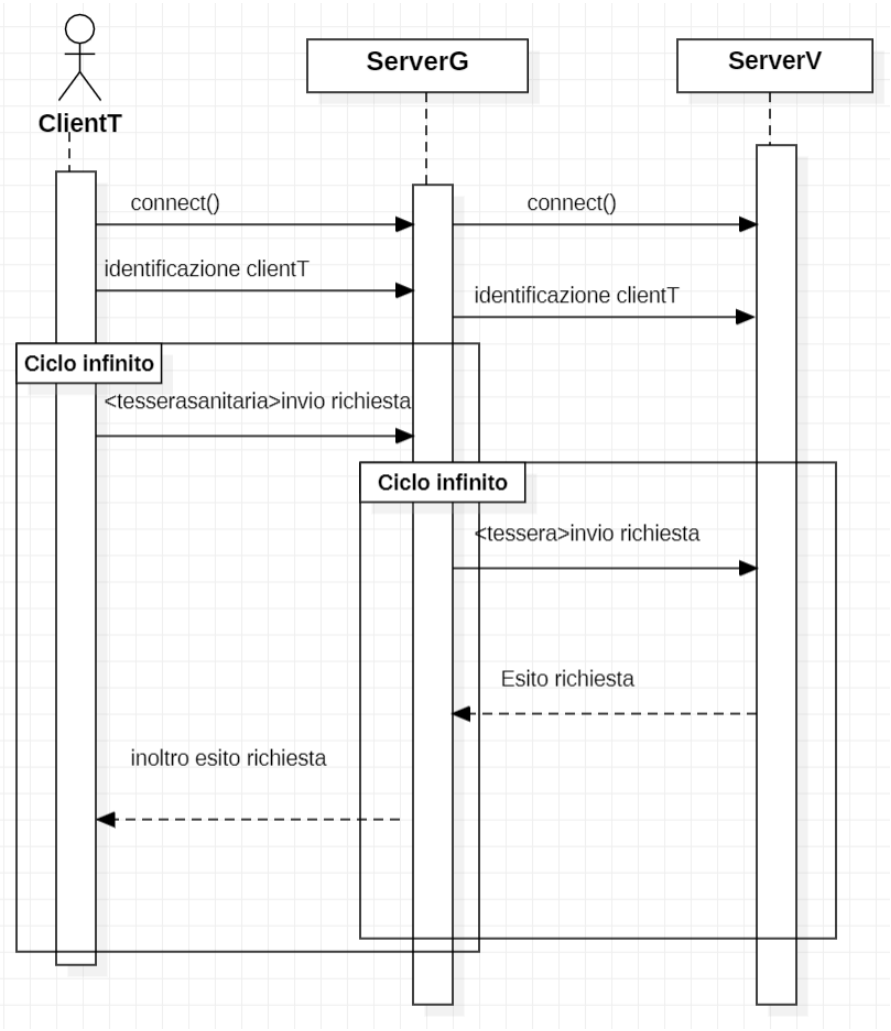
3.1 Protocollo per l'inserimento di un nuovo utente vaccinato

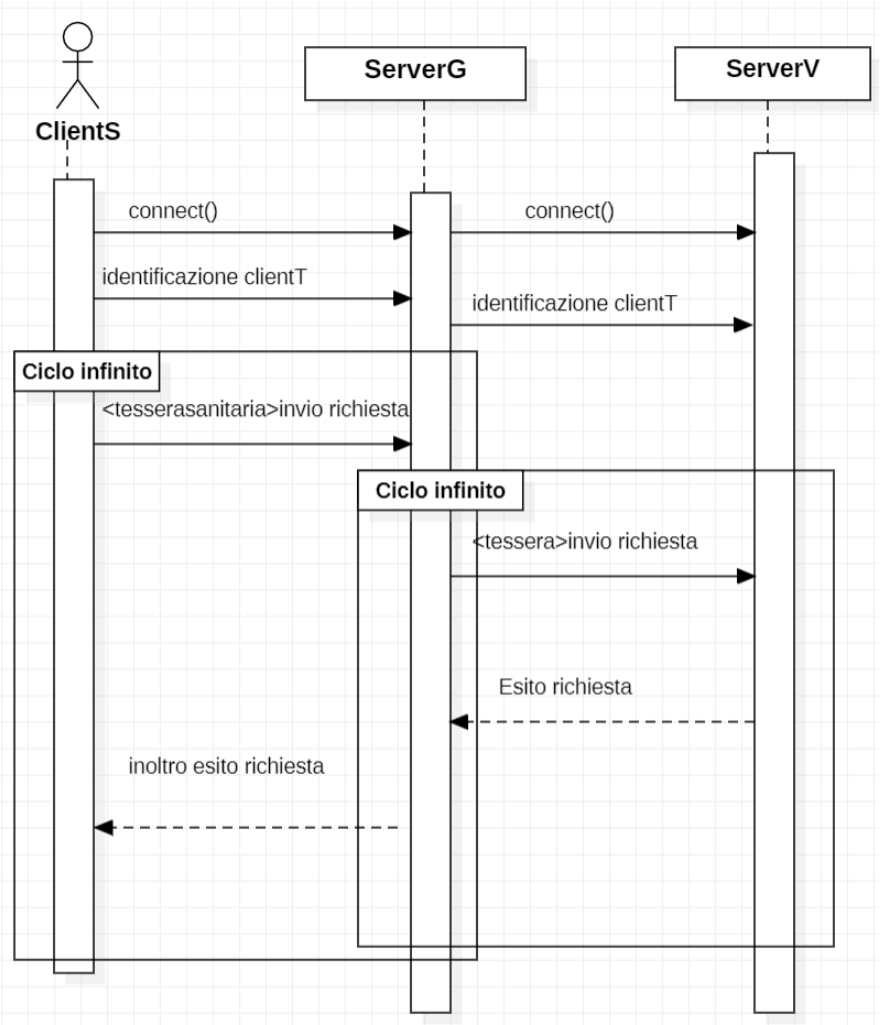
Il protocollo coinvolge il Client, il CentroVaccinale e il ServerV. Un utente per registrare la sua vaccinazione invia una richiesta, tramite il Client, al Centro Vaccinale contenente il codice della propria tessera sanitaria. Una volta ricevuto il codice della tessera e letta la tipologia della richiesta, il Centro Vaccinale invia il codice appena ricevuto al ServerV insieme al periodo di validità del GreenPass associato a quest'ultimo. Il ServerV, dopo aver ricevuto il pacchetto, inserisce le informazioni all'interno del file "db.dat" e invia un messaggio di risposta al centro vaccinale. Quest'ultimo invia una conferma di operazione avvenuta al client.



3.2 Protocollo per la verifica della validità di un GreenPass

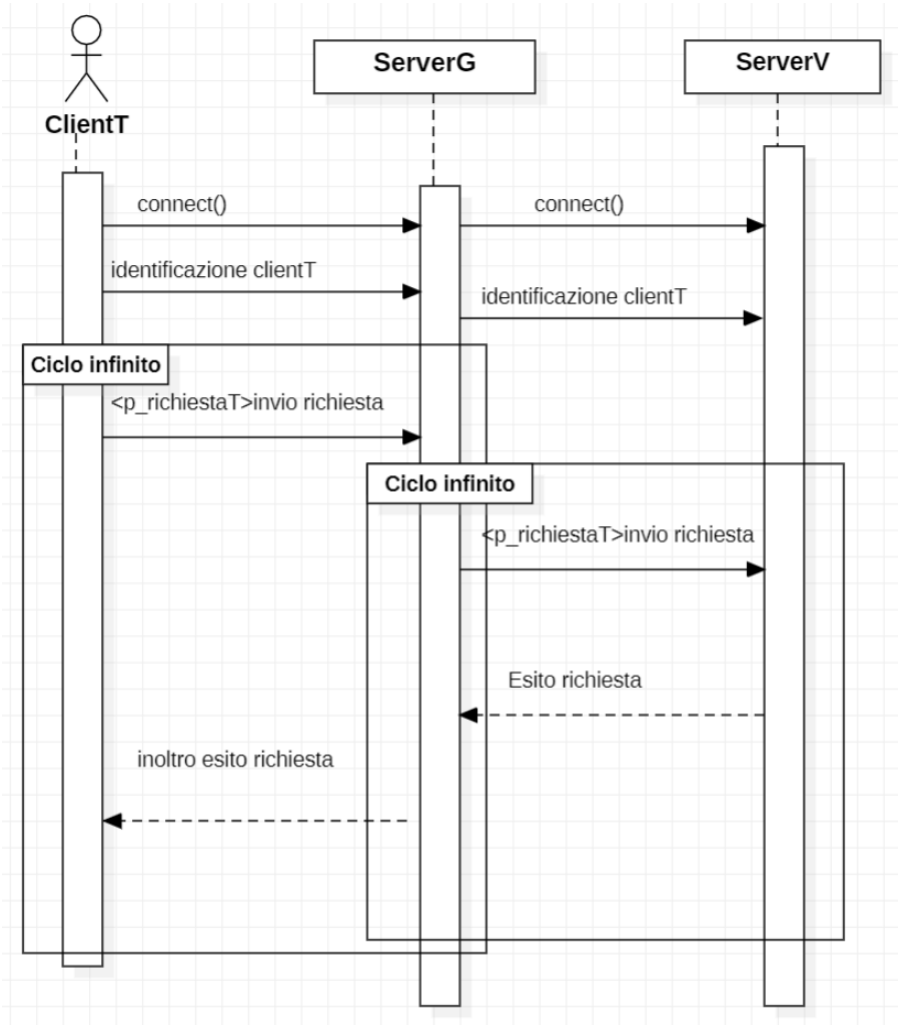
Il protocollo coinvolge il ClientS, ClientT, ServerG e ServerV. I Client inviano una richiesta di validità del GreenPass inviando al ServerG il codice della tessera sanitaria di un utente. Una volta ricevuto il codice e letta la tipologia della richiesta, il ServerG invia il codice appena ricevuto al ServerV che verifica la validità del GreenPass associato a quest'ultimo. Il ServerV invia, in seguito, l'esito della lettura della validità al ServerG, il quale la inoltra al client.


ClientT
ServerG
ServerV
connect()
connect()
identificazione clientT
identificazione clientT
Ciclo infinito
<tesserasanitaria>invio richiesta
Ciclo infinito
<tessera>invio richiesta
Esito richiesta
inoltro esito richiesta


ClientS
ServerG
ServerV
connect()
connect()
identificazione clientT
identificazione clientT
Ciclo infinito
<tesserasanitaria>invio richiesta
Ciclo infinito
<tessera>invio richiesta
Esito richiesta
inoltro esito richiesta

3.3 Protocollo per annullamento o ripristino della validità di un GreenPass

Il protocollo coinvolge ClientT, ServerG e ServerV. Il ClientT invia un pacchetto contenente il codice della tessera sanitaria e un carattere che indica l'effettivo contagio o guarigione di una persona vaccinata al ServerG. Quest'ultimo inoltra il pacchetto appena ricevuto al ServerV il quale, dopo aver visualizzato il carattere, annulla o ripristina la validità di un GreenPass. Eseguita l'operazione, invia l'esito di quest'ultima al ServerG, il quale lo invia al ClientT.



Capitolo 4

Dettagli implementativi dei client

Lo scambio di richieste e risposte tra ClientT, ServerG e ServerV avviene attraverso un pacchetto di richiesta definito *PRICHIESTAT*, rappresentato di seguito:

```
struct P_RICHIESTAT{
    char numero_tessera[20];
    char controllo_contagio[2];
};
```

Come affermato prima, i client sono stati sviluppati con delle chiamate bloccanti eseguiti in un blocco iterativo, in quanto le operazioni che gli utenti svolgono sono effettuate in maniera sequenziale.

Tutti i client, inoltre, sono caratterizzati da una funzione **StartMenu** che stampa, all'avvio, le operazioni che ognuno di essi può effettuare e una funzione **Logo** che stampa un messaggio di identificazione del Client specificato.

Capitolo 5

Dettagli implementativi dei server

5.1 Il ServerV

Il ServerV si occupa materialmente della gestione e la manipolazione dei dati salvati all'interno del file **"db.dat"**.

Le operazioni svolte dal Server sono:

1. Lettura della richiesta

2. Accesso al file per scrivere o leggere i dati

3. Inviare i risultati ottenuti al ServerG e al Centro Vaccinale

Le operazioni del server sono rese possibili grazie all'utilizzo di tre funzioni

```c
void rispostaCV(int sfd){

    time_t t = time( Time: NULL);
    struct tm tm = *localtime( Time: &t);
    struct UTENTE inserimento_utente;
    char CodiceTs[20];

    read( FileHandle: sfd,  DstBuf: &CodiceTs,  MaxCharCount: sizeof (CodiceTs));
    strcpy( Dest: inserimento_utente.numerotessera, Source: CodiceTs);
    inserimento_utente.validita = 1;
    inserimento_utente.giorno = tm.tm_mday;
    inserimento_utente.mese = tm.tm_mon +1;
    inserimento_utente.anno = tm.tm_year +1900;

    FILE *fp = fopen( Filename: "db.dat",  Mode: "ab");
    fwrite( Str: &inserimento_utente,  Size: sizeof(struct UTENTE),  Count: 1,  File: fp);
    fclose( File: fp);

    write( Filehandle: sfd, Buf: "1", MaxCharCount: 2);

}
```

```c
void ricercaValidita(int connfd, char numerotessera[20]){
    FILE *fp = fopen( Filename: "db.dat",  Mode: "rb");
    struct UTENTE Utente;

    while (fread( DstBuf: &Utente,  ElementSize: sizeof(struct UTENTE),  Count: 1,  File: fp) > 0)
    {
        if (strcmp(Utente.numerotessera,numerotessera) == 0)
        {
            if(Utente.validita == 1){
                write( Filehandle: connfd,  Buf: "1",  MaxCharCount: 2);
            }else if(Utente.validita == 0){
                write( Filehandle: connfd,  Buf: "2",  MaxCharCount: 2);
            }

        }
    }
    fclose( File: fp);
}
```

```c
void modificavalidita(int connfd, struct P_RICHIESTAT richiestaT){
    FILE *fp = fopen( Filename: "db.dat",  Mode: "rb+");
    struct UTENTE Utente;

    while (fread( DstBuf: &Utente,  ElementSize: sizeof(struct UTENTE),  Count: 1,  File: fp) > 0)
    {
        if (strcmp(Utente.numerotessera,richiestaT.numero_tessera) == 0)
        {
            if(Utente.validita == 0 && strcmp(richiestaT.controllo_contagio,"G") == 0){
                Utente.validita = 1;
                fseek( File: fp,  Offset: -sizeof(struct UTENTE),  Origin: SEEK_CUR);
                fwrite( Str: &Utente,  Size: sizeof(struct UTENTE),  Count: 1,  File: fp);
                break;

            } else if (Utente.validita == 1 && strcmp(richiestaT.controllo_contagio,"C") == 0){
                Utente.validita = 0;
                fseek( File: fp,  Offset: -sizeof(struct UTENTE),  Origin: SEEK_CUR);
                fwrite( Str: &Utente,  Size: sizeof(struct UTENTE),  Count: 1,  File: fp);
                break;
            }
        }
        write( Filehandle: connfd,  Buf: "1",  MaxCharCount: 2);
    }
    fclose( File: fp);
}
```

La concorrenza è garantita attraverso la creazione di un processo figlio che gestisce la connessione con i client.

5.2 Il ServerG e il CentroVaccinale

I ServerG e il CentroVaccinale fanno da tramite per la comunicazione tra i rispettivi client. Una volta connessi al server principale (ServerV), si mettono in attesa di una interazione da parte dei client. Il dialogo viene effettuato per mezzo di strutture definite come pacchetti, inviate come richieste al ServerV che, una volta effettuate le operazioni, inoltra l'esito di queste al ServervG e al Centro Vaccinale. Questi ultimi si occupano di restituire gli esiti delle operazioni ai rispettivi Client.

Capitolo 6

Manuale utente

La cartella principale del progetto contiene due sottocartelle: in *Header* sono presenti gli header dei codici delle funzioni utilizzate nei file .c, mentre nella cartella *libreria* è contenuta la struttura che indica le informazioni di ogni utente salvate all'interno del file acceduto dal ServerV.

6.1 Istruzioni per la compilazione

Per la compilazione è richiesto l'utilizzo di un sistema Linux, per l'utilizzo su dispositivi Windows bisogna installare una macchina virtuale e l'utilizzo in essa di un sistema operativo Linux(consigliamo per un buon funzionamento l'ultima versione di *"Ubuntu"*). Per compilare i codici sorgenti bisogna creare una cartella di compilazione, entrarci e utilizzare il comando gcc <nomedelfile.c> -o <nomedaassegnarealfile .o>

6.2 Istruzioni per l'esecuzione

Per eseguire i programmi basta recarsi nella cartella generata e lanciare tutte le unità in sequenza. A partire dal ServerV è possibile lanciarlo richiamando semplicemente il file *.o* e questo verrà aperto sulla porta 8025.

Successivamente bisogna lanciare i due server intermedi, ServerG e Centro Vaccinale, passando come secondo parametro l'indirizzo IP del ServerV (localhost: 127.0.0.1) entrambi aperti sulla stessa porta del ServerV per garantirne la comunicazione.

Infine è possibile lanciare Client(porta:8027), ClientS(porta:8026) e ClientT(porta:8026), passando come secondo parametro l'indirizzo IP del ServerV (localhost: 127.0.0.1).