Docenti:
Aniello Castiglione
Alessio Ferone

Studente:
Riccardo Andrea Spinosa

Matricola 0124002253

1.0 Introduzione

Traccia:

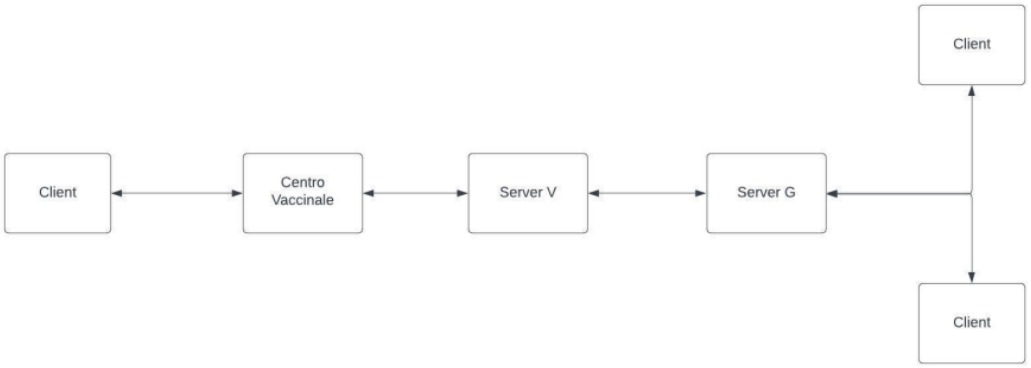
Progettare ed implementare un servizio di gestione dei green pass secondo le seguenti specifiche. Un utente, una volta effettuata la vaccinazione, tramite un client si collega ad un centro vaccinale e comunica il codice della propria tessera sanitaria. Il centro vaccinale comunica al ServerV il codice ricevuto dal client ed il periodo di validità del green pass. Un ClientS, per verificare se un green pass 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.

2.0 Descrizione progetto

Il progetto in questione prevede che un utente vaccinato si connetta al centro vaccinale attraverso un client e comunichi il proprio codice tessera sanitaria. Il centro vaccinale invia quindi al ServerV il codice ricevuto e il periodo di validità del green pass.
Inoltre, è presente un ClientS che viene utilizzato per verificare la validità di un green pass. Esso invia il codice della tessera sanitaria al ServerG, che richiede al ServerV il controllo della validità.
Infine, il ClientT è utilizzato per 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.

Per implementare questo progetto sono necessari diversi componenti, tra cui un client per l'utente, un client per il centro vaccinale, un Server G, un Server V e un client per la verifica della validità. Inoltre, potrebbe essere necessario un client per la modifica della validità, ad esempio per farmacie, ospedali o altri punti appositi.
In sintesi, il progetto prevede l'utilizzo di una serie di componenti e client/server per gestire la vaccinazione, la validità e la verifica del green pass.

3.0 Schema dell’architettura



Il "Client" rappresenta l'utente che interagisce con il client "Centro Vaccinale" dopo aver completato la vaccinazione. Il "Centro Vaccinale" è il client che comunica al "Server V" l'avvenuto vaccino da parte dell'utente. Il "Server V" conserva tutti i green pass degli utenti. Il "Server G" risponde alle richieste effettuate dai "Client T" e "Client S". Il "Client S" richiede al "Server G" la verifica della validità del green pass di un utente, e quest'ultimo comunica con il "Server V". Il "Client T" effettua modifiche sullo stato di validità di un green pass, comunicandolo al "Server G" che a sua volta comunica con il "Server V".

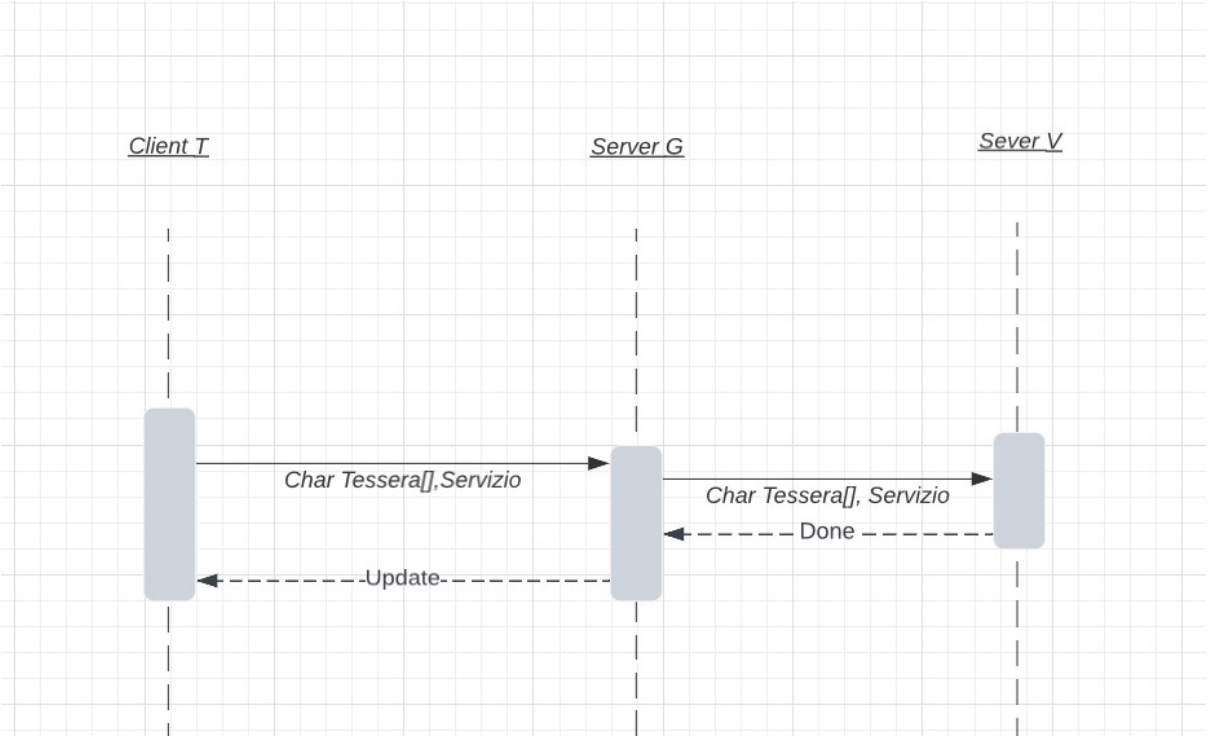
4.0 Descrizione e schemi del protocollo applicazione

4.1 Descrizione protocollo applicativo

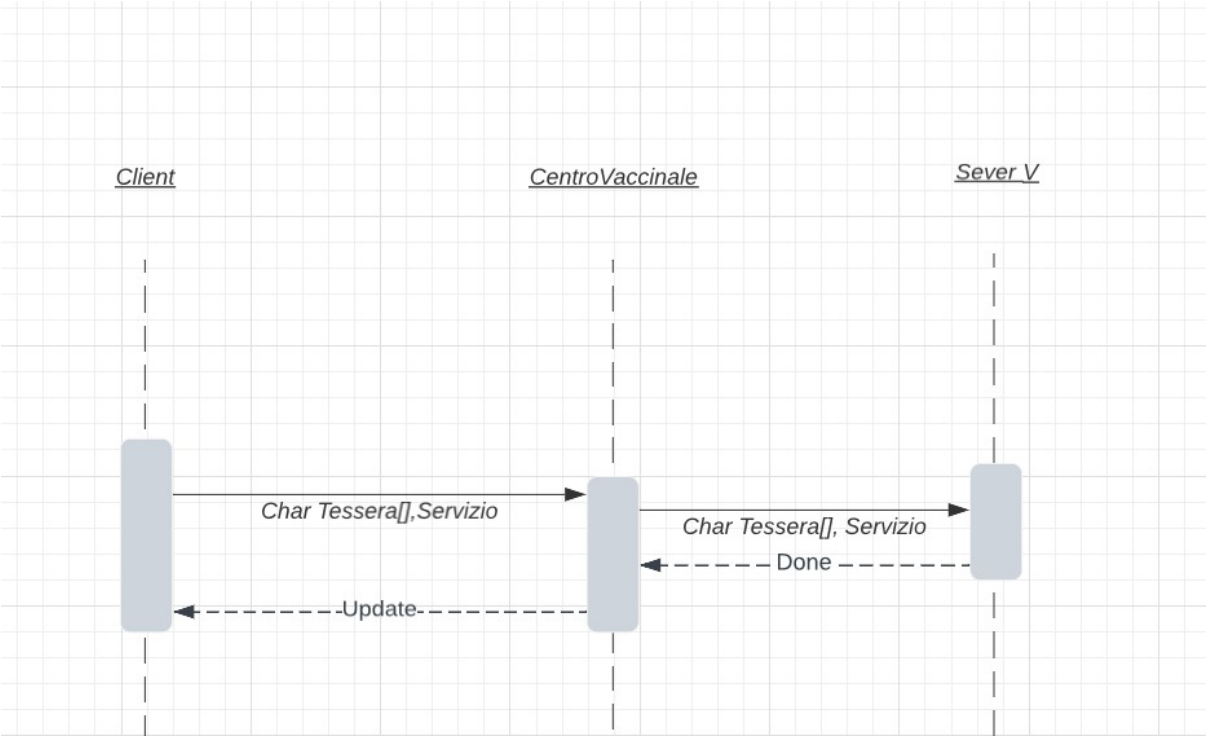
Il client del centro vaccinale riceve il codice e lo invia, insieme al periodo di validità del green pass, al server V per registrare i dettagli del green pass dell'utente.
Successivamente, il client S richiede la verifica della validità del green pass inviando il codice della tessera sanitaria al server G.
Il server G, a sua volta, richiede al server V la verifica della validità del green pass e riceve il risultato, il quale viene comunicato al client S.
Invece, il client T invia al server G il codice della tessera sanitaria insieme alla richiesta di invalidare o ripristinare la validità del green pass.
Il server G esegue le modifiche richieste sul server V e invia una risposta al client T.

4.2 Schema Protocollo Applicativo

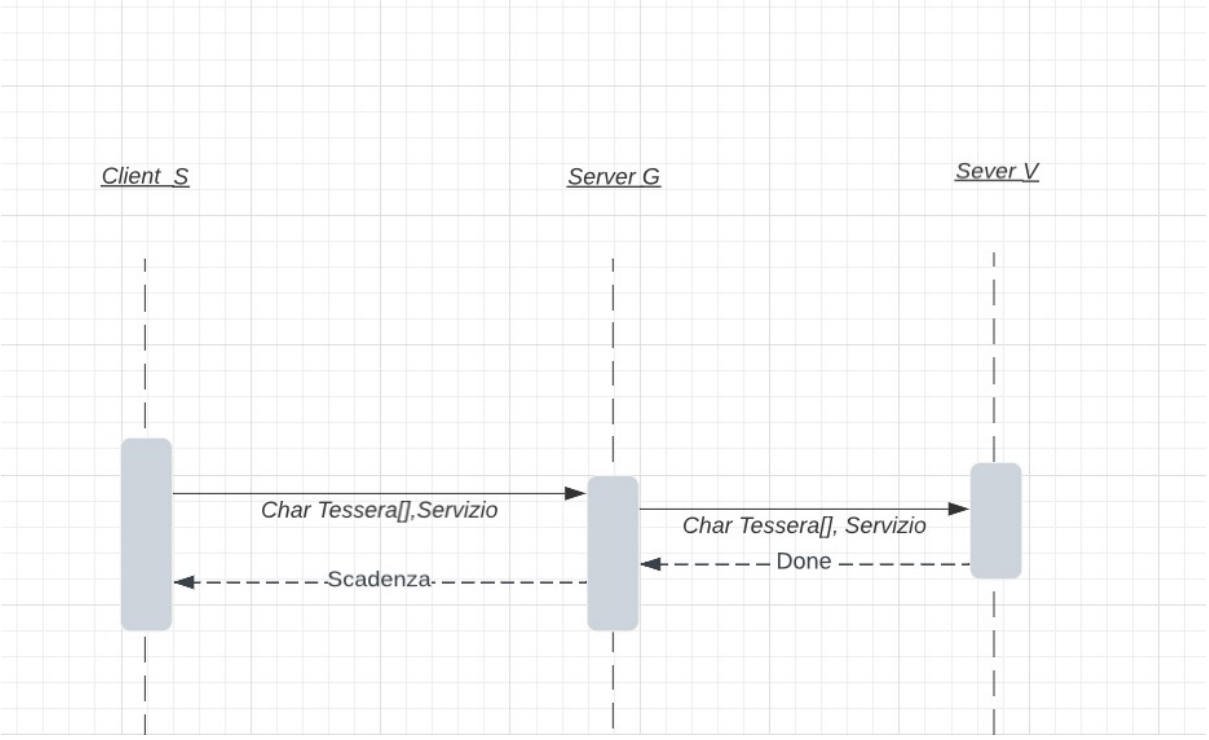
4.2.1 Cambio Stato



4.2.2 Vaccinazione



4.2.3 Richiesta Scadenza



5.0 Implementazione

Ecco parti rilevanti del codice

5.1 Dettagli implementativi lato Client

5.1.1 Client

In sintesi, si connette a un server specificato dall'utente e invia una tessera sanitaria. Prima di inviare la tessera sanitaria, il programma verifica che siano stati passati due argomenti e che la lunghezza della tessera sanitaria sia corretta. Se si verificano errori durante la connessione o l'invio della tessera sanitaria, il programma esce con un messaggio di errore. Infine, il programma chiude il socket.

```c
int main(int argc,char **argv) {
    // descrittore socket
    int sockFd;
    // server address
    struct sockaddr_in serverAddr;

    // Verifica argomenti
    if(argc != 3){
    fprintf(stderr,"Uso: %s <Indirizzo IP> <Tessera Sanitaria>\n",argv[0]);
    exit(1);
    }

    // lunghezza tessera sanitaria
    if(strlen(argv[2]) != 20){
    //errore
        fprintf(stderr,"Tessera non valida \n");
        exit(1);
    }

    // Creazione socket
    sockFd = Socket(AF_INET,SOCK_STREAM,0);

    // Impostazione parametri della connessione
    serverAddr.sin_family = AF_INET;
    serverAddr.sin_port = htons(1024);

    // Conversione da stringa a binario
    if(inet_pton(AF_INET,argv[1], &serverAddr.sin_addr) < 0) {
    fprintf(stderr,"Errore per %s\n",argv[1]);
    exit(1);
    }
    // Connessione al server
    Connect(sockFd,(struct sockaddr *) &serverAddr,sizeof(serverAddr));

    // Invio della tessera sanitaria
    if( write(sockFd,argv[2],strlen(argv[2])) != strlen(argv[2])) {
      perror("errore in scrittura");
      exit(1);
    }
    printf("Tessera Consegnata\n");

    // Chiude il socket
    close(sockFd);
}
```

5.1.2 Client S

il programma verifica che siano stati passati 3 argomenti sulla linea di comando, ovvero il nome dell'eseguibile, l'indirizzo IP del server e il numero della tessera sanitaria. Se gli argomenti non sono corretti, il programma stampa un messaggio di errore e termina. Successivamente, il programma verifica che la lunghezza della tessera sanitaria sia corretta (20 caratteri). Se la tessera non è valida, viene stampato un messaggio di errore e il programma termina.
Viene poi creata una variabile greenPass di tipo Pass e inizializzata con i valori della tessera sanitaria passati come argomento. Il campo servizio viene impostato a 0.
Il programma crea quindi un socket tramite la funzione Socket e prepara l'indirizzo del server in una variabile serverAddr di tipo sockaddr_in. Viene impostata la famiglia dell'indirizzo (AF_INET), la porta del server (1026) e l'indirizzo IP del server.

Successivamente, il programma effettua la connessione al server tramite la funzione Connect, passando come parametri il descrittore del socket, l'indirizzo del server e la dimensione dell'indirizzo.
Il programma invia quindi al server la richiesta di verifica della tessera sanitaria tramite la funzione write, passando come parametro il descrittore del socket e la variabile greenPass. Se la scrittura non va a buon fine, viene stampato un messaggio di errore e il programma termina.
Viene poi stampato a video un messaggio di attesa della risposta del server.
Il programma legge la risposta del server tramite la funzione read, passando come parametro il descrittore del socket e la variabile greenPass. Se la lettura non va a buon fine, viene stampato un messaggio di errore e il programma termina.
Infine, il programma verifica se la tessera sanitaria è valida confrontando la data di scadenza presente nella struttura greenPass con la data corrente ottenuta tramite la funzione time. Se la tessera è valida, viene stampato il messaggio "Greenpass: valido", altrimenti viene stampato il messaggio "Greenpass: non valido". Infine, il socket viene chiuso e il programma termina.

```c
struct Pass {
   int servizio;
   char tessera[21];
   time_t scadenza;
};

int main(int argc, char** argv){
    // Descrittore del socket
    int sockFd;
    // Indirizzo del server
    struct sockaddr_in serverAddr;
    // Struct per la tessera sanitaria
    struct Pass greenPass;

    // Verifico inserimento 3 argomenti (nome eseguibile, indirizzo IP e tessera sanitaria)
    if(argc != 3){
    fprintf(stderr, "Uso: %s <Indirizzo IP> <Tessera Sanitaria>\n",argv[0]);
    exit(1);
    }

    // Verifico che la lunghezza della tessera sanitaria sia di 20 caratteri
    if(strlen(argv[2]) != 20){
    fprintf(stderr, "Tessera invalida\n");
    exit(1);
    }
```

5.1.3 Client T

Il codice rappresenta un client che utilizza un socket per connettersi a un server e inviare un'istanza della struttura "Pass". La struttura è composta da una stringa di 20 caratteri che rappresenta il numero di tessera sanitaria, un valore intero che rappresenta il tipo di servizio e un valore di data di scadenza. Il client acquisisce i valori dei campi

della struttura da riga di comando e li utilizza per compilare un'istanza della struttura Pass. Successivamente, la struttura viene inviata al server tramite la funzione write(). Infine, la connessione viene chiusa.

```
// Connessione al server
Connect(sockFd,(struct sockaddr *)&serverAddr,sizeof(serverAddr));

// Scrittura del greenPass e invio richiesta al server
if( write(sockFd,&greenPass,sizeof(greenPass)) != sizeof(greenPass)) {
perror("Errore di scrittura");
exit(1);
}
// la richiesta è stata inviata con successo
printf("Richiesta inviata\n");
close(sockFd);
exit(0);
```

5.2 Dettagli implementativi Lato server

5.2.1 CentroVaccinale

implementa un server che accetta connessioni dai client, riceve un green pass dal client, lo invia al server V (che è identificato dall'indirizzo IP passato come argomento al programma) e termina la connessione.

Il green pass è un oggetto definito dalla struttura Pass, che contiene l'identificatore del servizio, il numero di tessera del green pass e la data di scadenza del green pass.

La scadenza del green pass è fissata a 3 mesi (in secondi).

Il programma utilizza le librerie standard del sistema operativo Linux per creare e gestire le connessioni di rete. In particolare, le librerie usate sono:

<stdio.h> per le funzioni standard di input/output.
<sys/types.h> per la definizione di tipi di dati utilizzati dalle chiamate di sistema.
<string.h> per le funzioni di manipolazione di stringhe.
<unistd.h> per la definizione di costanti e funzioni di sistema.
<stdlib.h> per la definizione di funzioni standard e di gestione della memoria.
<sys/socket.h> per la definizione di funzioni e strutture per la gestione dei socket di rete.
<arpa/inet.h> per la definizione di funzioni per la conversione tra indirizzi IP e numeri di porta.
<time.h> per la definizione di funzioni per la gestione del tempo.

Il programma usa le funzioni Socket(), Bind(), Listen(), Accept(), Connect() per creare e gestire le connessioni di rete.

La funzione Socket() viene usata per creare un socket. La funzione Bind() viene usata per associare il socket all'indirizzo e alla porta specificati. La funzione Listen() viene usata per mettere in ascolto il socket in attesa di connessioni in ingresso. La funzione Accept() viene usata per accettare una nuova connessione in ingresso.

Il programma usa inoltre la funzione fork() per creare un nuovo processo figlio per gestire la connessione del client. Il processo padre continua ad accettare nuove connessioni mentre il figlio gestisce la connessione del client.

Il programma legge dal socket il green pass inviato dal client, ne controlla la lunghezza e lo invia al server V specificato come argomento al programma. Il green pass ha una scadenza di 3 mesi.

Infine, il processo figlio termina la connessione e si chiude. Il processo padre continua ad ascoltare nuove connessioni in ingresso.

```
// sono figlio
if(pid == 0)
{
  close( lisenFd);
  // Chiude socket
  while((n = read(conFd,greenPass.tessera,20)) > 0 ) {
    // controllo che la lunghezza sia 20
    if (n < 20)
        greenPass.tessera[n] = 0;
    else
        greenPass.tessera[20] = 0;
    if (fputs(greenPass.tessera,stdout) == EOF) {
      fprintf(stderr,"error\n");
      exit(1);
    }
    greenPass.scadenza = time(NULL) + SCADENZA;
    greenPass.servizio = -1;

    if(n<0){
      fprintf(stderr,"errore in lettura\n");
      exit(1);
    }

    //invia dati a serverV
    serverVFd = Socket(AF_INET,SOCK_STREAM,0);
```

5.2.2 Server V

Il server è progettato per gestire le richieste di servizi riguardanti i green pass sanitari. Utilizza una socket per accettare le richieste in entrata e utilizza il fork per creare un processo figlio per gestire ogni singola richiesta. Quando una richiesta arriva, la struttura ricevuta viene letta dal socket tramite una chiamata a read. Il valore della proprietà servizio viene utilizzato per determinare l'azione da intraprendere.
Se la proprietà servizio in ricevuto è -1, il server verifica se esiste già un green pass associato alla tessera sanitaria specificata e, in caso contrario, lo inserisce nel file. Se la proprietà servizio vale 0, il server cerca nel file il green pass associato alla tessera sanitaria specificata in ricevuto e lo restituisce al client tramite una chiamata a write. Se la proprietà servizio è 1 o 2, il server cerca nel file il green pass associato alla tessera sanitaria specificata in ricevuto e ne estende la scadenza, scrivendo nuovamente nel file oppure la annulla.
Il codice utilizza semafori per garantire la sincronizzazione tra i processi durante l'accesso al file. Al termine del servizio richiesto, il processo figlio chiude la socket e termina.

```
  case 1:
  //la richiesta arriva dal serverG ma viene inviata dal ClientT, gestiamo quindi la convalida del greenPass
i = 0;
  printf("Convalida greenPass in corso\n");

        /*Si utilizza fseek per posizionare il puntatore del file all'inizio, quindi si utilizza un ciclo while per leggere ogni green pass nel file fino a quando non vien

  fseek(file,0,SEEK_SET);
  while(fread(&temp,sizeof(struct Pass),1,file) == 1){
    if(strcmp(temp.tessera,ricevuto.tessera) == 0){
        //caso in cui il greenPass viene registarto da un tampone
      if(temp.servizio == 3){
        temp.scadenza = time(NULL) + SCADENZA2GIORNI;
      }else{
          //caso in cui il GreenPass viene validato
      temp.scadenza = time(NULL) + SCADENZA;
    }
      printf("Green Pass validato\n");
      fseek(file,i*sizeof(struct Pass),SEEK_SET);
      sem_wait(access);
      fwrite(&temp,sizeof(struct Pass),1,file);
      sem_post(access);
      close(conFd);
      exit(0);
    }else{i++;}
  }
  sem_post(access);


//nel caso non sia nel file dei vaccinati la tessera sanitaria ottiene il greenPass valido 2 giorni(ha effettuato un tampone)
  strcpy(temp.tessera,ricevuto.tessera);
  temp.scadenza = time(NULL) + SCADENZA2GIORNI;
  temp.servizio = 3;
  printf("Tessera Sanitaria: %s\n",temp.tessera);
  printf("Scadenza: %.24s\r\n",ctime(&temp.scadenza));
  sem_wait(access);
  fseek(file,0,SEEK_END);
  fwrite(&temp,sizeof(struct Pass),1,file);
  sem_post(access);


  //chiudiamo la connessione
  close(conFd);
  exit(0);
    break;
```

5.2.3 Server G

Il server gestisce le richieste di Green Pass da parte di client (clientS e clientT) e inoltra tali richieste a un altro server (serverV).

In particolare, il serverG crea un socket per ascoltare nuove connessioni da clientS o clientT. Per ogni nuova connessione accettata, viene creato un processo figlio che gestisce la richiesta.

Il processo figlio riceve i dati della Green Pass e la richiesta di servizio da parte del client. Quindi, crea un nuovo socket per connettersi al serverV, converte l'indirizzo IP del serverV da stringa a formato di rete e si connette al serverV.

Il processo figlio quindi scrive i dati della Green Pass al serverV e attende una risposta. Se il servizio richiesto dal client è quello di clientS, il processo figlio legge la risposta dal serverV e la scrive al clientS. Se il servizio richiesto è quello di clientT, il processo figlio termina senza scrivere alcuna risposta.

Infine, il processo figlio chiude i socket e termina. Il processo padre continua ad accettare nuove connessioni e crea nuovi processi figli per gestirle.

```
        // Creazione del client per connettersi a serverV
  requestFd = Socket(AF_INET,SOCK_STREAM,0);

  requestAddr.sin_family = AF_INET;
        // Impostazione del numero di porta
  requestAddr.sin_port = htons(1025);

     // Converte un indirizzo di stringa in un indirizzo di rete, argv[1] contiene l'indirizzo IP
if(inet_pton(AF_INET,argv[1], &requestAddr.sin_addr) < 0) {
  fprintf(stderr,"Errore per %s\n",argv[1]);
  exit(1);
}

//connessione  a serverV
Connect(requestFd,(struct sockaddr *) &requestAddr,sizeof(requestAddr));
     // Scrittura su serverV
if( write(requestFd,&greenPass,sizeof(greenPass)) != sizeof(greenPass)) {
  perror("write");
  exit(1);
}
printf("Richiesta Consegnata\n");
printf("Servizio : %d\n",greenPass.servizio);

     //serivizio clientS
if(greenPass.servizio == 0){
  int r = read(requestFd,&greenPass,sizeof(greenPass));

    //se green pass non esiste
  if( (r != sizeof(greenPass))  && (r!= sizeof(int))   ){
    perror("read");
    exit(1);
  }else if(r < sizeof(greenPass)){
    printf("Green Pass non trovato\n");
  }else{

// greenPass a clientS
if( write(sockFd,&greenPass,sizeof(greenPass)) < sizeof(greenPass)){
  perror("Errore in scrittura");
  exit(1);
}
  }
```

6.0 Manuale

6.1 Istruzioni per la compilazione

- gcc -c -o Helper.o Helper.c
- gcc -o Client Client.c Helper.o
- gcc -o CentroVaccinale CentroVaccinale.c Helper.o
- gcc -pthread -o ServerV ServerV.c Helper.o
- gcc -o ServerG ServerG.c Helper.o ● gcc -o ClientS ClientS.c Helper.o
- gcc -o ClientT ClientT.c Helper.o

6.2 Istruzioni per l’esecuzione

1. ./ServerV
2. ./CentroVaccinale “indirizzo ServerV”
3. ./Client “indirizzo di CentroVaccinale” “tessera”

4. ./ServerG “indirizzo ServerV”
5. ./ClientS “indirizzo ServerG” “tessera sanitaria”
6. ./ClientT “indirizzo ServerG” “tessera sanitaria” “V o I”

7.0 Scenari

7.1 Primo Scenario

Nome Scenario	Un utente richiede di aggiornare il proprio GreenPass presso il serverV.
Attori Partecipanti	• Utente: chi avvia il clientS e richiede l'aggiornamento del GreenPass. • ClientS: l'applicazione client che l'utente avvia per richiedere l'aggiornamento del GreenPass. • ServerG: il server che gestisce le richieste di GreenPass e invia al clientS il GreenPass corrente dell'utente. • ServerV: il server che gestisce le richieste di aggiornamento del GreenPass e aggiorna il GreenPass corrente dell'utente nel file dei GreenPass.
Flusso Eventi	1. L'utente avvia il clientS. 2. L'utente seleziona l'opzione di aggiornamento GreenPass. 3. Il clientS richiede il GreenPass corrente dal serverG, fornendo la tessera sanitaria dell'utente. 4. Il serverG cerca nel file dei GreenPass e invia il GreenPass corrispondente al clientS. 5. Il clientS mostra il GreenPass corrente dell'utente e chiede all'utente di inserire i dati per aggiornare il GreenPass. 6. L'utente inserisce i dati richiesti e conferma l'aggiornamento del GreenPass. 7. Il clientS invia i nuovi dati al serverV, fornendo il servizio di aggiornamento GreenPass e la tessera sanitaria dell'utente. 8. Il serverV cerca nel file dei GreenPass la tessera sanitaria corrispondente e aggiorna il GreenPass con i nuovi dati. 9. Il serverV invia una conferma di aggiornamento al clientS. 10. Il clientS mostra un messaggio di conferma all'utente.

Nome Scenario	Un utente richiede di aggiornare il proprio GreenPass presso il serverV.

7.2 Secondo Scenario

Nome Scenario	Un utente richiede la verifica di un GreenPass presso il serverV
Attori Partecipanti	• Utente: l'utente che richiede la verifica del proprio GreenPass. • ClientS: il client che comunica con il serverG e il serverV per richiedere e inviare il GreenPass e la verifica del GreenPass. • ServerG: il server che gestisce i GreenPass e fornisce il GreenPass richiesto dal clientS. • ServerV: il server che gestisce la verifica dei GreenPass e fornisce una risposta all'esito della verifica richiesta dal clientS.
Flusso Eventi	1. L'utente avvia il clientS. 2. L'utente seleziona l'opzione di verifica GreenPass. 3. Il clientS richiede il GreenPass corrente dal serverG, fornendo la tessera sanitaria dell'utente. 4. Il serverG cerca nel file dei GreenPass e invia il GreenPass corrispondente al clientS. 5. Il clientS mostra il GreenPass corrente dell'utente e chiede all'utente di mostrare il proprio GreenPass. 6. L'utente mostra il proprio GreenPass al clientS. 7. Il clientS invia il GreenPass dell'utente al serverV, fornendo il servizio di verifica GreenPass e la tessera sanitaria dell'utente. 8. Il serverV cerca nel file dei GreenPass la tessera sanitaria corrispondente e verifica se il GreenPass è valido. 9. Il serverV invia una risposta al clientS con l'esito della verifica.

Nome Scenario	Un utente richiede la verifica di un GreenPass presso il serverV
	10. Il clientS mostra all'utente l'esito della verifica. Se il GreenPass è valido, l'utente può accedere al servizio richiesto. Se il GreenPass non è valido, l'utente riceve un messaggio di errore.

7.3 Terzo Scenario

Nome Scenario	Registrazione di un nuovo GreenPass presso il serverV da parte del CentroVaccinale
Attori Partecipanti	• Operatore del CentroVaccinale. • CentroV, • ClientV, • ServerV
Flusso Eventi	1. Un operatore del CentroVaccinale avvia il clientV 2. L'operatore seleziona l'opzione di registrazione GreenPass. 3. Il clientV richiede all'operatore i dati del nuovo GreenPass da registrare, ovvero la tessera sanitaria del paziente, il servizio richiesto e la data di scadenza del GreenPass. 4. L'operatore inserisce i dati richiesti e conferma la registrazione del GreenPass. 5. Il clientV invia i dati del nuovo GreenPass al serverV, utilizzando il servizio di registrazione GreenPass. 6. Il serverV riceve i dati del GreenPass inviati dal clientV. 7. Il serverV verifica la validità dei dati ricevuti e controlla se esiste già un GreenPass associato alla tessera sanitaria fornita. Se esiste già un GreenPass, il serverV restituisce un messaggio di

Nome Scenario	Registrazione di un nuovo GreenPass presso il serverV da parte del CentroVaccinale
	errore al clientV. Altrimenti, il serverV procede con la registrazione del nuovo GreenPass. 8. Il serverV aggiunge il nuovo GreenPass al file dei GreenPass registrati, associandolo alla tessera sanitaria fornita dal CentroVaccinale. 9. Il serverV invia al clientV una conferma di avvenuta registrazione del GreenPass. 10. L'operatore del CentroVaccinale riceve la conferma di registrazione del GreenPass dal clientV e comunica al paziente che il GreenPass è stato registrato con successo.

8.0 Sicurezza:

In fase di progettazione, sono stati considerati i seguenti problemi di sicurezza.

Autenticazione: è importante che il sistema di gestione dei green pass sia in grado di autenticare in modo sicuro i clienti, il centro vaccinale e il server. In particolare, il client dovrebbe autenticarsi con le proprie credenziali per impedire l'accesso non autorizzato al proprio green pass.

Crittografia: è importante utilizzare la crittografia per proteggere la comunicazione tra i vari componenti del sistema. Ad esempio, la comunicazione tra il client e il centro vaccinale dovrebbe essere protetta mediante crittografia a chiave pubblica.

Controllo degli accessi: è importante definire i diritti di accesso per ogni utente e garantire che ogni utente abbia accesso solo alle risorse a cui ha il permesso di accedere. In particolare, l'accesso ai dati personali e ai dettagli di green pass dovrebbe essere limitato solo ai clienti autorizzati.

Monitoraggio e logging: è importante monitorare continuamente il sistema per individuare eventuali attività sospette e registrare tutte le attività di accesso e modifica ai dati del green pass.

Gestione delle vulnerabilità: è importante gestire eventuali vulnerabilità del sistema in modo rapido ed efficace. Ciò può includere l'implementazione di patch di sicurezza e l'uso di tecnologie di rilevamento delle intrusioni.

Disaster recovery: è importante prevedere un piano di disaster recovery per garantire la disponibilità del servizio anche in caso di problemi hardware o software.

9.0 Future Implementazioni

Le implementazioni future:
Implementare le esigenze di sicurezza definite.

Implementazione di un database per il salvataggio delle informazioni sui green pass e dei relativi codici di servizio. In questo modo, i centri vaccinali possono controllare se una persona ha già ricevuto il vaccino e generare un nuovo green pass senza dover inserire nuovamente le informazioni.

Aggiunta di una funzionalità di notifica tramite email o SMS per ricordare agli utenti la scadenza del loro green pass.

Implementazione di un sistema di autenticazione per l'accesso alle informazioni sui green pass. In questo modo, solo le persone autorizzate possono accedere ai dati personali sensibili contenuti nei pass.

Integrazione con una piattaforma di prenotazione online per i vaccini, in modo da semplificare il processo di prenotazione e generazione del green pass.

Aggiunta di un sistema di tracciamento dei contatti basato sui green pass per notificare le persone che sono state a contatto con un individuo positivo al COVID-19.

Implementazione di un'interfaccia utente grafica per semplificare l'utilizzo dell'applicazione da parte degli utenti.

Aggiunta di un sistema di localizzazione GPS per mostrare ai utenti i centri vaccinali più vicini a loro e le relative disponibilità.