

RETI DI CALCOLATORI
E
LABORATORIO DI RETI DI CALCOLATORI
GREEN PASS

Michele Fiorentino
0124002085



Prof. Aniello CASTIGLIONE
Prof. Alessio FERONE

Indice

1	Descrizione del progetto	3
1.1	Introduzione	3
1.2	Specifiche	4
2	Descrizione e schemi dell'architettura	5
3	Descrizione e schemi del protocollo applicazione	6
3.1	Interazione Client-CentroVaccinale-ServerV	7
3.2	Interazione ClientS-ServerG-ServerV	8
3.3	Interazione ClientT-ServerG-ServerV	9
4	Dettagli implementativi	10
4.1	Dettagli implementativi dei client	11
4.1.1	Client	12
4.1.2	ClientS	12
4.1.3	ClientT	12
4.1.4	CentroVaccinale e ServerG	13
4.2	Dettagli implementativi dei server	14
4.2.1	CentroVaccinale e ServerG	14
4.2.2	ServerV	15
5	Manuale utente	18
5.1	Istruzioni per la compilazione	18
5.2	Istruzioni per l'esecuzione	19
6	Esempi	20

1 Descrizione del progetto

1.1 Introduzione

Come si legge dal sito FAQ Certificazione verde COVID-19:

“La Certificazione verde COVID-19 nasce per facilitare la libera circolazione in sicurezza dei cittadini nell’Unione europea durante la pandemia di COVID-19. Attesta di aver fatto la vaccinazione o di essere negativi al test o di essere guariti dal COVID-19. La Certificazione contiene un QR Code che permette di verificarne l'autenticità e la validità. La Commissione europea ha creato una piattaforma comune (Gateway europeo) per garantire che i certificati emessi dagli Stati europei possano essere verificati in tutti i Paesi dell’UE. In Italia la Certificazione viene emessa esclusivamente attraverso la Piattaforma nazionale del Ministero della Salute in formato sia digitale sia cartaceo.”

La Certificazione verde COVID-19 viene generata in automatico e messa a disposizione gratuitamente a chi:

- ha fatto la vaccinazione, a ogni dose di vaccino viene rilasciata una nuova certificazione;
- è risultato negativo a un test molecolare nelle ultime 72 ore o antigenico rapido nelle 48 ore precedenti;
- è guarito da COVID-19 da non più di sei mesi.

Data la natura didattica del progetto, sono state effettuate alcune semplificazioni: si considera il caso in cui venga rilasciato un GreenPass solo se il soggetto effettua una vaccinazione o guarisce dal Covid (e sempre per semplicità supponiamo che il tempo di validità del GreenPass sia sempre di 6 mesi).

Non si effettua una distinzione fra prima, seconda e dose booster (o eventuali che potrebbero risultare in futuro). Semplicemente ogni qual volta viene effettuata una nuova vaccinazione viene sostituito il GreenPass precedente con uno aggiornato, il quale sarà valido per **6 mesi**, dove per un mese si considera il “numero di giorni in media”, ovvero 30.42 giorni (il tempo verrà calcolato in secondi).

1.2 Specifiche

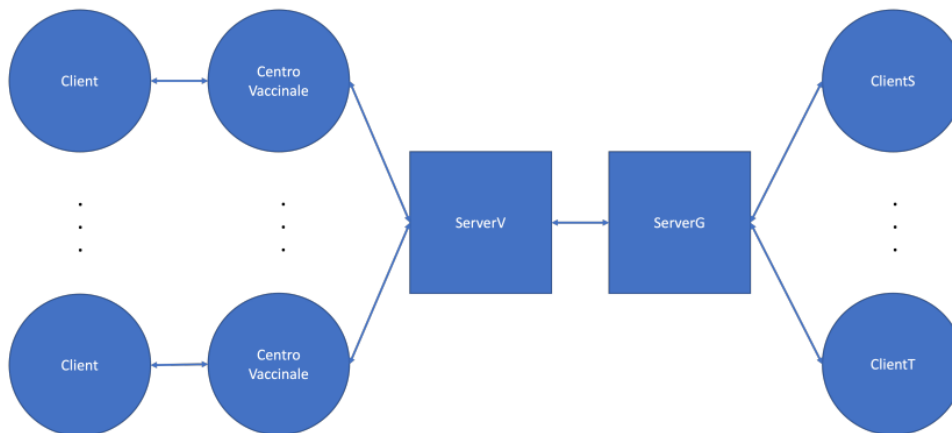
Il servizio della gestione dei green pass dovrà prevedere 3 specifiche:

- un **Client**, 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** 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.

Per tutta la trattazione si indicherà con "GreenPass" l'entità software realizzata. Quando si utilizzerà il termine "client" con la lettera iniziale in minuscolo ci si riferirà ad un generico processo client. Quando si utilizzerà il termine "Client" con la lettera iniziale in maiuscolo, ci si riferirà all'entità che fa parte del sistema GreenPass.

2 Descrizione e schemi dell'architettura

GreenPass si basa su un'architettura di rete di tipo client-server. Un'architettura client/server è un'architettura all'interno della quale vi sono una o più entità client che richiedono un servizio e una o più entità Server che offrono un servizio.



Di seguito riportiamo quali entità fungono da client, quali da server, quali invece da entrambi:

- **Client** invia la propria tessera sanitaria al **Centro Vaccinale**, il quale funge da intermediario fra di lui e il **ServerV**. Infatti il Centro Vaccinale comunica al **ServerV** la tessera sanitaria e il periodo di validità del GP di Client.
- **ClientS** e **ClientT** inviano la propria tessera sanitaria a **ServerG**, il quale anch'esso funge da intermediario fra di loro e il **Server V**. Il **ServerG** comunica al **ServerV** il tipo di richiesta e le informazioni opportune. Nello specifico: **ClientS** vuole verificare se il GP è valido, mentre **ClientT** è colui che può invalidare (in caso di contagio) o ripristinare (in caso di guarigione) il GP.
- Il cuore dell'architettura è rappresentato dal **ServerV** in quanto è colui che verifica le informazioni richieste dal Centro Vaccinale e dal **ServerG**.

Dunque è chiaro che i “client puri” siano Client, ClientS e ClientT, mentre colui che fornisce il servizio vero e proprio sia **ServerV**. CentroVaccinale e **ServerG** sono invece due entità che hanno lo scopo di mettere in comunicazione i client con **ServerV** (è come se fossero un livello intermedio), in particolare accettano le richieste dei client e chiedono la fornitura di un servizio a **ServerV**.

3 Descrizione e schemi del protocollo applicazione

GreenPass fa uso di due pacchetti di livello applicazione:

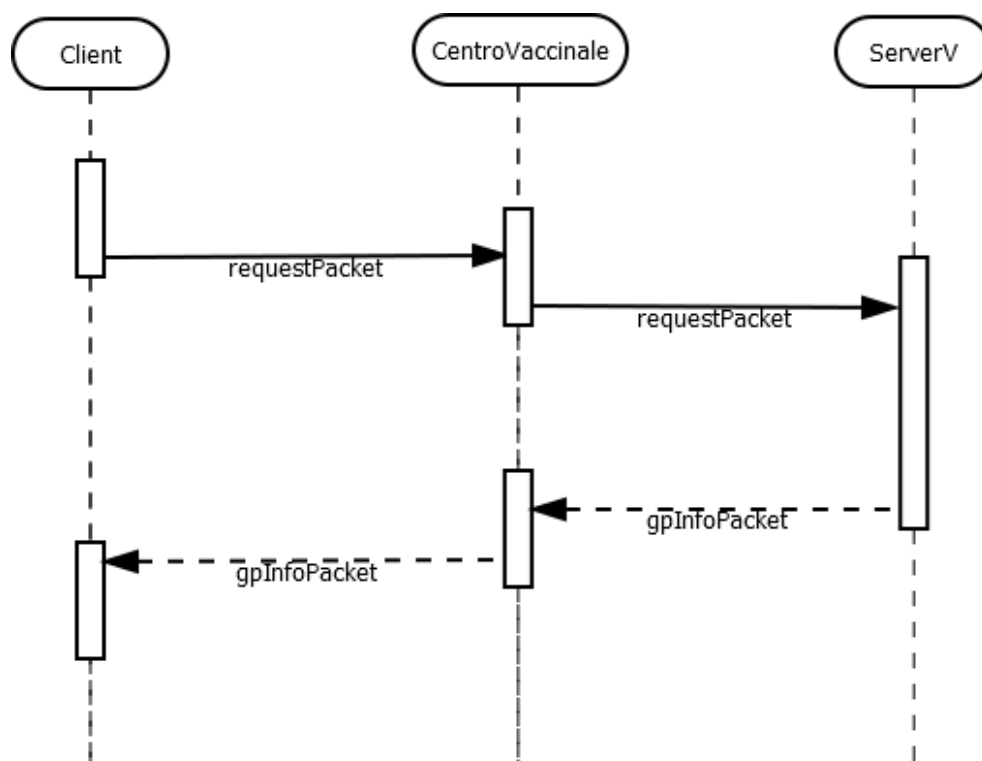
- **gpInfoPacket:**
 - **SSN:** numero associato alla tessera sanitaria (nonostante il nome, si fa riferimento solo al codice fiscale italiano);
 - **expDate:** data di scadenza del GreenPass;
 - **opResult:** risultato dell'operazione. In base a questo valore il Client sarà in grado di capire l'esito della richiesta effettuata.
- **requestPacket:**
 - **SSN:** numero associato alla tessera sanitaria;
 - **code:** codice della richiesta.

Possiamo distinguere diversi “codici di richiesta”, i quali saranno rappresentati da delle macro (es. REGISTER_GREEN_PASS per richiedere la registrazione del GreenPass). Il pacchetto di richiesta viaggia attraverso CentroVaccinale o ServerG fino a ServerV. Ognuno di loro tiene traccia di ogni richiesta accettata del client da cui la richiesta è arrivata, e useranno questa informazione per spedire un pacchetto “gpInfoPacket” al mittente, di cui il campo più importante è opResult in quanto contiene l'esito della richiesta (spesso il client non avrà bisogno di tutte le informazioni del pacchetto (tipo l'SSN), ma che possono essere comunque usate per “verifica”). Esempi di esiti sono: VACCINE_OK, GP_VALID, SSN_ERROR, ecc. . .

3.1 Interazione Client-CentroVaccinale-ServerV

Per effettuare una vaccinazione, è previsto che un Client si colleghi ad un CentroVaccinale. Una volta collegato, il Client invia un “requestPacket” al CentroVaccinale, il quale a sua volta inoltra il pacchetto a ServerV.

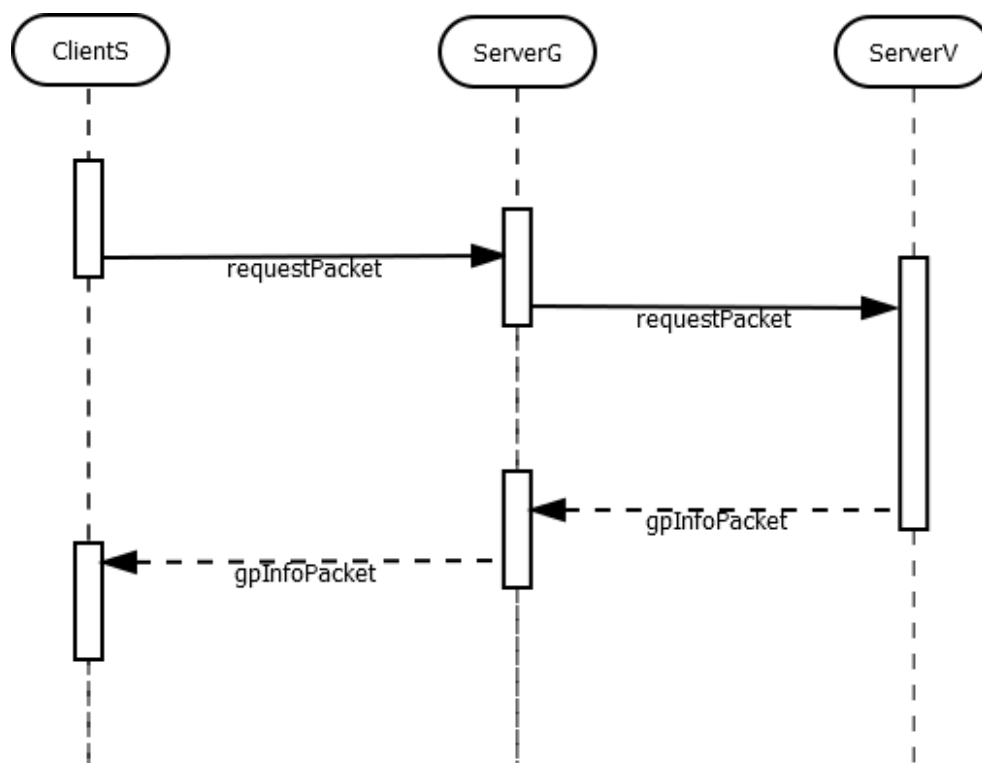
Il ServerV si occuperà di registrare la vaccinazione per il soggetto a cui corrisponde l’SSN inviato, e ritornerà l’esito attraverso un pacchetto “gpInfoPacket” al CentroVaccinale che aveva effettuato la richiesta, e che a sua volta si occuperà di inoltrare il pacchetto al Client.



3.2 Interazione ClientS-ServerG-ServerV

Per conoscere lo stato di validità di un Green Pass, è previsto che un ClientS si colleghi ad un ServerG. Una volta collegato, il ClientS invia un “requestPacket” al ServerG, il quale a sua volta inoltra il pacchetto a ServerV.

Il ServerV si verifica la validità del Green Pass per il soggetto a cui corrisponde l’SSN inviato, e ritornerà l’esito attraverso un pacchetto “gpInfoPacket” al ServerG che aveva effettuato la richiesta, e che a sua volta si occuperà di inoltrare il pacchetto al ClientS.

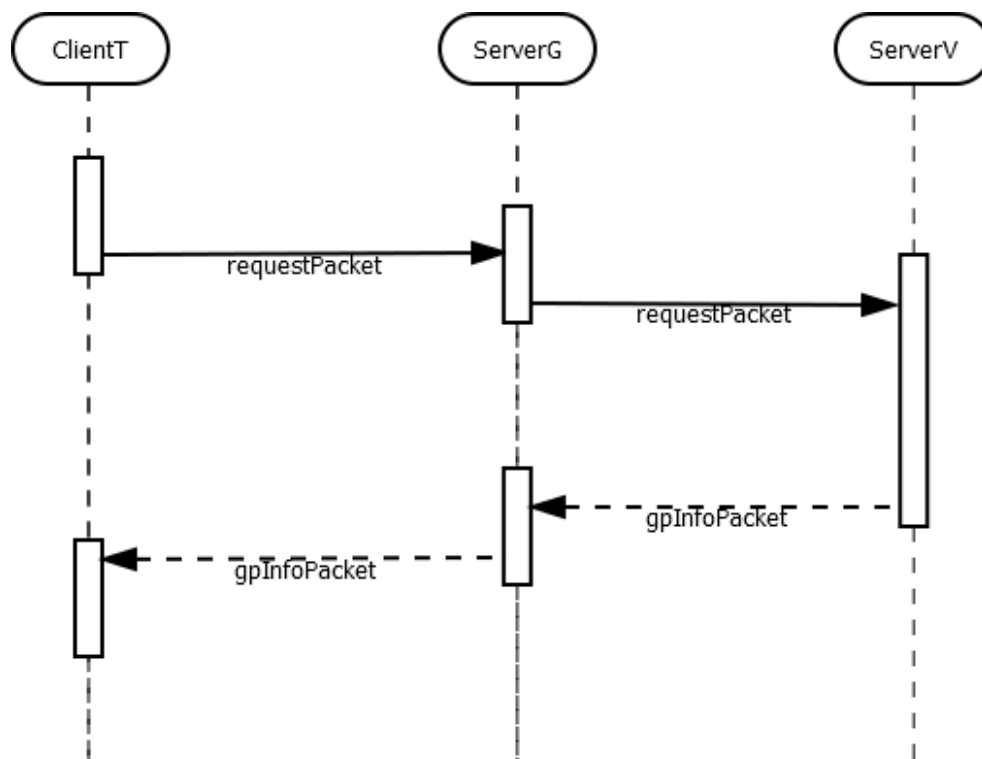


3.3 Interazione ClientT-ServerG-ServerV

Per poter abilitare/disabilitare è previsto che un ClientT si colleghi ad un ServerG. Una volta collegato, il ClientT invia un “requestPacket” al ServerG, il quale a sua volta inoltra il pacchetto a ServerV.

Il ServerV si occuperà di effettuare l’operazione di abilitazione/disabilitazione del Green Pass per il soggetto a cui corrisponde l’SSN inviato, e ritornerà l’esito attraverso un pacchetto “gpInfoPacket” al ServerG che aveva effettuato la richiesta, e che a sua volta si occuperà di inoltrare il pacchetto al ClientT.

Prima di inviare il pacchetto di richiesta, deve essere chiesto al ClientT qual è l’operazione da effettuare: **abilitare** o **disabilitare**.



4 Dettagli implementativi

Il progetto è costituito da diversi file, tra cui:

- **file sorgente:** Client, ClientS, ClientT, ServerG, ServerS, gpLib, gpUtilities, threadRequestHandlers;
- **file header:** gpLib(.h), gpUtilities(.h), e threadRequestHandlers(.h);
- **makefile:** lo useremo per compilare;
- **database:** conterrà le informazioni relative ai GP degli utenti.

I file riguardanti le entità dell'architettura del sistema verranno approfonditi nelle loro sezioni. Adesso ci concentreremo sui file gpLib e gpUtilities, che sono alla base di tutti gli altri file.

La libreria **gpLib** (GreenPass Library) definisce la struttura dei pacchetti, il percorso dei file, funzioni di comune utilità (es. controllo della validità di un SSN), e gli enumeratori per quanto riguarda i codici di richiesta e di risposta.

La libreria **gpUtilities** (GreenPass Utilities) definisce le wrapper per le funzioni di Berkeley, la FullWrite() e la FullRead(), e funzioni di comune utilità (in questo caso che hanno più a che fare con la parte di trasporto).

In entrambe i file header sono inclusi tutte le librerie standard di cui hanno bisogno gli altri file per funzionare correttamente, per questo motivo tali file avranno bisogno di importare solo questi due file.

Il file **threadRequestHandlers** contiene le funzioni per gestire le richieste dei client, e verrà utilizzato da ServerV.

nb: CentroVaccinale e ServerG sono sia client che server, dunque potrebbero essere inseriti sia nella sezione “client” che “server”. La parte client verrà trattata nella sezione dei client mentre la parte server verrà tratta nella sezione dei server.

4.1 Dettagli implementativi dei client

Client, ClientS e ClientT sono molto simili fra di loro.

Il codice dei client è costituito dalla sola funzione *main(...)*.

All'avvio del software dovrà essere inserito, tramite riga di comando, il numero di porta del server a cui ci si vuole connettere (il Centro Vaccinale per Client, o ServerG per ClientS e ClientT).

Verrà poi definito l'indirizzo: la porta la otteniamo attraverso la funzione *strtol* (string to unsigned short) in quanto è da 16 bit, mentre l'indirizzo IP corrisponde all'indirizzo di loopback 127.0.0.1.

La parte della definizione della struttura *sockaddr_in* è astratta dalla funzione *getInitAddr()*: utilizzeremo IPv4 e il protocollo di trasporto TCP.

Avviamo la connessione con il Centro Vaccinale/ServerG.

Popoleremo poi la struttura *requestPacket*, facendo inserire all'utente il codice fiscale, e in base al tipo di client inseriremo anche il tipo di richiesta.

Per Client e ClientS questo passaggio avverrà in automatica in quanto possono richiedere di effettuare una sola operazione (registrazione del GP per Client, controllo della validità del GP per ClientS) mentre per ClientT abbiamo due scelte: abilitazione o disabilitazione del Green Pass. La scelta verrà letta da una *scanf()*.

Verrà poi inviato il *requestPacket* al Centro Vaccinale/ServerG e il client si metterà in attesa dell'esito.

Ricevuto l'esito, lo analizza mostrando poi a schermo il risultato.

Infine chiude la connessione con il Centro Vaccinale/ServerG.

4.1.1 Client

Il **codice per richiedere la registrazione** del soggetto sarà rappresentato dalla macro: **REGISTER_GREEN_PASS**.

I possibili valori di ritorno sono:

- **VACCINE_OK**: la registrazione nel sistema è avvenuta con successo;
- **VACCINE_ERROR**: si è verificato un errore (es. manipolazione dei file);
- **SSN_ERROR**: Il codice Fiscale immesso non è valido.

4.1.2 ClientS

Il **codice per conoscere lo stato di validità di un Green Pass** corrispondente al numero di tessera sanitaria del soggetto sarà rappresentato dalla macro: **CHECK_GREEN_PASS_VALIDITY**.

I possibili valori di ritorno sono:

- **GP_VALID**: il Green Pass risulta valido (non è scaduto);
- **GP_NOT_VALID**: il Green Pass non risulta valido (è scaduto);
- **GP_NOT_FOUND**: il Green Pass associato a quel numero di tessera sanitaria non risulta essere presente nel sistema;
- **SSN_ERROR**: Il codice Fiscale immesso non è valido.

4.1.3 ClientT

Prima di inviare il pacchetto di richiesta, deve essere chiesto al ClientT qual è l'operazione da effettuare: **abilitare** o **disabilitare**.

Il **codice per abilitare un Green Pass** corrispondente al numero di tessera sanitaria del soggetto sarà rappresentato dalla macro: **ENABLE_GREEN_PASS**.

I possibili valori di ritorno sono:

- **GP_ENABLED**: guarigione. Il Green Pass è stato aggiornato (di fatto consiste nel rilasciare un nuovo Green Pass);
- **GP_NOT_DISABLED**: il Green Pass non è stato disabilitato/non è scaduto, quindi non ha senso abilitarlo;
- **ENABLING_ERROR**: il Green Pass era in corso di abilitazione ma si è verificato un errore durante il processo (es. nella manipolazione dei file);
- **SSN_ERROR**: Il codice Fiscale immesso non è valido.

Il codice per disabilitare un Green Pass corrispondente al numero di tessera sanitaria del soggetto sarà rappresentato dalla macro: **DISABLE_GREEN_PASS**.

I possibili valori di ritorno sono:

- **GP_DISABLED**: contagio. Il Green Pass è stato disabilitato (consiste nel porre la data di scadenza al 1° Gennaio 1970, è come se ci assicurassimo che risulti scaduto).
- **DISABLING_ERROR**: il Green Pass era in corso di disabilitazione ma si è verificato un errore durante il processo (es. nella manipolazione dei file);
- **GP_ALREADY_DISABLED**: si sta provando a disabilitare un Green Pass già disabilitato (la sua data di scadenza è posta al 1° Gennaio 1970);
- **GP_EXPIRED**: si sta provando a disabilitare un Green Pass scaduto (non ha senso farlo in quanto il Green Pass risulterebbe comunque non valido);
- **SSN_ERROR**: Il codice Fiscale immesso non è valido.

4.1.4 CentroVaccinale e ServerG

La parte client di CentroVaccinale e ServerG consiste nella connessione al ServerV. Poiché questa operazione deve essere effettuata sia dal ServerG che dal CentroVaccinale, è stata creata una funzione apposita chiamata `connectWithServerV()`.

In questo caso dovrà essere passato da riga di comando sia la porta per accettare le richieste dei client che la porta per permettere il collegamento al ServerV.

4.2 Dettagli implementativi dei server

CentroVaccinale e ServerG sono molto simili fra di loro, in quanto in entrambi i casi si tratta solo di intermediari, mentre deve essere fatto un discorso a parte per ServerV.

4.2.1 CentroVaccinale e ServerG

Il CentroVaccinale e il ServerG fungono da server rispettivamente per Client, ClientS e ClientT. Proprio perché devono poter accettare connessioni provenienti da un qualsiasi indirizzo IP viene specificato l'indirizzo IP 0.0.0.0. Sono entrambi implementati come server concorrenti.

I **server concorrenti** mi permettono di gestire più connessioni contemporaneamente. Ogni qualvolta un nuovo client vuole connettersi al server, viene utilizzata la system call `fork()` per generare un processo figlio che si occuperà di gestire la richiesta del client in questione. Il padre contemporaneamente si occuperà di accettare le nuove connessioni.

La `fork()` restituisce 0 al figlio, e restituisce il pid del figlio al padre.

Ricordiamo che un server utilizza due socket per gestire due situazioni diverse:

- `listenfd`: è il socket in attesa di connessioni;
- `connfd`: è il socket della connessione con il client.

Il processo figlio è una copia esatta del padre, e da questo eredita anche i descrittori, tuttavia il figlio non è interessato alle nuove connessioni, dunque chiuderà la socket `listenfd`. Al contrario il padre non è interessato a gestire la connessione con il client, dunque chiuderà la socket `connfd` e si preparerà ad accettare una nuova connessione.

Quando un processo figlio **termina**, viene inviato il segnale `SIGCHLD` al padre e il processo figlio diventa “zombie”, ovvero ha terminato l'esecuzione ma resta presente nella tabella dei processi.

Per **evitare che i figli rimangano nella condizione di zombie** una volta terminati, possiamo utilizzare la chiamata `signal(SIGCHLD, SIG_IGN)`.

Un CentroVaccinale può ricevere richieste solo da processi Client, mentre un ServerG può ricevere richieste solo da ClientS e ClientT.

4.2.2 ServerV

Il ServerV rappresenta il cuore di questo sistema. Viene implementato come un server **multithreaded**, ossia un server che genera un nuovo thread per ogni richiesta accettata.

Questa soluzione risulta molto efficiente in quanto mentre un thread continuerà ad accettare le richieste, gli altri thread si occuperanno del soddisfacimento di queste.

La **tipologia di thread** che verranno generati saranno thread detached, in quanto non siamo interessati a recuperare il loro stato una volta terminati (è più o meno la stessa cosa che facevamo ignorando il segnale di terminazione dei processi figli di CentroVaccinale/ServerG).

Questi thread, tuttavia, si potrebbero trovare ad accedere e modificare contemporaneamente il database (rappresentato da un file di testo), questo significa che il database è soggetto a **race condition**. Per evitare di creare situazioni di inconsistenza dei dati, utilizziamo un **semaforo binario** così da permettere ad un solo thread per volta l'accesso in **mutua esclusione** del file.

Anche in questo caso, deve essere inserita la porta su cui il ServerV si metterà in ascolto da riga di comando. La definizione della struttura sockaddr_in e le fasi di avvio della connessione sono le stesse di quelle viste per CentroVaccinale e ServerG.

All'interno di un ciclo infinito, il thread principale accetterà le richieste di connessione e per ogni richiesta accettata creerà un nuovo thread che se ne occuperà. Per tener traccia del client, verrà allocato spazio per una copia di connfd e questo verrà passato al thread come attributo.

Il thread generato richiama la funzione “handleRequest()”, nella quale riceve il pacchetto da parte del CentroVaccinale/ServerG e in base al codice del requestPacket richiama la funzione opportuna.

Prima di richiamare la funzione viene effettuato un controllo sulla validità del codice fiscale (almeno nella forma), e in caso non risultasse valido (perché il numero di caratteri è insufficiente) viene restituito come risultato SSN_ERROR.

Se l'SSN è valido, viene finalmente richiamata la funzione opportuna.

Effettuata l'operazione, prepara un pacchetto di risposta di tipo gpInfoPacket() e lo invia al socket passato come argomento (connfd, ovvero un client di tipo CentroVaccinale o ServerG).

Le **funzioni per gestire le richieste** sono definite all'interno del file **threadRequestHandlers**.

Possiamo distinguere **quattro funzioni** che possono essere richiamate direttamente dal "handleRequest".

- **request_RegisterGreenPass**: copia nel pacchetto gpInfo l'SSN e la nuova data di scadenza del GreenPass. Poi richiama la funzione registerGreenPassInDatabase() per registrare il GreenPass all'interno del database. La funzione restituisce un esito con il quale si verifica se la registrazione della vaccinazione è andata a buon fine. Infine viene restituito il pacchetto gpInfo.
- **request_CheckGreenPassValidity**: copia nel pacchetto gpInfo l'SSN. Poi individua la data di scadenza all'interno del database attraverso la funzione findGreenPassExpDate(). Possiamo distinguere 3 situazioni: il Green Pass è valido, il Green Pass non è valido, il Green Pass non è stato trovato. Infine viene restituito il pacchetto gpInfo.
- **request_EnableGreenPass**: abilitare un Green Pass significa in pratica rilasciarne uno nuovo, dunque si tratta di richiamare la funzione per registrare il Green Pass. Questo però potremo farlo solo il GreenPass risultava già disabilitato o scaduto dunque dovremo anche controllare il campo expDate. La data di scadenza all'interno del database verrà individuata attraverso la funzione findGreenPassExpDate(). La registrazione nel database avviene attraverso la funzione registerGreenPassInDatabase(), la quale restituisce un esito con il quale si verifica se è andata a buon fine. Infine viene restituito il pacchetto gpInfo.
- **request_DisableGreenPass**: disabilitare un Green Pass significa in pratica porre la sua data di scadenza a 0, che significa impostare la data al 1° Gennaio 1970. Questo significa che il GP risulterà sempre "scaduto". Inoltre verificheremo se il GP è stato già disabilitato (non ha senso disabilitare un GP già disabilitato) o se è scaduto (anche qui, non ha senso disabilitare un GP scaduto) La data di scadenza all'interno del database verrà individuata attraverso la funzione findGreenPassExpDate(). La registrazione nel database avviene attraverso la funzione registerGreenPassInDatabase(), la quale restituisce un esito con il quale si verifica se è andata a buon fine. Infine viene restituito il pacchetto gpInfo.

Ricordiamo che il database deve essere acceduto in mutua esclusione per evitare situazioni di race condition, per questo motivo le funzioni *registerGreenPassInDatabase()* e *findGreenPassExpDate()* possono essere richiamate dai thread solo una volta che il semaforo glielo permette.

Abbiamo già citato 2 delle 3 **funzioni di utilità**. In totale queste sono:

- **registerGreenPassInDatabase**: per poter inserire una nuova riga all'interno del file potremmo banalmente aprirlo in modalità “append” e scrivere i dati, tuttavia quest'operazione si complica se dobbiamo “sostituire” una riga (rimpiazzare un vecchio GreenPass con uno più nuovo), per questo motivo dobbiamo creare una copia del file e copiare in questo tutte le righe tranne quella che abbiamo intenzione di sostituire. Poi inseriremo nel file copia anche la nuova riga da aggiungere. Infine eliminiamo il file originale e rinominiamo il file copia così che possa diventare il nuovo file database.
Infine ritorna l'esito.
- **findGreenPassExpDate**: tale funzione si occupa di restituire la data di scadenza associata al Green Pass nel database del soggetto specificato attraverso il numero di tessera sanitaria.
Anche in questo caso iteriamo su tutte le righe del file finché non troviamo una corrispondenza. Se la troviamo, utilizziamo la funzione *getExpDateFromDBLine()* per estrarre solo la parte relativa alla data di scadenza della riga.
Infine ritorna la data di scadenza (-1 se il GP non è stato trovato).
- **getExpDateFromDBLine**: tale funzione consente di estrarre la data di scadenza di una riga della forma (SSN+expDate) e di convertirla da stringa nel giusto formato numerico. Infine ritorna la data di scadenza.

5 Manuale utente

5.1 Istruzioni per la compilazione

Per la compilazione del progetto viene messo a disposizione un **makefile**. Grazie all'utility make, all'utente basterà recarsi, utilizzando la shell, nella directory "GreenPass" ed eseguire il comando make. Tale comando provvederà alla compilazione dei sorgenti in modo del tutto automatico.

Nella remota ed eventuale possibilità che il comando make non dovesse funzionare, si riportano di seguito i comandi da eseguire per la compilazione del progetto. I comandi devono essere eseguiti nell'ordine in cui sono presentati:

```
gcc -c gpLib.c gpUtilities.c threadRequestHandlers.c
```

```
gcc -c Client.c
```

```
gcc Client.o gpLib.o gpUtilities.o -o Client.out
```

```
gcc -c ClientS.c
```

```
gcc ClientS.o gpLib.o gpUtilities.o -o ClientS.out
```

```
gcc -c ClientT.c
```

```
gcc ClientT.o gpLib.o gpUtilities.o -o ClientT.out
```

```
gcc -c CentroVaccinale
```

```
gcc CentroVaccinale.o gpLib.o gpUtilities.o -o CentroVaccinale.out
```

```
gcc -c ServerG.c
```

```
gcc ServerG.o gpLib.o gpUtilities.o -o ServerG.out
```

```
gcc -c ServerV.c
```

```
gcc ServerV.o gpLib.o gpUtilities.o threadRequestHandlers.o -pthread -o  
ServerV.out
```

5.2 Istruzioni per l'esecuzione

Di seguito viene spiegato come eseguire ciascun entità software, quali argomenti passare a riga di comando e cosa più importante, in che ordine effettuare l'esecuzione delle entità. Un'entità software appartenente al progetto viene eseguita con il comando `./nome_entità.out`, dove a *nome_entità* bisogna sostituire il nome dell'entità software che si desidera eseguire. Ad esempio se vogliamo eseguire il ServerV ci basterà digitare da terminale `./ServerV.out` seguito da eventuali parametri.

Ogni entità riceve in input diversi parametri:

- **Client.out** riceve come parametro il numero di porta del CentroVaccinale. **ClientT.out** e **ClientS.out** ricevono in input il numero di porta del ServerG.
Si noti che non è necessario passare anche l'indirizzo IP in input poiché sia CentroVaccinale che ServerG hanno come indirizzo IP l'indirizzo IP di loopback (127.0.0.1);
- **CentroVaccinale.out** e **ServerG.out** ricevono come parametro di input il numero di porta su cui devono rimanere in ascolto per le richieste dei client e il numero di porta del ServerV. Anche in questo caso non è necessario passare l'indirizzo IP del ServerV in quanto il suo indirizzo è l'indirizzo di loopback.
- **ServerV.out**: riceve come parametro solo il numero di porta su cui deve rimanere in ascolto.

Al fine di assicurare il corretto funzionamento del sistema software, le entità software del sistema dovrebbero essere eseguite nel seguente ordine:

1. `./Server.out <Numero di porta>`
2. `./ServerG.out <Numero di porta><Numero di porta ServerV>`
3. `./CentroVaccinale.out <Numero di porta><Numero di porta ServerV>`
4. Eseguire uno dei client in base alla richiesta che bisogna effettuare, ricordando che:
 - se si vuole eseguire il Client bisogna scrivere:
`./Client.out <Numero di porta CentroVaccinale>`
 - se si vuole eseguire ClientT o ClientS bisogna scrivere:
`./ClientX.out <Numero di porta ServerG>`
dove X può assumere il valore di T oppure S.

È importante rispettare l'ordine e i parametri di ciascuna entità per garantire il corretto funzionamento del sistema.

6 Esempi

Schermata con tutte le entità in esecuzione

<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Doc michele@DESKTOP-MIKI:/mnt/c/Users/miche/Doc \$./Client.out 50000 Inserire Tesser</pre>	<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Docur michele@DESKTOP-MIKI:/mnt/c/Users/miche/Docur /ServerV.out 60000 ServerV attivo</pre>
<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Doc michele@DESKTOP-MIKI:/mnt/c/Users/miche/Doc \$./ClientS.out 55000 Inserire Tesser</pre>	<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Docur michele@DESKTOP-MIKI:/mnt/c/Users/miche/Docur /ServerG.out 55000 60000 ServerG attivo</pre>
<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Doc michele@DESKTOP-MIKI:/mnt/c/Users/miche/Doc \$./ClientT.out 55000 Inserire Tesser</pre>	<pre>michele@DESKTOP-MIKI: /mnt/c/Users/miche/Docur michele@DESKTOP-MIKI:/mnt/c/Users/miche/Docur /CentroVaccinale.out 50000 60000 Centro Vaccinale attivo</pre>

Registrazione effettuata con successo (VACCINE_OK)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti
$ ./Client.out 50000
Inserire Tesser
```

Vaccinazione effettuata con successo per FYHTBB40L24G846M.
Data di scadenza del GreenPass: Tue Jan 17 07:11:46 2023

Controllo del Green Pass, risultante valido (GP_VALID)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolato
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolato
$ ./ClientS.out 50000
Inserire Tesser
```

Controllo effettuato con successo per: FYHTBB40L24G846M.
Il Green Pass risulta valido, e scade il: Tue Jan 17 07:11:46 2023

Controllo del Green Pass, non risultante valido (GP_NOT_VALID)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/R
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/A
$ ./ClientS.out 50000
Inserire Tessera Sanitaria: GLHBHW75S57F627W

Controllo effettuato con successo per: GLHBHW75S57F627W.
Il Green Pass non risulta valido
```

Disabilitazione di un Green Pass (GP_DISABLED)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/
$ ./ClientT.out 50000
Inserire Tessera Sanitaria: FDPCCF86L07B984M
Inserire Operazione (E = Enable / D = Disable): d

Controllo effettuato con successo per: FDPCCF86L07B984M.
Il Green Pass e' stato disabilitato
```

Abilitazione di un Green Pass (GP_ENABLED)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolatori/Gr
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolatori/Gr
$ ./ClientT.out 50000
Inserire Tessera Sanitaria: FDPCCF86L07B984M
Inserire Operazione (E = Enable / D = Disable): E

Controllo effettuato con successo per: FDPCCF86L07B984M.
Il Green Pass e' stato abilitato, e scade il: Tue Jan 17 07:23:07 2023
```

Abilitazione di un Green Pass ancora valido (GP_NOT_DISABLED)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolatori/Gr
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di Calcolatori/Gr
$ ./ClientT.out 50000
Inserire Tessera Sanitaria: FDPCCF86L07B984M
Inserire Operazione (E = Enable / D = Disable): e

Il Green Pass risulta ancora valido, non e' possibile "abilitarlo"
```

Disabilitazione di un Green Pass già disattivato (GP_ALREADY_DISABLED)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni/Reti di C
michele@DESKTOP-MIKI:/mnt/c/Users/miche/Documents/GitHub/AppuntiUni$ ./ClientT.out 50000
Inserire Tessera Sanitaria: LNRTZJ69A69B742Z
Inserire Operazione (E = Enable / D = Disable): D
Si sta provando a disabilitare un Green Pass già disabilitato.
```

Disabilitazione di un Green Pass scaduto (GP_EXPIRED))

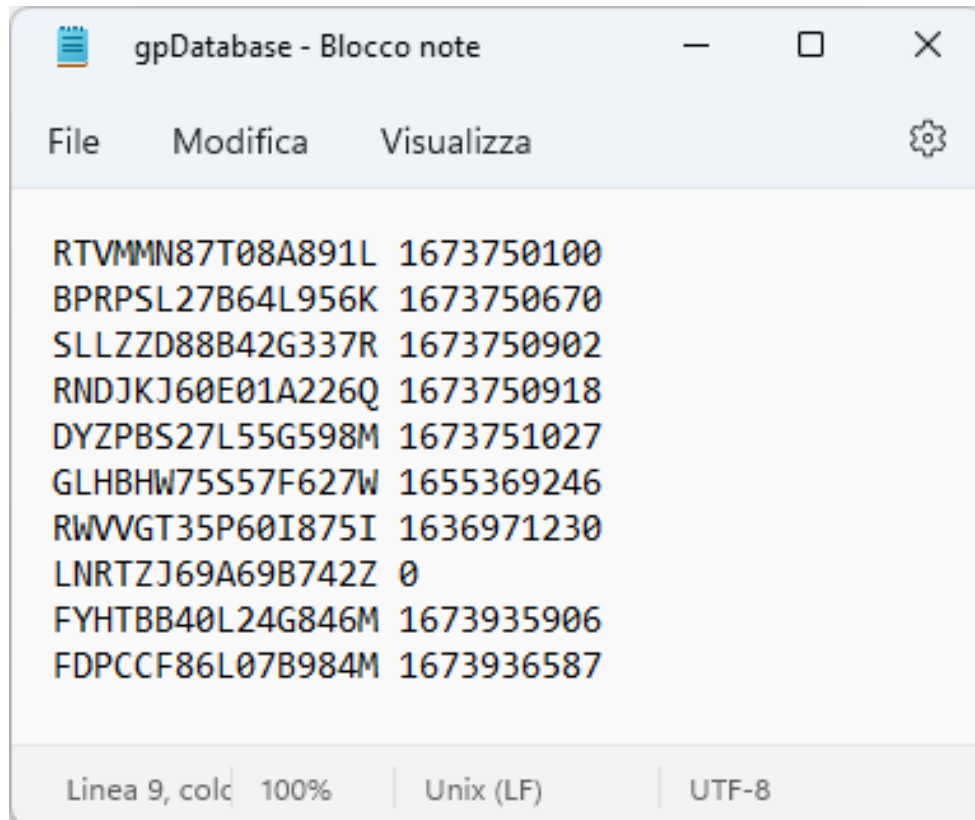
```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni$ ./ClientT.out 50000
Inserire Tessera Sanitaria: GLHBHW75S57F627W
Inserire Operazione (E = Enable / D = Disable): D
Si sta provando a disabilitare un Green Pass scaduto.
```

Inserimento di un codice fiscale non valido (SSN_ERROR)

```
michele@DESKTOP-MIKI: /mnt/c/Users/miche/Documents/GitHub/AppuntiUni$ ./Client.out 50000
Inserire Tessera Sanitaria: FRNMHL

Il numero di caratteri inserito e' insufficiente.
Il codice fiscale italiano ha 16 caratteri.
```

Dati nel database



```
RTVMMN87T08A891L 1673750100
BPRPSL27B64L956K 1673750670
SLLZZD88B42G337R 1673750902
RNDJKJ60E01A226Q 1673750918
DYZPBS27L55G598M 1673751027
GLHBHW75S57F627W 1655369246
RWVVG T35P60I875I 1636971230
LNRTZJ69A69B742Z 0
FYHTBB40L24G846M 1673935906
FDPCCF86L07B984M 1673936587
```

Linea 9, colc | 100% | Unix (LF) | UTF-8