

UNIVERSITÀ DEGLI STUDI DI NAPOLI “PARTHENOPE”  
SCUOLA INTERDIPARTIMENTALE DELLE SCIENZE  
DELL’INGEGNERIA E DELLA SALUTE

CORSO DI LAUREA IN INFORMATICA



CORSO DI RETI DI CALCOLATORI  
E  
LABORATORIO DI RETI DI CALCOLATORI

**GREEN PASS**

DOCENTI

Castiglione Aniello  
Ferone Alessio

CANDIDATO

Iannucci Vincenzo  
matricola: 0124002093  
Fiorentino Michele  
matricola: 0124002085

Anno Accademico 2021-2022

# Indice

<b>1</b>	<b>Green Pass</b>	<b>1</b>
1.1	Descrizione del progetto . . . . .	1
1.2	Descrizione e schemi dell'architettura . . . . .	3
1.3	Descrizione e schemi del protocollo applicazione . . . . .	4
1.3.1	Interazione Client-CentroVaccinale-ServerV . . . . .	5
1.3.2	Interazione ClientT-ServerG-ServerV . . . . .	6
1.3.3	Interazione ClientS-ServerG-ServerV . . . . .	7
1.4	Dettagli implementativi dei client . . . . .	8
1.4.1	Dettagli implementativi del Client . . . . .	8
1.4.2	Dettagli implementativi del ClientS . . . . .	9
1.4.3	Dettagli implementativi del ClientT . . . . .	10
1.4.4	Dettagli implementativi di ServerG e CentroVaccinale . . . . .	11
1.5	Dettagli implementativi dei server . . . . .	11
1.5.1	Dettagli implementativi di ServerV . . . . .	11
1.5.2	Dettagli implementativi di ServerG e CentroVaccinale . . . . .	12
1.6	Manuale utente . . . . .	13
1.6.1	Istruzioni per la compilazione . . . . .	13
1.6.2	Istruzioni per l'esecuzione . . . . .	14

# Elenco delle figure

1.1	Schema dell'architettura . . . . .	4
1.2	Diagramma delle sequenze dell'interazione Client-CentroVaccinale- ServerV . . . . .	6
1.3	Diagramma delle sequenze dell'interazione ClientT-ServerG-ServerV	7
1.4	Diagramma delle sequenze dell'interazione ClientS-ServerG-ServerV	8
1.5	Avvio del processo ServerV . . . . .	16
1.6	Avvio del processo ServerG . . . . .	16
1.7	Avvio del processo CentroVaccinale . . . . .	17
1.8	Esecuzione del processo Client . . . . .	17
1.9	Esecuzione del processo ClientS . . . . .	18
1.10	Esecuzione del processo ClientT . . . . .	18

# Capitolo 1

## Green Pass

### 1.1 Descrizione del progetto

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 apportate alcune semplificazioni. La prima riguarda le modalità di assegnazione di un green pass ad un cittadino.

Non si effettua una distinzione tra prima, seconda e dose booster. Ogni qual volta viene effettuata una nuova vaccinazione si aggiorna la data di validità del green pass relativa al numero della tessera sanitaria del cittadino. La seconda riguarda il calcolo della data di validità del green pass. Un cittadino può effettuare un'altra dose di vaccino solo dopo che sono trascorsi **cinque mesi** dall'ultima inoculazione. I cinque mesi sono calcolati approssimando la data al primo del mese corrente. Ad esempio se la data odierna è 09/05/2022, la data in cui è possibile effettuare la prossima vaccinazione sarà 01/10/2022. Questa semplificazione permette di calcolare velocemente la data della prossima vaccinazione senza tener conto di anni bisestili, mesi con trenta giorni e mesi con trentuno giorni.

Di seguito si riportano le specifiche software del sistema, con le relative scelte effettuate:

- Un utente, una volta effettuata la vaccinazione, tramite un client si collega ad un centro vaccinale e comunica il codice della propria tessera sanitaria. Si suppone che il codice della tessera sanitaria contenga caratteri alfanumerici e sia lungo al più venti caratteri. Non si effettuano controlli circa la validità di tale codice ma si controlla solo che il codice inserito dall'utente rispetti la lunghezza prefissata.
- Il centro vaccinale comunica al ServerV il codice ricevuto dal client ed il periodo di validità del green pass. Il periodo di validità è nel formato DD-MM-YYYY (ad esempio 15-05-2022). Si è scelto tale formato in quanto è di più facile interpretazione per i cittadini (in Italia) poiché standard di riferimento per le date.
- 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. Valgono le stesse considerazioni fatte prima sul numero della tessera sanitaria.
- 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. Molto banalmente il client inserirà il valore 0 quando vorrà comunicare la sua positività al COVID-19, disattivando così il green pass. Inserirà 1 quando vorrà ripristinare la validità del suo green pass in seguito a guarigione.

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.

## 1.2 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 e quali da server, quali invece da entrambi:

- **Client**
  - Client: richiede la nuova data di validità del green pass dopo l'inoculazione del vaccino. Dialoga con il ServerV per mezzo del Centro Vaccinale;
  - ClientS: richiede la verifica dello stato di validità di un green pass. Dialoga con il ServerV per mezzo del ServerG;
  - ClientT: richiede di poter comunicare l'infezione o la guarigione da COVID-19. Dialoga con il ServerV per mezzo del ServerG.
- **Server:** il ServerV offre una serie di servizi ai client. Infatti offre un controllo circa lo stato di validità del green pass e permette di disattivare o attivare un green pass.

CentroVaccinale e ServerG fungono da client per il ServerV e da server rispettivamente per Client e ClientS e ClientT. Lo scopo di queste due entità è quello di mettere in comunicazione i vari client con il server. È come se l'architettura client-server avesse un livello intermedio, rappresentato da queste due entità.

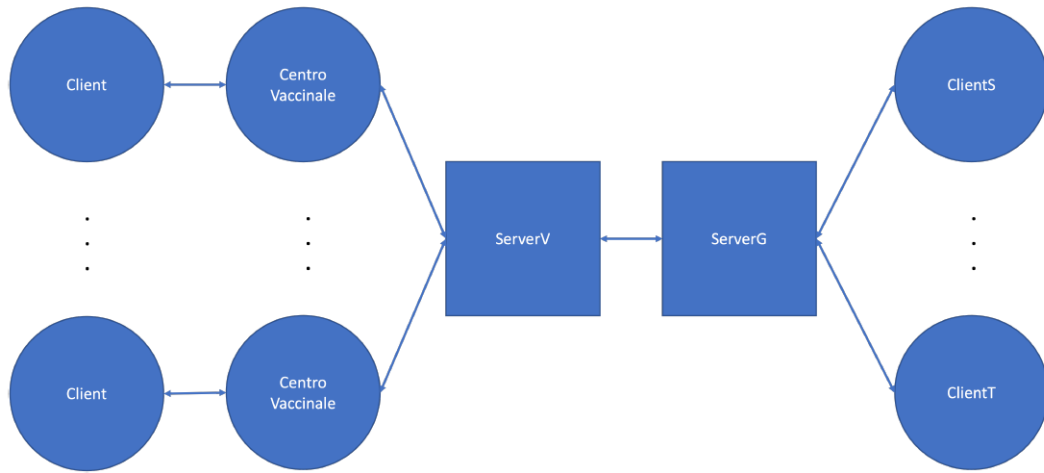


Figura 1.1: Schema dell'architettura

### 1.3 Descrizione e schemi del protocollo applicazione

GreePass fa uso principalmente di due tipi di pacchetto di livello applicazione:

- **packet**: contiene
  - **healthInsureCardNumber**: il numero associato alla tessera sanitaria;
  - **greenPassValidityDate**: la data di validità del green pass;
  - **returnedValue**: l'esito della richiesta effettuata;
- **packet2**: contiene
  - **healthInsureCardNumber**: il numero associato alla tessera sanitaria
  - **updateValue**: il valore con il quale si desidera aggiornare lo stato di validità del green pass (0 o 1)



Per far capire al ServerV da quale client proviene la richiesta, viene usato un `operationCode` che potrà assumere i seguenti valori:

- 0: la richiesta proviene dal Client via CentroVaccinale;
- 1: la richiesta proviene dal ClientT via ServerG;
- 2: la richiesta proviene dal ClientS via ServerG.

### 1.3.1 Interazione Client-CentroVaccinale-ServerV

Per effettuare una vaccinazione, il protocollo di livello applicazione di un Client prevede di collegarsi con un CentroVaccinale. Una volta collegato, il Client invia il codice associato alla tessera sanitaria al CentroVaccinale. Il CentroVaccinale (che è già connesso al ServerV), dopo un'interazione "pronto-risposta" con il ServerV utilizzando l'`operationCode`, invia il pacchetto di tipo `packet` al ServerV contenente il numero della tessera sanitaria. Il ServerV provvederà a verificare se è possibile effettuare la vaccinazione e setterà i campi del pacchetto di ritorno (di tipo `packet`) relativi alla data in cui è possibile effettuare la prossima vaccinazione e al codice di ritorno dell'operazione. I codici di ritorno possibili sono due:

- 0: il vaccino non può essere effettuato poiché non sono passati ancora 5 mesi dall'ultima dose
- 1: il vaccino è stato inoculato con successo. Viene letta la data in cui è possibile effettuare la prossima vaccinazione

Successivamente il ServerV inoltra il pacchetto al CentroVaccinale, il quale provvede direttamente ad inoltrarlo al Client. A seconda dell'esito ottenuto, il Client stamperà un opportuno messaggio a video.

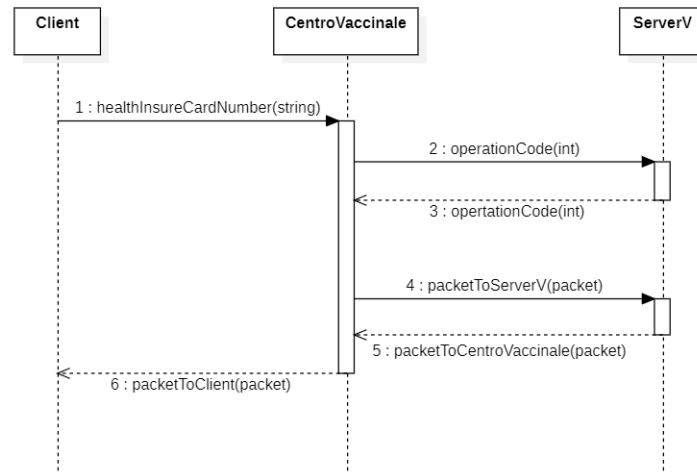


Figura 1.2: Diagramma delle sequenze dell'interazione Client-CentroVaccinale-ServerV

### 1.3.2 Interazione ClientT-ServerG-ServerV

Per aggiornare lo stato di validità di un green pass, il protocollo di livello applicazione prevede che un ClientT si connetta ad un ServerG. Ci sarà poi un'interazione "pronto-risposta" tra ClientT e ServerG che ha come unico scopo quello di comunicare al ServerG da quale client proviene la richiesta. Dopo aver inserito il numero di tessera sanitaria e valore per aggiornare la validità del green pass (0 per disattivarlo, 1 per riattivarlo), questi vengono messi all'interno di un pacchetto di tipo packet2. Il ServerG riceve il pacchetto e dopo un'interazione di tipo "pronto-risposta" con il ServerV, inoltra il pacchetto ricevuto dal ClientT. Il ServerV farà i dovuti controlli e risponderà con un pacchetto di tipo packet, scegliendo tra quattro possibili valori di ritorno:

- 0: non è stato possibile aggiornare lo stato di validità del green pass perché non esiste alcun green pass associato al numero di tessera sanitaria oppure il green pass è scaduto;
- 1: l'aggiornamento dello stato di validità del green pass è avvenuto con successo;
- 2: il client sta provando a riattivare il green pass ma questo risulta già attivo;

- 3: il client sta provando a disabilitare il green pass ma questo risulta già disabilitato.

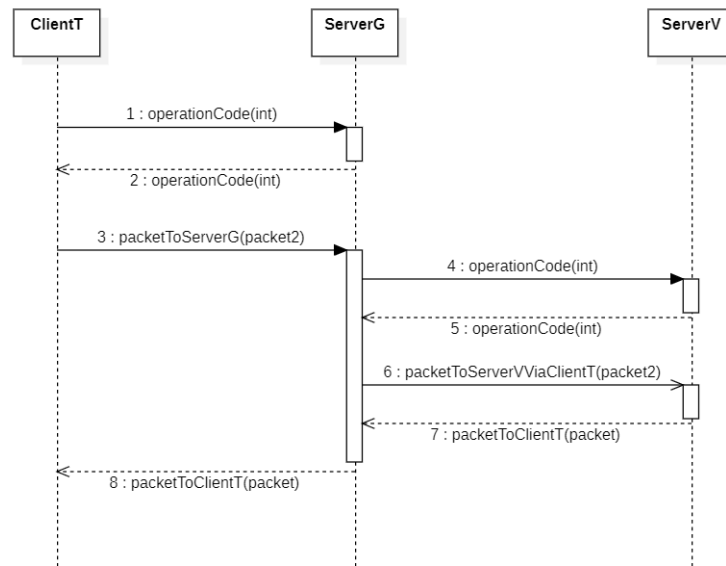


Figura 1.3: Diagramma delle sequenze dell'interazione ClientT-ServerG-ServerV

Il ServerV provvederà ad inoltrare il pacchetto al ServerG, che a sua volta provvederà ad inoltrarlo al ClientT. In base al risultato del pacchetto, il ClientT stamperà un adeguato messaggio a video.

### 1.3.3 Interazione ClientS-ServerG-ServerV

Per conoscere lo stato di validità di un green pass, il protocollo di livello applicazione prevede che un ClientS si connetta ad un ServerG. Ci sarà poi un'interazione "pronto-risposta" tra ClientS e ServerG che ha come unico scopo quello di comunicare al ServerG da quale client proviene la richiesta. Dopo aver inserito il numero di tessera sanitaria, il ClientS lo inoltrerà al ServerG. Il ServerG riceve la stringa e dopo un'interazione di tipo "pronto-risposta" con il ServerV, costruisce un pacchetto di tipo packet con il numero della tessera sanitaria ricevuto e lo invia al ServerV. Il ServerV farà i dovuti controlli e risponderà con un pacchetto di tipo packet, scegliendo tra quattro possibili valori di ritorno:

- 0: il green pass risulta essere scaduto;

- 1: il green pass risulta essere valido;
- 2: non è stato trovato alcun green pass associato al numero di tessera sanitaria fornito;
- 3: il green pass è ancora valido ma risulta essere disattivato.

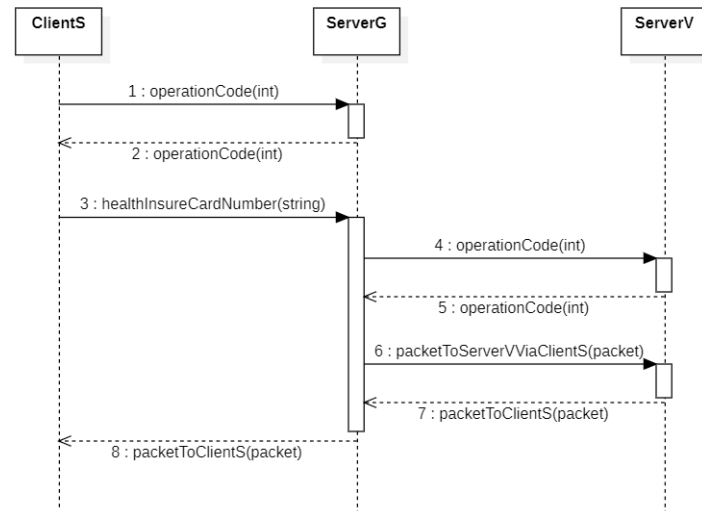


Figura 1.4: Diagramma delle sequenze dell'interazione ClientS-ServerG-ServerV

Il ServerV provvederà ad inoltrare il pacchetto al ServerG, che a sua volta provvederà ad inoltrarlo al ClientS. In base al risultato del pacchetto, il ClientS stamperà un adeguato messaggio a video.

## 1.4 Dettagli implementativi dei client

### 1.4.1 Dettagli implementativi del Client

Il codice del Client è costituito dalla sola funzione *main(...)*. Per prima cosa si richiede di inserire come parametro da riga di comando il numero di porta del centro vaccinale al quale il Client desidera collegarsi. Il valore inserito da riga di comando viene convertito in **unsigned short int** dalla funzione *stringToPort(...)*. Si richiede poi l'inserimento del numero della tessera sanitaria mediante *scanf(...)*. Si noti che la *scanf* non effettua alcun controllo sulla lunghezza della

stringa inserita, per questo si provvede ad utilizzare una funzione *checkHealthInsureCardNumber(...)* per verificare che la stringa inserita sia di venti caratteri. Successivamente si crea la socket mediante omonima system call, si inseriscono numero di porta e indirizzo IP (che sarà l'indirizzo di loopback ossia 127.0.0.1) del CentroVaccinale nella struttura di tipo **struct sockaddr\_in**, si effettua la connessione con il centro vaccinale mediante *connect(...)*, passando proprio la struttura prima citata e altri parametri. L'invio e la ricezione dei dati avviene mediante le funzioni *FullWrite(...)* e *FullRead(...)*. In particolare viene inviato il numero della tessera sanitaria al CentroVaccinale e questi risponde con un pacchetto contenente l'esito e la data a partire dalla quale sarà possibile effettuare la prossima vaccinazione. Per evitare confusioni, i valori di ritorno per ogni client sono stati mappati in apposite **enum**, un set di costanti di tipo **int** denominate opportunamente. Per il Client sono previsti due valori di ritorno:

- **VACCINE\_ALREADY\_DONE** = 0
- **VACCINE\_OK** = 1

### 1.4.2 Dettagli implementativi del ClientS

Il codice del ClientS è costituito dalla sola funzione *main(...)*. Per prima cosa si richiede di inserire come parametro da riga di comando il numero di porta del ServerG al quale il ClientS desidera collegarsi. Il valore inserito da riga di comando viene convertito in **unsigned short int** dalla funzione *stringToPort(...)*. Si richiede poi l'inserimento del numero della tessera sanitaria mediante *scanf(...)*. Si noti che la *scanf* non effettua alcun controllo sulla lunghezza della stringa inserita, per questo si provvede ad utilizzare una funzione *checkHealthInsureCardNumber(...)* per verificare che la stringa inserita sia di venti caratteri. Successivamente si crea la socket mediante omonima system call, si inseriscono numero di porta e indirizzo IP (che sarà l'indirizzo di loopback ossia 127.0.0.1) del ServerG nella struttura di tipo **struct sockaddr\_in**, si effettua la connessione con il centro vaccinale mediante *connect(...)*, passando proprio la struttura prima citata e altri parametri. L'invio e la ricezione dei dati avviene mediante le funzioni *FullWrite(...)* e *FullRead(...)*. In prima istanza c'è una sorta di handshake tra ClientS e ServerG tramite l'*operationCode*. È stato già spiegato in precedenza il motivo di questa operazione. Successivamente, il ClientS invia il numero del-

la tessera sanitaria al ServerG. Questi, dopo aver dialogato con il ServerV per verificare lo stato del green pass, ritorna un pacchetto che viene letto mediante l'apposita funzione. Per il ClientS sono previsti quattro valori di ritorno:

- **GREEN\_PASS\_EXPIRED** = 0,
- **GREEN\_PASS\_OK** = 1,
- **GREEN\_PASS\_NOT\_FOUND** = 2,
- **GREEN\_PASS\_DISABLED** = 3

### 1.4.3 Dettagli implementativi del ClientT

Il codice del ClientT è costituito dalla sola funzione *main(...)*. Per prima cosa si richiede di inserire come parametro da riga di comando il numero di porta del ServerG al quale il ClientT desidera collegarsi. Il valore inserito da riga di comando viene convertito in **unsigned short int** dalla funzione *stringToPort(...)*. Si richiede poi l'inserimento del numero della tessera sanitaria mediante *scanf(...)*. Si noti che la *scanf* non effettua alcun controllo sulla lunghezza della stringa inserita, per questo si provvede ad utilizzare una funzione *checkHealthInsureCardNumber(...)* per verificare che la stringa inserita sia di venti caratteri. In più il ClientT richiede di inserire il valore relativo allo stato di validità del green pass. Se si desidera disattivare il green pass si digiterà 0, altrimenti 1. Successivamente si crea la socket mediante omonima system call, si inseriscono numero di porta e indirizzo IP (che sarà l'indirizzo di loopback ossia 127.0.0.1) del ServerG nella struttura di tipo **struct sockaddr\_in**, si effettua la connessione con il centro vaccinale mediante *connect(...)*, passato proprio la struttura prima citata e altri parametri. L'invio e la ricezione dei dati avviene mediante le funzioni *FullWrite(...)* e *FullRead(...)*. In prima istanza c'è una sorta di handshake tra ClientT e ServerG tramite l'*operationCode*. È stato già spiegato in precedenza il motivo di questa operazione. Successivamente, il ClientT invia un pacchetto contenente il numero della tessera sanitaria più il valore di update del green pass al ServerG. Questi, dopo aver dialogato con il ServerV per verificare lo stato del green pass, ritorna un pacchetto che viene letto mediante l'apposita funzione. Per il ClientT sono previsti quattro valori di ritorno:

- **GREEN\_PASS\_NOT\_UPDATED** = 0,

- `GREEN_PASS_UPDATED = 1`,
- `GREEN_PASS_ALREADY_ACTIVE = 2`,
- `GREEN_PASS_ALREADY_DISABLED = 3`

#### 1.4.4 Dettagli implementativi di ServerG e CentroVaccinale

Un discorso a parte va fatto per ServerG e CentroVaccinale. La loro parte client consiste nella connessione al ServerV. Poiché questa funzionalità è in comune sia al ServerG che al CentroVaccinale, è stata creata un'apposita funzione chiamata *connectWithServerV(...)*. Si noti che sia il numero di porta dell'entità in questione (CentroVaccinale o ServerG) e sia il numero di porta del ServerV sono passati come argomenti da riga di comando. Questi due valori vengono poi convertiti da stringa ad unsigned short int da un'apposita funzione denominata *stringToPort(...)*.

### 1.5 Dettagli implementativi dei server

#### 1.5.1 Dettagli implementativi di ServerV

Il ServerV rappresenta il fulcro del sistema GreenPass. Viene implementato come un server iterativo, ossia un server che accoglie e soddisfa una sola richiesta alla volta, attraverso la tipica procedura a coda di attesa. Si noti che la scelta di implementare il ServerV come server iterativo ha come vantaggio quello di avere buone prestazioni su macchine poco performanti o con poche risorse a disposizione. Di contro, gestendo una richiesta alla volta, se una richiesta impiega più tempo del previsto si possono generare ritardi nel soddisfacimento delle richieste successive. Il ServerV iterativo viene implementato attraverso il ricorso alla funzione *select(...)*, che permette di monitorare più descrittori di file contemporaneamente. In particolare ci interessa monitorare l'insieme dei descrittori pronti in lettura. Semplificando al massimo, possiamo dire che il ServerV effettua le seguenti operazioni:

- dopo aver eseguito *socket(...)*, *bind(...)* e *listen(...)* dichiara un array *fd\_open[]* in cui sono contrassegnati con 1 i descrittori *fd\_open[i]* aperti. Si tiene traccia del descrittore massimo con la variabile *max\_fd*.
- inizialmente viene posto a 1 solo il descrittore di ascolto *listenSockFd* e rappresenta il descrittore massimo
- si impostano i descrittori da monitorare nell'*fset*
- si effettua la *select()*. Ogni qual volta c'è una nuova connessione, la *select* ritorna, la nuova connessione viene accettata tramite la funzione *accept(...)* e il descrittore ritornato da tale funzione viene impostato come attivo. Eventualmente viene ricalcolato il massimo descrittore.

Il ServerV utilizza l'*operationCode* per capire da quale client proviene la richiesta. L'*operationCode* può assumere i seguenti valori:

- **FROM\_CENTRO\_VACCINALE** = 0
- **FROM\_CLIENT\_T** = 1
- **FROM\_CLIENT\_S** = 2

Anche il ServerV fa utilizzo delle funzioni *FullWrite(...)* e *FullRead(...)* per l'invio e la ricezione dei dati. Per poter comunicare con il ServerV si ha la necessità di inviare l'*operationCode*. Il ServerV provvederà a re-inoltrare tale codice per confermare l'avvenuta ricezione del codice. Da qui in poi la gestione del ServerV prende una differente strada a seconda se la richiesta proviene dal CentroVaccinale, dal ClientT tramite ServerG o dal ClientS tramite ServerG.

### 1.5.2 Dettagli implementativi di ServerG e CentroVaccinale

Il CentroVaccinale e il ServerG fungono da server rispettivamente per Client, ClientS e ClientT. Vengono entrambi implementati come server concorrenti, ossia ogni richiesta viene gestita dal server con la creazione di un nuovo processo mediante *fork(...)*. Dopo aver effettuato le classiche *socket(...)*, *bind(...)* e *listen(...)*, i due "server" accettano una nuova connessione mediante *accept(...)*,



che restituisce un nuovo descrittore. Successivamente effettuano una *fork(...)*, che creerà un nuovo processo che andrà a gestire la richiesta del client, utilizzando per la comunicazione il descrittore ritornato dalla *accept*. La gestione delle richieste avviene in modalità differente poiché:

- il CentroVaccinale può ricevere richieste solo da processi Client;
- il ServerG può ricevere richieste sia da ClientS che da ClientT.

## 1.6 Manuale utente

### 1.6.1 Istruzioni per la compilazione

Per la compilazione del progetto viene messo a disposizione un makefile. Quando un progetto è composto da molti file può essere tedioso ricompilare ogni file ogni qual volta si effettua una modifica. Unix mette a disposizione l'utility *make*, che permette di automatizzare il processo di compilazione di un intero progetto. All'interno di un makefile sono contenute delle regole di dipendenza che consentono di ricompilare i sorgenti solo quando i file da cui essi dipendono vengono modificati. 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. I file sono stati compilati su una macchina con sistema operativo Windows 11 e processore Intel, utilizzando il sottosistema Windows per Linux. Come si legge dal [sito ufficiale di Microsoft](#) *“Il sottosistema Windows per Linux (WSL) è una funzionalità del sistema operativo Windows che consente di eseguire un file system Linux, insieme agli strumenti da riga di comando e alle app GUI di Linux, direttamente in Windows, insieme alle app desktop e alle app tradizionali Windows.”*. La WSL utilizza Ubuntu 20.04 LTS. Per una maggior sicurezza, i file sono stati compilati anche su una macchina virtuale con Linux Mint 20.02 "Uma".

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 e preferibilmente uno per volta.

```
rm -f *.o
rm -f *.out
gcc -c Client.c
gcc -c GreenPass.c LabUtilities.c
gcc Client.o GreenPass.o LabUtilities.o -o Client.out
gcc -c ClientS.c
gcc ClientS.o GreenPass.o LabUtilities.o -o ClientS.out
gcc -c ClientT.c
gcc ClientT.o GreenPass.o LabUtilities.o -o ClientT.out
gcc -c CentroVaccinale.c
gcc CentroVaccinale.o GreenPass.o LabUtilities.o -o CentroVaccinale.out
gcc -c ServerG.c
gcc ServerG.o GreenPass.o LabUtilities.o -o ServerG.out
gcc -c ServerV.c
gcc ServerV.o GreenPass.o LabUtilities.o -o ServerV.out
```

### 1.6.2 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 dietro cui dovrà girare il processo e il numero di porta del

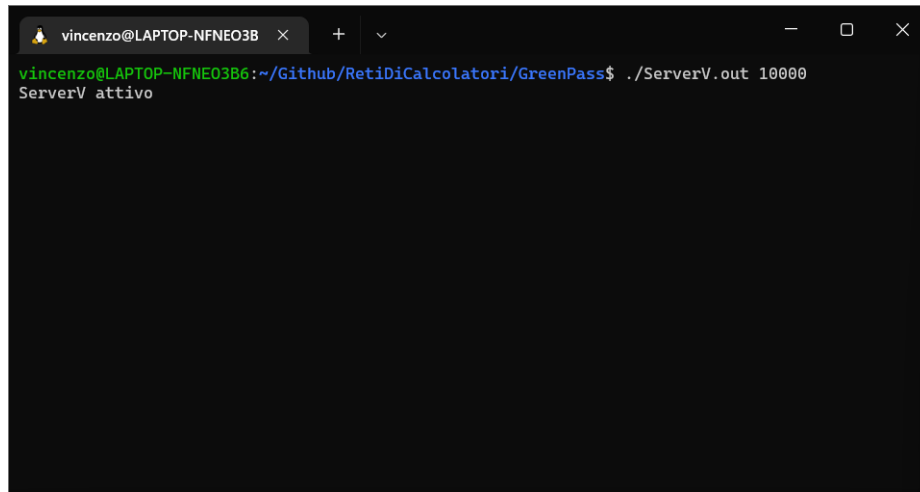
ServerV. Anche in questo caso non è necessario passare l'indirizzo IP del ServerV in quanto il suo indirizzo è l'indirizzo di loopback.

- ServerV.out riceve come parametro solo il numero di porta dietro cui deve essere eseguito. Non sono necessari altri parametri.

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

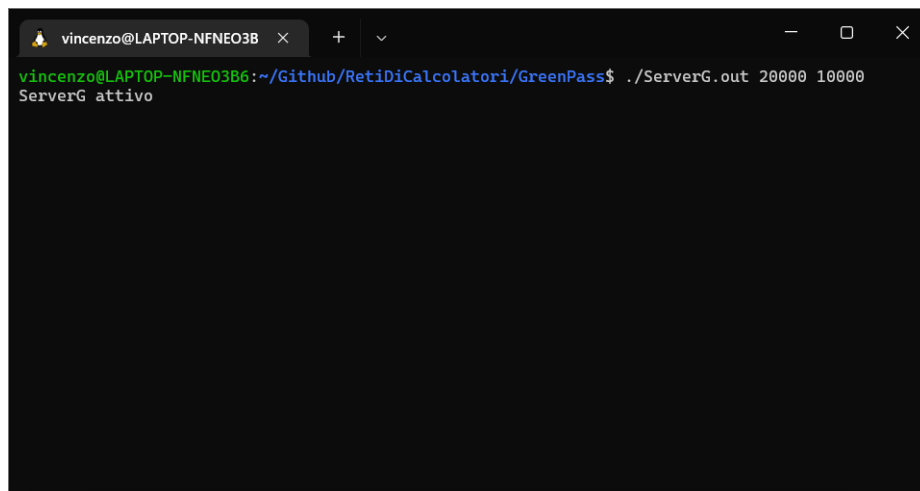
1. ./Server.out <Numero di porta>
2. ./ServerG.out <Numero di porta> <Numero di porta ServerV>
3. ./CentroVaccinale.out <Numero di porta> <Numero di porta ServerV>
4. Eseguire uno dei client in base alla richiesta che bisogna effettuare, ricordando che
  - Se si vuole eseguire il Client bisogna scrivere:  
./Client.out <Numero di porta CentroVaccinale>
  - Se si vuole eseguire o 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. Se questi non vengono rispettati, non vi è alcun modo di assicurare il corretto funzionamento del sistema proposto. Si allegano qui di seguito alcune screenshot relative all'esecuzione delle varie entità, al fine di poter mostrare come avviarle correttamente e cosa viene visualizzato durante l'esecuzione di ciascuna di esse.



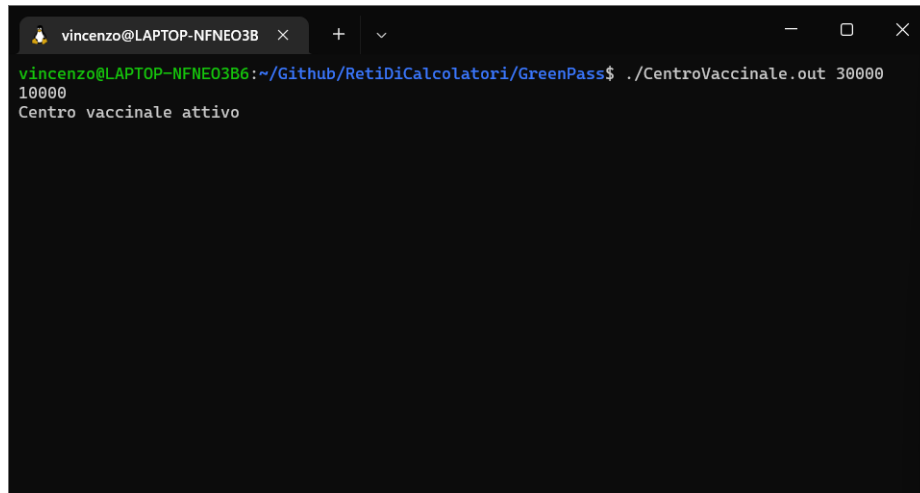
```
vincenzo@LAPTOP-NFNEO3B x + v
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetidiCalcolatori/GreenPass$ ./ServerV.out 10000
ServerV attivo
```

Figura 1.5: Avvio del processo ServerV



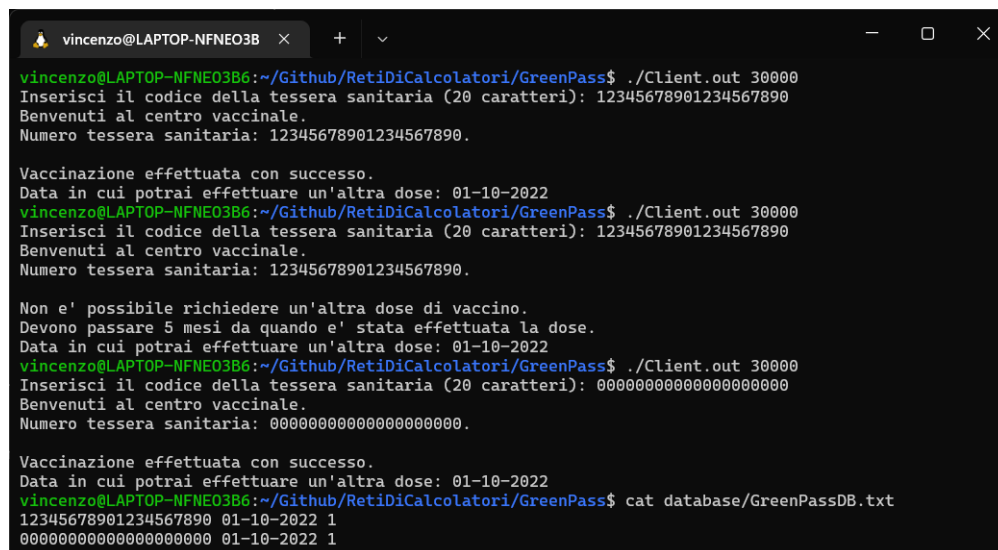
```
vincenzo@LAPTOP-NFNEO3B x + v
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetidiCalcolatori/GreenPass$ ./ServerG.out 20000 10000
ServerG attivo
```

Figura 1.6: Avvio del processo ServerG



```
vincenzo@LAPTOP-NFNEO3B x + v
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./CentroVaccinale.out 30000
10000
Centro vaccinale attivo
```

Figura 1.7: Avvio del processo CentroVaccinale



```
vincenzo@LAPTOP-NFNEO3B x + v
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./Client.out 30000
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567890
Benvenuti al centro vaccinale.
Numero tessera sanitaria: 12345678901234567890.

Vaccinazione effettuata con successo.
Data in cui potrai effettuare un'altra dose: 01-10-2022
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./Client.out 30000
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567890
Benvenuti al centro vaccinale.
Numero tessera sanitaria: 12345678901234567890.

Non e' possibile richiedere un'altra dose di vaccino.
Devono passare 5 mesi da quando e' stata effettuata la dose.
Data in cui potrai effettuare un'altra dose: 01-10-2022
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./Client.out 30000
Inserisci il codice della tessera sanitaria (20 caratteri): 00000000000000000000
Benvenuti al centro vaccinale.
Numero tessera sanitaria: 00000000000000000000.

Vaccinazione effettuata con successo.
Data in cui potrai effettuare un'altra dose: 01-10-2022
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ cat database/GreenPassDB.txt
12345678901234567890 01-10-2022 1
00000000000000000000 01-10-2022 1
```

Figura 1.8: Esecuzione del processo Client

```
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientS.out 20000
Verifica la validità del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567890
Numero tessera sanitaria: 12345678901234567890.
Il Green Pass associato alla tessera sanitaria numero 12345678901234567890 risulta ancora valido.
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientS.out 20000
Verifica la validità del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567891
Numero tessera sanitaria: 12345678901234567891.
Non è associato alcun Green Pass alla tessera sanitaria numero 12345678901234567891.
Un Green Pass può essere ottenuto solo dopo la vaccinazione.
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientS.out 20000
Verifica la validità del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 00000000000000000000
Numero tessera sanitaria: 00000000000000000000.
Il Green Pass associato alla tessera sanitaria numero 00000000000000000000 risulta scaduto.vincenzo@
LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ cat database/GreenPassDB.txt
12345678901234567890 01-10-2022 1
00000000000000000000 01-05-2022 1
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ date
Sun May 8 17:24:28 CEST 2022
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ |
```

Figura 1.9: Esecuzione del processo ClientS

```
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientT.out 20000
Aggiorna lo stato del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 010101010101010101
0 sospendi Green Pass
1 Riattiva Green Pass
0
Non è stato possibile abilitare o disabilitare il Green Pass associato alla tessera sanitaria numero 01
0101010101010101.
E' probabile che non sia associato alcun Green Pass alla tessera sanitaria o che il Green Pass sia scad
uto.
Verifica lo stato del tuo Green Pass tramite l'apposito client.
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ cat database/GreenPassDB.txt
12345678901234567890 01-10-2022 1
00000000000000000000 01-10-2022 1
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientT.out 20000
Aggiorna lo stato del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567890
0 sospendi Green Pass
1 Riattiva Green Pass
0
Il Green Pass associato alla tessera sanitaria numero 12345678901234567890 e' stato aggiornato con succ
esso.
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ cat database/GreenPassDB.txt
12345678901234567890 01-10-2022 0
00000000000000000000 01-10-2022 1
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ ./ClientT.out 20000
Aggiorna lo stato del tuo Green Pass.
Inserisci il codice della tessera sanitaria (20 caratteri): 12345678901234567890
0 sospendi Green Pass
1 Riattiva Green Pass
0
ATTENZIONE! il Green Pass associato alla tessera sanitaria numero 12345678901234567890 risulta già disa
ttivato.
Verifica lo stato del tuo Green Pass prima di effettuare qualsiasi operazione.
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ cat database/GreenPassDB.txt
12345678901234567890 01-10-2022 0
00000000000000000000 01-10-2022 1
vincenzo@LAPTOP-NFNEO3B6:~/Github/RetiDiCalcolatori/GreenPass$ |
```

Figura 1.10: Esecuzione del processo ClientT