RETI DI CALCOLATORI 2022/2023

Relazione Progetto GreenPass

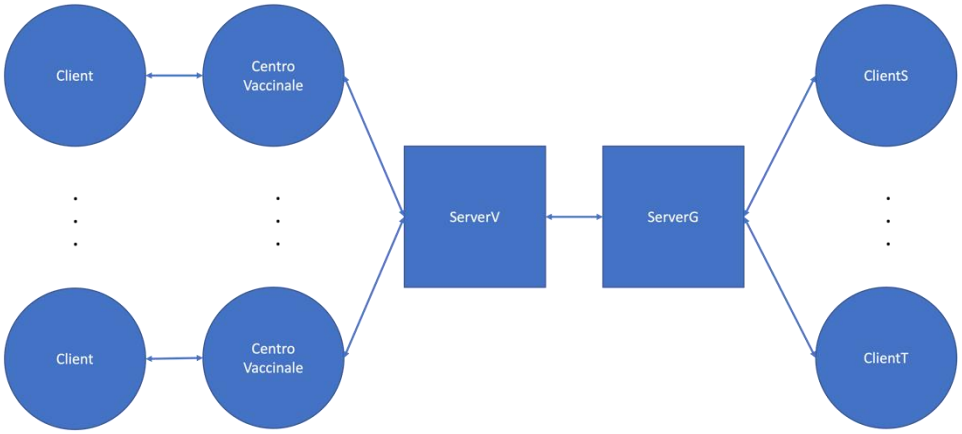
Crescenzo Esposito
N° 0124002375

Sommario

1. Descrizione del progetto

Il progetto consiste nella realizzazione di un servizio di gestione dei green pass. Il sistema permette ad un utente di comunicare il codice della propria tessera sanitaria ad un centro vaccinale tramite un client utente. Il centro vaccinale comunica il codice al serverV e invia le informazioni che caratterizzeranno il green pass, ovvero codice_tessera, data inizio, data scadenza. Il serverV elabora queste informazioni ottenute, genera la certificazione, la valida e la salva in un file che funge da database. Inoltre, il clientS può verificare la validità di un green pass inserendo il codice della tessera sanitaria, che viene inviato al serverG per la verifica. Il ServerG comunica col ServerV (gestore dei greenpass) al quale richiede il controllo della validità. Infine, il clientT può invalidare o ripristinare la validità di un green pass comunicando al serverG il contagio o la guarigione di una persona tramite il codice della tessera sanitaria.

2. Descrizione e schemi dell'architettura



L'architettura del sistema è basata sul modello client-server. Il sistema è composto da tre server e tre client. Centro Vaccinale e ServerG fungono sia da client che da server facendo così da intermediari tra il ServerV e i tre client:

- Il ServerV, il quale gestisce le informazioni inviate dal centro vaccinale e invia le informazioni al ServerG per verificare la validità del green pass.
- Il ServerG, il quale gestisce le informazioni riguardanti la validità del green pass e le informazioni di contagio o guarigione inviate dal clientT.
- Il CentroVaccinale, il quale riceve i codici delle tessere sanitarie inviati dal client utente e invia le informazioni riguardanti il green pass al ServerV.
- Il Client Utente, il quale comunica il codice della propria tessera sanitaria al centro vaccinale.
- Il ClientS, il quale verifica la validità del green pass inserendo il codice della tessera sanitaria e inviando la richiesta al ServerG.
- Il ClientT, il quale invalida o ripristina la validità del green pass comunicando al ServerG il contagio o la guarigione di una persona tramite il codice della tessera sanitaria.

3. Descrizione e schemi del protocollo applicazione

Il protocollo di applicazione utilizzato prevede l'uso del protocollo TCP/IP e il formato dei messaggi scambiati tra client e server è del tipo testo ed intero, organizzati in pacchetti implementati come struct.
Il client e il server scambiano messaggi seguendo una logica di richiesta-risposta, in cui il client invia una richiesta al server e il server risponde con un messaggio di risposta contenente le informazioni richieste.
Per quanto riguarda la gestione dei dati, il protocollo prevede l'uso di un formato di messaggio standard, in cui ogni campo è separato da un carattere di delimitazione e ogni messaggio è terminato da un carattere di fine riga.

4. Dettagli implementativi del client

Il client è stato sviluppato in linguaggio C e implementa le funzionalità di comunicazione con il server e di interfaccia utente. Vi sono 3 client.

Client Utente

Rappresenta la persona fisica che si è appena vaccinata e comunica il codice della propria tessera sanitaria al Centro Vaccinale. Vi è una verifica sull'inserimento del codice, nel caso in cui non rispetti lo standard di lunghezza verrà richiesto l'inserimento. Inserito il codice questo viene inoltrato al Centro Vaccinale che una volta ricevuto il codice invierà al client un messaggio di successo. Questo Client comunica esclusivamente con il Centro Vaccinale.

Client S

Dà la possibilità di verificare la validità di un certificato a partire da un codice di tessera sanitaria. Questo codice viene prima inoltrato al Server G, che dopo aver comunicato a sua volta col ServerV, manderà al nostro client la risposta sul certificato analizzato, valido, scaduto o non esistente.

Client T

Quest'ultimo client non solo richiederà come input il codice della tessera sanitaria ma chiederà anche un valore che può essere o 0 o 1 per attivare o disattivare un Green pass esistente. Anche in questo caso come per l'inserimento del codice vi è un controllo che non accetta valori al di fuori di 0 o 1. Il codice e il valore selezionato vengono inoltrati al Server G che provvede a comunicarli al Server V per effettuare la modifica. In questo caso non ci sono valori di risposta, il cambiamento di stato sarà visibile al ServerV, per controllare l'effettiva modifica si consiglia di usare il Client G.

5. Dettagli implementativi del server

Il server è stato sviluppato in linguaggio C e implementa le funzionalità di gestione delle connessioni dei client e di gestione delle richieste. Il server utilizza il metodo dei server concorrenti, mediante le fork le connessioni vengono gestite in un processo separato. All'avvio, il server si mette in ascolto su una porta specificata e si mette in attesa di connessioni da parte dei client. Una volta stabilita la connessione, il server riceve la richiesta del client e la processa, restituendo una risposta appropriata.

Centro Vaccinale

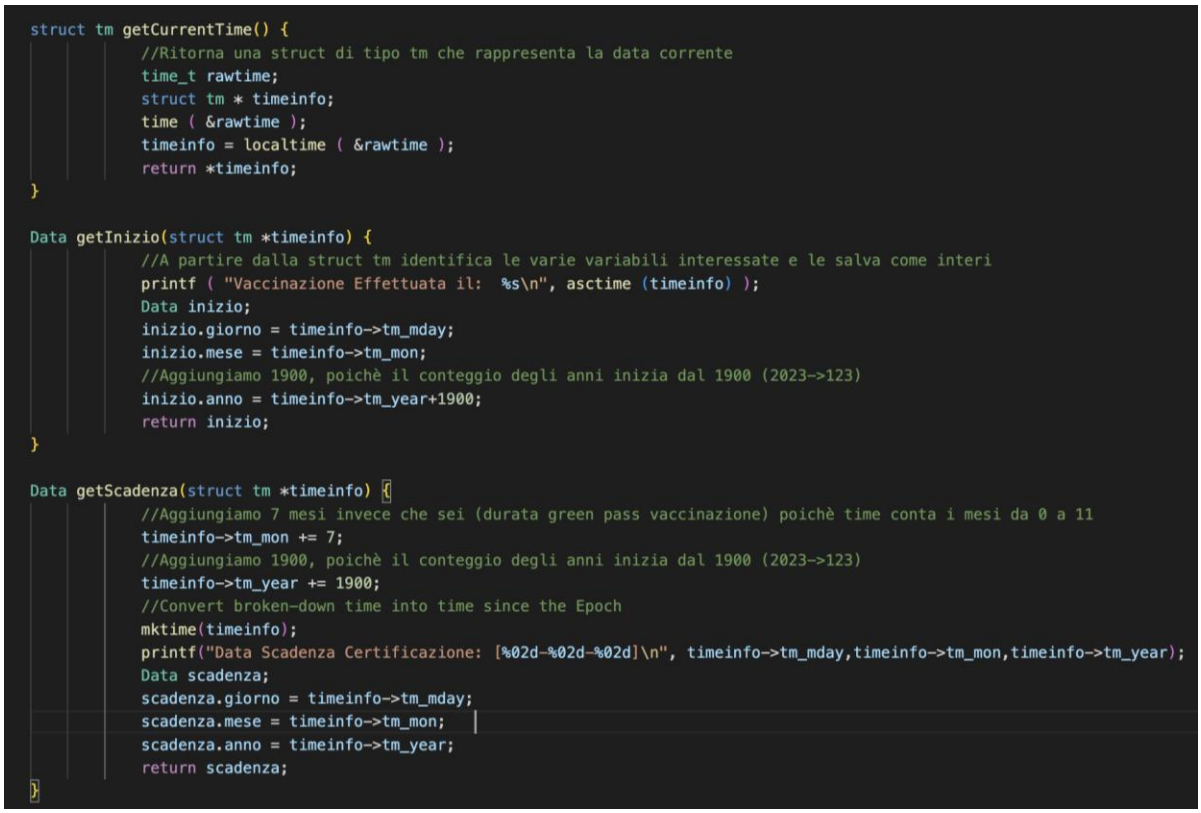
Riceve il codice tessera dell'utente e gli comunica la corretta ricezione del dato. Ha il compito di generare le date di **Inizio** e **Fine** certificazione. Ciò avviene grazie a due struct, una per la formattazione della data, in cui vengono salvati giorno, mese e anno come interi, e una per i dati riguardanti la certificazione. Queste vengono dichiarate all'interno dell'header file **grnpss.h** e inizializzate poi nel Centro Vaccinale.

```c
typedef struct {
    int giorno;
    int mese;
    int anno;
} Data;


typedef struct {
    char codice_tessera[ID_SIZE];
    Data inizio;
    Data scadenza;
    char valido;
} Certificato;

typedef struct {
    char codice_tessera[ID_SIZE];
    char valido;
} Notifica;
```

Ecco le funzioni che gestiscono le date:



```c
struct tm getCurrentTime() {
            //Ritorna una struct di tipo tm che rappresenta la data corrente
            time_t rawtime;
            struct tm * timeinfo;
            time ( &rawtime );
            timeinfo = localtime ( &rawtime );
            return *timeinfo;
}

Data getInizio(struct tm *timeinfo) {
            //A partire dalla struct tm identifica le varie variabili interessate e le salva come interi
            printf ( "Vaccinazione Effettuata il:  %s\n", asctime (timeinfo) );
            Data inizio;
            inizio.giorno = timeinfo->tm_mday;
            inizio.mese = timeinfo->tm_mon;
            //Aggiungiamo 1900, poichè il conteggio degli anni inizia dal 1900 (2023->123)
            inizio.anno = timeinfo->tm_year+1900;
            return inizio;
}

Data getScadenza(struct tm *timeinfo) {
            //Aggiungiamo 7 mesi invece che sei (durata green pass vaccinazione) poichè time conta i mesi da 0 a 11
            timeinfo->tm_mon += 7;
            //Aggiungiamo 1900, poichè il conteggio degli anni inizia dal 1900 (2023->123)
            timeinfo->tm_year += 1900;
            //Convert broken-down time into time since the Epoch
            mktime(timeinfo);
            printf("Data Scadenza Certificazione: [%02d-%02d-%02d]\n", timeinfo->tm_mday,timeinfo->tm_mon,timeinfo->tm_year);
            Data scadenza;
            scadenza.giorno = timeinfo->tm_mday;
            scadenza.mese = timeinfo->tm_mon;
            scadenza.anno = timeinfo->tm_year;
            return scadenza;
}
```

Generate dunque le date associate alla tessera viene generato un Certificato che ha le seguenti caratteristiche:
Prima di inviare il certificato compilato al ServerV inviamo un **bit di comunicazione** per avvisare il ServerV che la richiesta arriva dal CentroVaccinale.
Il valore di valido, che ha valore 1 nel caso in cui il certificato sia valido e 0 se non lo è, non viene ancora assegnato dal CentroVaccinale, verrà assegnato nel ServerV dopo una verifica nel filesystem.

ServerV

Si occupa della gestione dei Certificati, simulando un database mediante l'utilizzo di un file.dat. In questo modo i certificati vengono salvati in un unico file ed è possibile gestirne gli accessi concorrenti dato che il serverV può generare più processi figli che possono modificare o leggere in questo file. Questo è stato possibile grazie alla system call **flock**().

La prima funzione è **salvaCertificato** che riceve come parametro di input un socket connfd. Questa viene eseguita se il bit di comunicazione è uguale a 1 segnala al ServerV che la richiesta arriva dal Centro Vaccinale. Usando la funzione full_read leggiamo il certificato inoltrato dal Centro poi la funzione apre un file chiamato "file.dat" in modalità di lettura/scrittura (O_RDWR) e creazione se non esiste (O_CREAT) e con l'opzione di aggiungere i dati alla fine del file (O_APPEND). Se l'apertura del file fallisce, la funzione stampa un messaggio di errore e termina l'esecuzione.

La funzione poi esegue una lock esclusiva per garantire l'esclusività dell'accesso al file durante la scrittura del nuovo certificato.
Dopo aver eseguito la lock sul file, la funzione cerca all'interno del file un certificato con lo stesso codice della tessera del certificato appena ricevuto dal centro vaccinale. Per fare ciò, la funzione legge sequenzialmente i certificati dal file con la funzione read fino alla fine del file.
Se viene trovato un certificato con lo stesso codice tessera, la funzione controlla se sono passati almeno 4 mesi dall'ultima vaccinazione/guarigione registrata nel certificato già presente. Se sono passati 4 mesi, la funzione stampa un messaggio e consente la scrittura del nuovo certificato al posto del vecchio. In caso contrario, la funzione stampa un messaggio di errore e non permette la scrittura del nuovo certificato.
Se non viene trovato un certificato con lo stesso codice tessera, la funzione scrive il nuovo certificato alla fine del file usando la funzione write.
Infine, la funzione rilascia il lock, chiude il file e stampa il nuovo certificato usando la funzione **stampaCertificato**.

Se invece il bit di comunicazione è 0 vuol dire che la richiesta arriva dal ServerG. In questo caso è necessaria un'ulteriore verifica poiché la richiesta può essere stata inoltrata dal ClientS o dal ClientT, quindi viene usato un altro bit di comunicazione che ho chiamato bitComClient, se è 0 verrà eseguita la funzione verificaCertificato(connfd) e quindi verrà servito il ClientS. Al contrario verrà eseguita la funzione aggiornaStato(conffd) e servito il ClientT.

La funzione **verificaCertificato** legge il codice della tessera del certificato da ricercare nel file System. Se il certificato viene trovato, la funzione invia al socket connfd il valore del campo valido del certificato (che sarà 0 se certificato non è valido, 1 se è valido). Se il certificato non viene trovato, la funzione invia al ServerG il valore 2.

La funzione **aggiornaStato** invece riceve un messaggio contenente un dato di tipo Notifica (sempre dichiarato in grnpss.h) contenente il codice della tessera di un certificato e il nuovo valore del campo valido. La funzione cerca nel file il certificato associato al codice della tessera e, se lo trova, aggiorna il valore del campo valido con quello indicato nella struttura Notifica. Se il certificato non viene trovato, la funzione stampa un messaggio di errore e termina l'esecuzione. Anche in questo caso usiamo flock() per evitare la possibilità di conflitti dovuti a più client che tentano di modificare il file contemporaneamente.

Server G

È quello che instaura più connessioni dato che comunica direttamente sia col ServerV che coi due client T e S. Inizialmente il ServerG resta in attesa della connessione di uno dei due client che serve. Nel caso in cui si connetta il Client S, che invia il codice di una tessera sanitaria al ServerG per verificare la validità del green pass associato. Il ServerG richiede la verifica al ServerV e invia la risposta al ClientS.

Questo avviene grazie alla funzione **verificaCertificato**(). Il ServerG si connette al ServerV e riceve il certificato associato alla tessera richiesta. In questa funzione analizzerà il valore del campo valido, se esso è 0 invierà al ClientS il messaggio che il Green pass non è valido, se esso è 1 effettuerà una verifica sulla data attuale e sulla data di scadenza del certificato. Nel caso in cui la verifica sia positiva invierà un messaggio che il certificato è valido. Nel caso in cui il campo valido è 2 vorrà dire che il Green pass associato alla tessera richiesta non è esistente.

Nel caso in cui invece si connetta il Client T che invia una struct di tipo Notifica contenente il codice da modificare e il campo "valido" modificato. Il Server G invierà la richiesta di modifica al Server V. Per verificare se questo sia stato modificato bisognerà utilizzare il ClientS.

6. Manuale utente

Istruzioni per la compilazione
Per compilare il client e il server, è necessario disporre di un ambiente di sviluppo C/C++ e del compilatore GCC.
Per compilare i client, è necessario eseguire il seguente comando nella directory del client:

```
gcc -o ClientUtente ClientUtente.c
gcc -o ClientT ClientT.c
gcc -o ClientS ClientS.c
```

Per compilare i server, è necessario eseguire il seguente comando nella directory del server:

```
gcc -o ServerV ServerV.c
gcc -o CentroVaccinale CentroVaccinale.c
gcc -o ServerG ServerG.c
```

Istruzioni per l'esecuzione
Per eseguire il client, è necessario eseguire il seguente comando nella directory del client:
bash

Per eseguire da terminale digitare i seguenti comandi:

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