**Nomi dei Client e Server**

Client A = Client Vaccinazione;  
Centro Vaccinale = Server Centro Vaccinale;  
Sever V = Server Centrale;  
Server G = Server Assistente;  
Client S = Client Revisore;  
Client T = Client Amministratore;

**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.

**NOTE**

1. (F) Il centro vaccinale deve comunicare al Server Centrale i dati. Una volta inseriti (scritti su file) nel modo giusto il Server Centrale comunica un flag indicante che i dati sono stati ricevuti correttamente al CentroVaccinale, il quale, in caso di riuscita o errore, comunica l'esito al client Vaccinazione;
2. Il Server Centrale controlla se il codice della tessera sanitaria è stata già inserita o no;
3. (L’amministratore) il server Centrale controlla:

* Caso invalidante, vuol dire che l'utente ha il Covid, pertanto viene restituito un errore;
* Caso del codice della tessera sanitaria non presente nel file;
* Rinnovo effettuato dall’amministratore;
* L’amministratore può aggiungere una persona che è guarita all’interno del file;

1. Nel package del flag che viene mandato indietro mandiamo anche una stringa ed in base a quest’ultima interpretiamo il flag;
2. La data di scadenza non viene mandata indietro, la si controlla dal ClientS;
3. Non si può caricare la vaccinazione dopo un tot di tempo deciso a livello applicazione, <https://www.tutorialspoint.com/c-program-to-check-if-a-date-is-valid-or-not>;, Per “Leap year” guardare Udemy;;
4. Aggiungere una sorta di “Termini e condizioni d’uso” che l’utente dovrà accettare per usare correttamente il client;
5. Stringa di ritorno da parte del Server Centrale comunica il tipo di errore elencato dal punto (2);
6. Il centro vaccinale prima di aggiornare una data e fare computazione, dovrebbe contattare il serverV e quest’ultimo deve dare un esito se può procedere a calcolare la date, per poi, in base all’esito, mandare la data al Client o l’errore;
7. Il Server Centrale ha bisogno di un comando perché comunica con più Server;
8. Server Disconnesso;
9. Per il caso in cui il Centro Vaccinale si disconnette, si imposta una Select con un Timeout. Se il tempo scade allora il centro vaccinale è considerato irraggiungibile. Questo vale per tutti i client;
10. Il server centrale deve rispedire indietro al Client Revisore la validità, la scadenza, l’ultimo aggiornamento e causa;
11. Cosa rispedire indietro al Client revisore quando codice è già presente;
12. Quando ai server arriva una richiesta di controllo codice tessera sanitaria già esistente, il server centrale dovrà comunicare al mittente della richiesta 0 se non presente ed un valore >= di 0 se presente, il quale rappresenterà l’indice di riga all’interno del file in cui è stato trovato;
13. Richiesta Daytime attraverso Command

# **Documentazione**

# **Descrizione del progetto**: partendo dalla traccia descrivere cosa fa il programma e tutte le entità (client/server) del progetto;

# **Descrizione e schemi dell'architettura**: architettura che ci è stata fornita, spiegando chi fa da client e chi fa da server e le correlative connessioni, marcando i ruoli di ogni entità;

# **Descrizione e schemi del protocollo applicazione**: non inserire TCP in questa sezione!!! spiegare quali sono le regole che permettono alle nostre entità di comunicare. Quindi risulta fondamentale inserire un sequence diagram, per mostrare che tipo di dato si scambia il client con il server. In particolare, ci devono essere più sequence diagram, uno per ogni funzione del server/client;

# **Dettagli implementativi del/i client**: commentare come si è deciso di implementare i client (thread, concorrente, iterativo ecc..) e se necessario mostrare piccole porzioni di codice per le scelte implementative;

# **Dettagli implementativi del/i server**: uguale al 4;

# **Manuale utente**: - istruzioni per la compilazione - istruzioni per l'esecuzione - screenshot programma

# **Indicazioni Ferone**

1. Utilizzare C oppure un altro linguaggio, ma non è consigliabile in quanto le domande faranno comunque riferimento al C;
2. Utilizzare piattaforma Unix;
3. Consegna una settimana prima dell'appello all'indirizzo: [alessio.ferone@uniparthenope.it](mailto:alessio.ferone@uniparthenope.it) con un archivio compresso contenente:  
   - Il codice sorgente;  
   - La relazione in formato pdf;
4. Se il professore non risponde entro due giorni, sollecitare Ferone;
5. I gruppi devono sostenere l'esame nello stesso appello.

# **Protocollo Livello Applicazione**

1. Servizio di gestione green pass (Sistema)
2. Vaccinato (attore)
3. Centro vaccinale (sottosistema)
4. Amministratore (Attore)
5. Revisore (Attore)
6. Server V (Sottosistema)
7. Server G (Sottosistema)
8. Dopo una vaccinazione, un utente (Vaccinato)
9. Il server ponte riceverà un flag per riconoscere la tipologia del client
10. Il server V deve essere di tipo command
11. il centro vaccinale deve avere una coda? ??????

**Prototipo file**

|  |  |  |  |
| --- | --- | --- | --- |
| **Codice** | **Data Scadenza** | **Causa** | **Data Aggiornamento** |
| 012430 | 10/12/21 | Guarigione | 11/10/2022 |
| 049221 | 20/1/21 | Vaccinazione | 11/10/2022 |
| 321312 | - | Covid | 11/10/2022 |