Università degli Studi di Napoli Parthenope

Reti di Calcolatori e Laboratori

Anno 2022/2023

Progetto

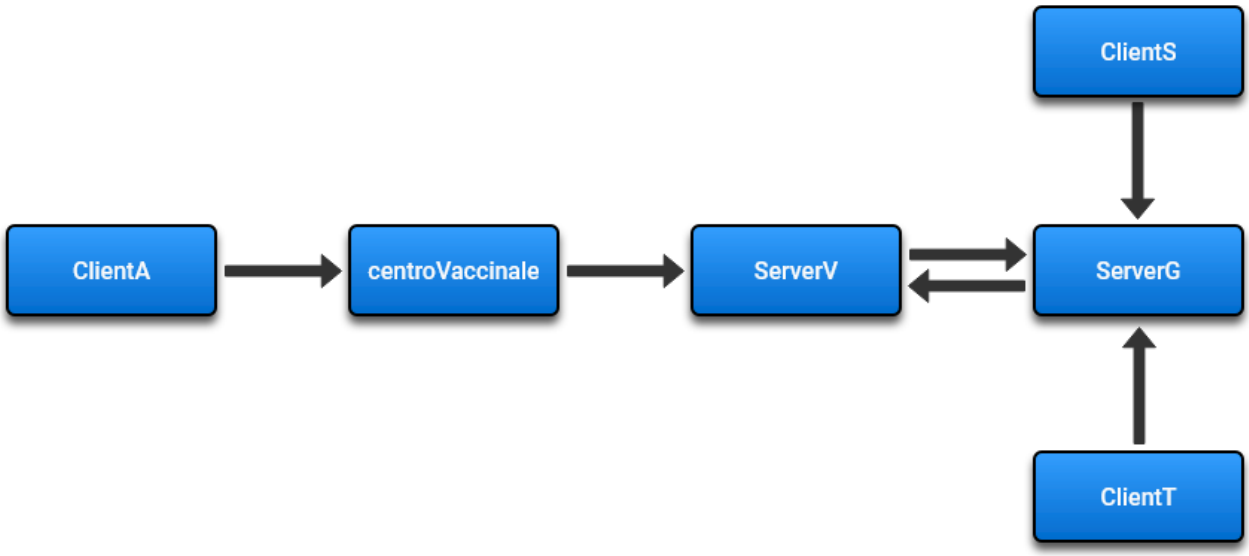
Green Pass

Studente: Capobianco Salvatore 0124/974 Professore: Aniello Castiglione

Obiettivo:

Realizzare la gestione dei green pass (gp), associati a dei numeri di tessera sanitaria univoci (ts), tramite il modello client/server, basato su server concorrenti (in grado di gestire più connessioni e quindi più richieste contemporanee).
La struttura e gli attori coinvolti sono i seguenti:

- Un operatore, una volta effettuate le vaccinazioni quotidiane, tramite il *Client A*, si collega al portale riservato del centro vaccinale e registra i codici delle ts dei cittadini vaccinati.
- Il centro vaccinale comunica al ***Server V*** (il sistema centrale, alla base del portale) il codice ricevuto dal *client A*, ed il periodo di validità del relativo gp. Quindi è sia un server (per il *client A*) che un client che si connette al ***Server V***.
- Il *Server G* è l'apposito portale che fornisce servizi a utenti comuni (richiedendoli a sua volta al ***Server V***) e medici curanti. Quindi è sia un server che un client.
- Un cittadino, tramite il *Client S*, può verificare se un gp è valido, inviando l'associato numero di ts, al *Server G*.
- Un medico curante, tramite il *Client T*, può revocare o ripristinare la validità di un gp, a causa, rispettivamente, del contagio o guarigione del suo paziente. Quindi, comunica al *Server G* il relativo numero di tessera sanitaria.



Schema del protocollo applicazione

Protocollo applicazione:

- *Client A* si connette al *Centro Vaccinale* e gli inoltra la ts;
- *Centro Vaccinale* quando riceve una ts, calcola la scadenza del gp, e inoltra quest'ultimo al ***Server V***;
- ***Server V*** mantiene i gp;
- *Client S* e *Client T* si connettono al *Server G* inviandogli una ts. *Client S* richiede il gp associato, mentre il *Client T* può validare o annullare il gp associato;
- *Server G* riceve le richieste da *Client S* e *Client T* e le inoltra a ***Server V***;
- ***Server V*** riceve le richieste di *Server G* e gli risponde;
- *Server G* una volta ricevute le risposte da ***Server V*** le inoltra al *Client* che le ha richieste;

Logica del pacchetto applicazione:

- Tessera Sanitaria -> Stringa di 20 caratteri (**campo chiave** che identifica univocamente)
- Green Pass -> Tempo scadenza
- Servizio -> Indica il tipo di servizio richiesto a ***Server V***:
    - “-1” - aggiunta gp;
    - “0” - verifica gp;
    - “1” - convalida gp;
    - “2” - revoca gp.

Quando ***Server V*** riceve una richiesta di tipo 1 (convalida), sposta l’indice di lettura con la funzione fseek(), all’inizio del file e inizia a scorrere i gp, alla ricerca di quello richiesto. Se è già presente, lo rinnova:

- nel caso in cui il servizio, fosse precedentemente impostato a “3”, il gp viene rinnovato di 2 giorni in seguito ad un tampone;
- altrimenti viene rinnovato di 6 mesi, in seguito alla guarigione;
- oppure, non è mai stata incontrata prima d’ora quella ts; pertanto viene registrata nel file (database), e associata ad un gp con validità 2 giorni (un cittadino mai vaccinato e mai contagiato/guarito ha effettuato il suo primo tampone).

```
case 1:     // TAMPONE O COMUNICAZIONE GUARIGIONE
  // Il Client T, tramite Server G, richiede la convalida di un green pass (associato alla tessera sanitaria passata)

  int cgp = 0; // counter di green pass all'interno del file (offset, scostamento)
  printf("Convalida green pass in corso...\n");

  fseek(file, 0, SEEK_SET); // dall'inizo del file:
  while(fread(&temp, sizeof(struct green_pass), 1, file) == 1){ // per ogni gp memorizzato sul file
    if(strcmp(temp.ts, received.ts) == 0){ // se è associato alla stessa tessera (è già presente):
      if(temp.servizio == 3){ // nel caso in cui, precedentemente, sia stato aggiunto come gp da tampone
        temp.scadenza = time(NULL) + EXPIRATION2; // imposta 2 giorni di validità
      }else{
        temp.scadenza = time(NULL) + EXPIRATION6; // altrimenti 6 mesi di validità (guarigione)
      }
      printf("Green Pass convalidato!\n");
      // sovrascrivi nella opportuna posizione all'interno del file
      fseek(file, cgp*sizeof(struct green_pass), SEEK_SET);
      sem_wait(access);
      fwrite(&temp, sizeof(struct green_pass), 1, file);
      sem_post(access);
      close(conn_fd);
      exit(0);
    }else { cgp++; } // passa al successivo gp nel file
```

```
// nel caso in cui non sia nel file dei vaccinati, la tessera sanitaria ottiene un green pass valido 2 giorni (ha effettuato un tampone)
strcpy(temp.ts, received.ts);
temp.scadenza = time(NULL) + EXPIRATION2;
temp.servizio = 3;
printf("Tessera Sanitaria: %s\n", temp.ts);
printf("Scadenza: %.24s\r\n", ctime(&temp.scadenza));
sem_wait(access);
fseek(file,0,SEEK_END); // aggiungi in coda
fwrite(&temp, sizeof(struct green_pass), 1, file);
sem_post(access);

close(conn_fd);
exit(0);
break;
```

Quando invece, il **Server V** riceve una richiesta di tipo “2” (invalida, revoca), ricerca il gp all’interno del file:

- se è presente, lo annulla;
- se non è presente, lo aggiunge ma già scaduto (sarà attivato, in seguito, con la guarigione del cittadino)

```
case 2:     // CONTAGIO
   // Il Client T, tramite Server G, richiede l'invalida di un green pass (associato alla tessera sanitaria passata)

   printf("Invalidamento green pass in corso...\n");
   int cgp = 0; // counter di green pass all'interno del file (offset, scostamento)
   fseek(file, 0, SEEK_SET); // dall'inizo del file:
   while(fread(&temp, sizeof(struct green_pass), 1, file) == 1){ // per ogni gp memorizzato sul file
      if(strcmp(temp.ts, received.ts) == 0){ // se è associato alla stessa tessera (è già presente):
         temp.scadenza = time(NULL) - EXPIRATION2;
         printf("Green Pass invalidato!\n");
         // sovrascrivi nella opportuna posizione all'interno del file
         fseek(file, cgp*sizeof(struct green_pass), SEEK_SET);
         sem_wait(access);
         fwrite(&temp, sizeof(struct green_pass), 1, file);
         sem_post(access);
         close(conn_fd);
         exit(0);
      }else { cgp++; } // passa al successivo gp nel file
   }
   sem_post(access);
```

```
// nel caso in cui non sia nel file dei vaccinati, la tessera sanitaria viene inserita comunque
// (una volta guarito avrá il greenPass da guarigione da Covid)

strcpy(temp.ts, received.ts);
temp.scadenza = time(NULL) - EXPIRATION2;
printf("Tessera Sanitaria: %s\n", temp.ts);
printf("Scadenza: %.24s\r\n", ctime(&temp.scadenza));
sem_wait(access);
fseek(file, 0, SEEK_END);
fwrite(&temp,sizeof(struct green_pass), 1, file);
sem_post(access);
```

Client T riceve come argomento oltre alla tessera sanitaria anche o “V” o “I” per scegliere se “Validare” o “Invalidare” il gp.

```
char valid[] = "V";
char invalid[] = "I";

if(argc != 4){
   fprintf(stderr,"usage: %s <IPaddress> <TesseraSanitaria> <V or I>\n", argv[0]);
   exit(1);
}

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

strcpy(greenPass.tessera, argv[2]);
if (strcmp(argv[3], valid) == 0)
   greenPass.servizio = 1;
else if (strcmp(argv[3], invalid) == 0)
   greenPass.servizio = 2;
else{
   printf("Scelta %s non valida...Riprovare...", argv[3]);
   exit(1);
```

Compilazione ed esecuzione

Istruzioni per la compilazione:

- gcc -c -o wrapper.o wrapper.c
- gcc -o ClientA clientA.c wrapper.o
- gcc -o CentroVaccinale centroVaccinale.c wrapper.o
- gcc -o ServerV serverV.c wrapper.o
- gcc -o ServerG serverG.c wrapper.o
- gcc -o ClientS clientS.c wrapper.o
- gcc -o ClientT clientT.c wrapper.o

Istruzioni per l'esecuzione:

1. ./ServerV
2. ./CentroVaccinale "*indirizzo ServerV*"
3. ./ClientA "*indirizzo di CentroVaccinale*" "*tessera sanitaria*"
4. ./ServerG "*indirizzo ServerV*"
5. ./ClientS "*indirizzo ServerG*" "tessera sanitaria"
6. ./ClientT "*indirizzo ServerG*" "tessera sanitaria" ("V" or "I")