UNIVERSITÀ DEGLI STUDI DI NAPOLI PARTHENOPE

DIPARTIMENTO DI SCIENZE E TECNOLOGIE
CORSO DI RETI DI CALCOLATORI E LABORATORIO DI RETI DEI CALCOLATORI

Traccia - Università

CATEGORIA: Gestionale

CODICE GRUPPO: jajl3pkezpi

DOCENTE

Emmanuel Di Nardo

CANDIDATI

Viscillo Nicola 0124002557
Galiero Nicola 0124002671

Anno accademico 2023 - 2024

Indice

1 Descrizione del progetto

Scrivere un'applicazione client/server parallelo per gestire gli esami universitari.

La **segreteria**:

- Inserisce gli esami sul server dell'università (salvare in un file o conservare in memoria il dato);
- Inoltra la richiesta di prenotazione degli studenti al server universitario;
- Fornisce allo studente le date degli esami disponibili per l'esame scelto dallo studente.

Lo **studente**:

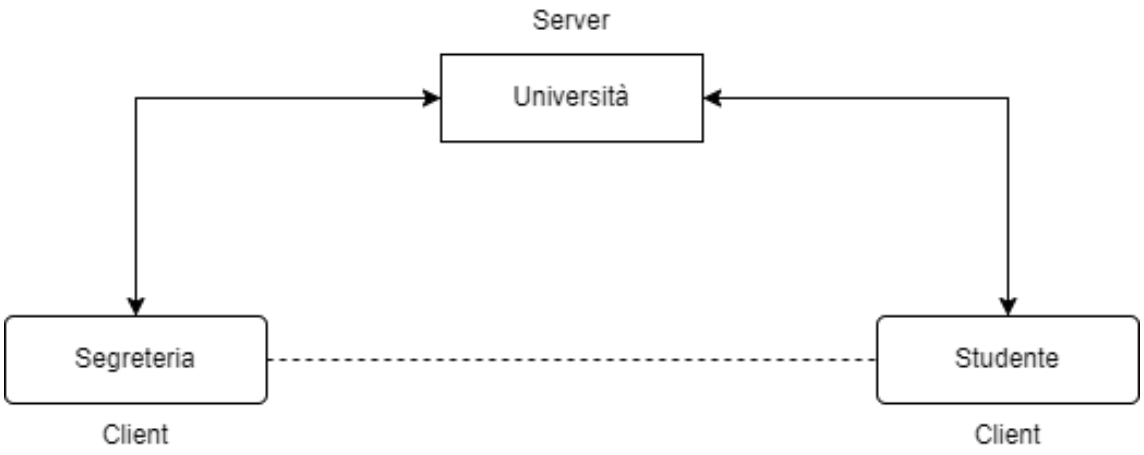
- Chiede alla segreteria se ci siano esami disponibili per un corso;
- Invia una richiesta di prenotazione di un esame alla segreteria.

Il **server universitario**:

- Riceve l'aggiunta di nuovi esami;
- Riceve la prenotazione di un esame.

Il server universitario ad ogni richiesta di prenotazione invia alla segreteria il numero di prenotazione progressivo assegnato allo studente e la segreteria a sua volta lo inoltra allo studente.

2 Descrizione e schema dell'architettura



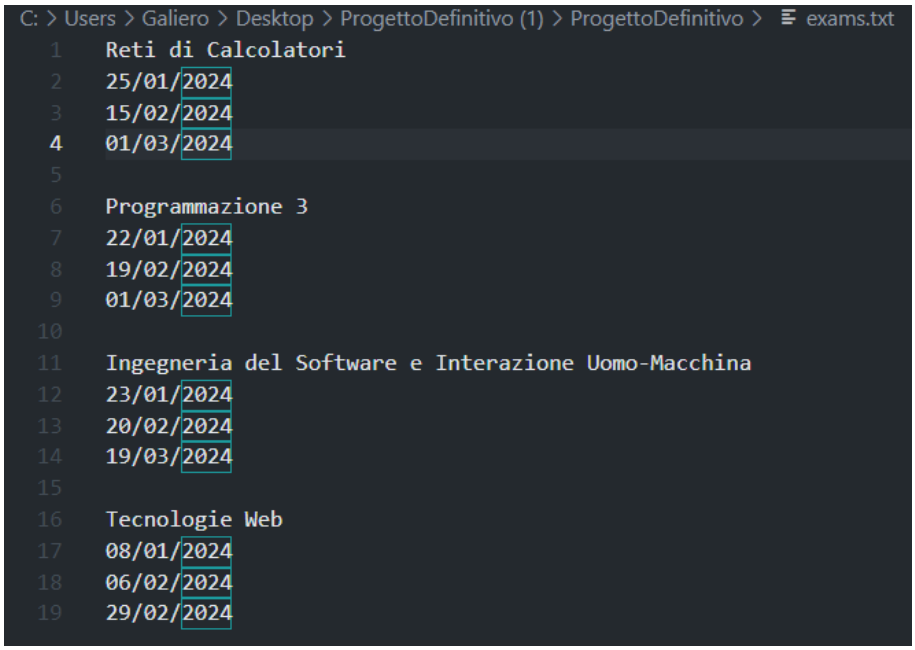
3 Dettagli implementativi dei client/server

3.1 Linguaggio di programmazione

I codici sono scritti in C, un linguaggio di programmazione di alto livello noto per le sue prestazioni elevate e una sintassi semplice. Il linguaggio C è particolarmente adatto per l'implementazione di applicazioni client-server grazie alla sua efficienza e facilità d'uso.

3.2 Persistenza dei dati

Nel nostro progetto, non abbiamo implementato l'uso di un database; invece, abbiamo adottato un approccio basato su file. In particolare, il file "exams.txt" conserva le informazioni sugli esami creati e registrati dalla segreteria. Gli studenti possono successivamente visualizzare questi dati direttamente sul terminale per effettuare prenotazioni.



Screenshot exams.txt

3.3 Modello di comunicazione utilizzato

Il codice implementato si avvale del protocollo TCP/IP per consentire la comunicazione efficace tra i client e il server. Questo protocollo di rete, noto per la sua affidabilità, garantisce la consegna sicura dei dati tra i dispositivi coinvolti nel processo.

3.4 Gestioni delle sessioni

Ogni volta che un utente desidera accedere al server dell'università per aggiungere o visualizzare gli esami di ciascuna materia, è necessario eseguire il codice. Questo approccio è stato adottato per assicurare che ogni utente, durante la prima interazione con il nostro progetto, viva un'esperienza simile a quella di utilizzare il programma per la prima volta.

4 Parti rilevanti del codice sviluppato

```
//Funzione per gestire le richieste degli studenti
void studentRequest(int client_socket) {
  int request_type;
  recv(client_socket, &request_type, sizeof(request_type), 0);

  if (request_type == 2) { //Se la richiesta e' per prenotare un esame
     char exam_name[MAX_EXAM_LENGTH]; //Buffer per il nome dell'esame
     recv(client_socket, exam_name, sizeof(exam_name), 0);

     int dates_count = 0; //Contatore per il numero di date disponibili
     char available_dates[3][MAX_DATE_LENGTH]; //Buffer per le data
         disponibili

     for (int i = 0; i < exams_count; i++) { //Ciclo per cercare l'esame
         richiesto
        if (strcmp(exams[i].exam_name, exam_name) == 0) { //Se l'esame
            stato trovato
            for (int j = 0; j < 3; j++) { //Ciclo per copiare le date
                disponibili
              if (strcmp(exams[i].dates[j], "") != 0) { //Se la data
                  disponibile
                 strncpy(available_dates[dates_count], exams[i].dates[j],
                     MAX_DATE_LENGTH - 1); //Copia della data
                 available_dates[dates_count][MAX_DATE_LENGTH - 1] = '\0';
                     //Aggiunta del terminatore di stringa
                 dates_count++;
               }
            }
            break; //Uscita dal ciclo una volta trovato l'esame
        }
     }

    send(client_socket, &dates_count, sizeof(dates_count), 0); //Invio del
        numero di date disponibili al client

    for (int i = 0; i < 3; i++) { //Ciclo per inviare le date disponibili
        al client
        int string_length = strlen(available_dates[i]) + 1; //Lunghezza
            della stringa

        send(client_socket, &string_length, sizeof(string_length), 0);
            //Invio della lunghezza della stringa
        send(client_socket, available_dates[i], string_length, 0); //Invio
            della data
        }

    int choice; //Scelta dell'utente
    recv(client_socket, &choice, sizeof(choice), 0); //Ricezione della
        scelta dell'utente

    if (choice >= 1 && choice <= dates_count) { //Se la scelta    valida
        booking_number++; //Incremento del numero di prenotazioni
        send(client_socket, &booking_number, sizeof(booking_number), 0);
            //Invio del numero di prenotazione al client

        //Invia il nome dell'esame e la data di prenotazione
        send(client_socket, exam_name, sizeof(exam_name), 0);
        send(client_socket, available_dates[choice - 1],
            strlen(available_dates[choice - 1]) + 1, 0);

```

```
46          printf("Booking␣number␣%d␣for␣exam␣'%s'␣on␣date␣%s\n",
                booking_number, exam_name, available_dates[choice - 1]);
47
48      } else { //Se la scelta non    valida
49          int confirmation = 0; //Conferma di fallimento
50          send(client_socket, &confirmation, sizeof(confirmation), 0);
                //Invio della conferma al client
51          }
52      }
53  }
```

```c
//Funzione per gestire le richieste della segreteria
void secretariatRequest(int client_socket) {
  int request_type;   //Variabile per il tipo di richiesta

  //Ricezione del tipo di richiesta
  recv(client_socket, &request_type, sizeof(request_type), 0);

  if (request_type == 1) { //Se la richiesta e' per aggiungere un nuovo esame
    char exam_name[MAX_EXAM_LENGTH]; //Buffer per il nome dell'esame
    char dates[3][MAX_DATE_LENGTH]; //Buffer per le date dell'esame

    //Ricezione del nome dell'esame dal client
    recv(client_socket, exam_name, sizeof(exam_name), 0);

    int dates_count = 0; //Contatore per il numero di date ricevute

    //Ricezione del numero di date
    recv(client_socket, &dates_count, sizeof(dates_count), 0);

    for (int i = 0; i < dates_count; i++) { //Ciclo per trovare le date
        int date_length;

        //Ricezione della lunghezza della data
        recv(client_socket, &date_length, sizeof(date_length), 0);

        if (date_length > 0) { //Se la lunghezza della data    valida
            char date[MAX_DATE_LENGTH]; //Buffer per la data

            //Ricezione della data dal client
            recv(client_socket, date, date_length, 0);

            date[date_length] = '\0'; //Aggiunta del terminatore di stringa

            strncpy(exams[exams_count].dates[i], date, MAX_DATE_LENGTH - 1); //Copia della data nella struttura

            exams[exams_count].dates[i][MAX_DATE_LENGTH - 1] = '\0'; //Aggiunta del terminatore di stringa
            } else { //Se la lunghezza della data non    valida
            exams[exams_count].dates[i][0] = '\0'; //Impostazione della data come vuota
            }
        }

    strcpy(exams[exams_count].exam_name, exam_name); //Copia del nome dell'esame nella struttura

    exams_count++;

    int response = 1; //Risposta per la conferma della ricezione
    send(client_socket, &response, sizeof(response), 0); //Invio della conferma al client
    }
}
```

5 Manuale utente con istruzioni di compilazione ed esecuzione

All'interno della repository, nella cartella "Codice" troviamo vari script, tra cui i principali: "server.c", "student.c" e "secretariat.c" .

Come prima cosa da fare bisogna eseguire il file "server.c". Per fare ciò si devono inserire i seguenti comandi sul terminale:

1. gcc server.c wrapper.c -o server
2. ./server

Successivamente, bisogna eseguire il file "secretariat.c" in quanto non posso eseguire student.c, poichè non ho nessun esame prenotabile.

1. gcc secretariat.c wrapper.c -o secretariat
2. ./secretariat

Una volta che il server è in ascolto sulla porta 1024, si può nuovamente eseguire "secretariat.c"per inserire un nuovo esame o eseguire il file "student.c" per effettuare una prenotazione:

1. gcc student.c wrapper.c -o student
2. ./student