Documentazione del Progetto di:
Reti Di Calcolatori

Gaetano Maisto: 0124002569

A.A. 2023/2024

Indice

Capitolo 1

Descrizione del Progetto

Questo progetto implementa un sistema client-server per la gestione degli esami universitari. Il server gestisce le richieste di inserimento, visualizzazione e prenotazione degli esami, mentre i client (segreteria e studenti) interagiscono con il server per eseguire queste operazioni.

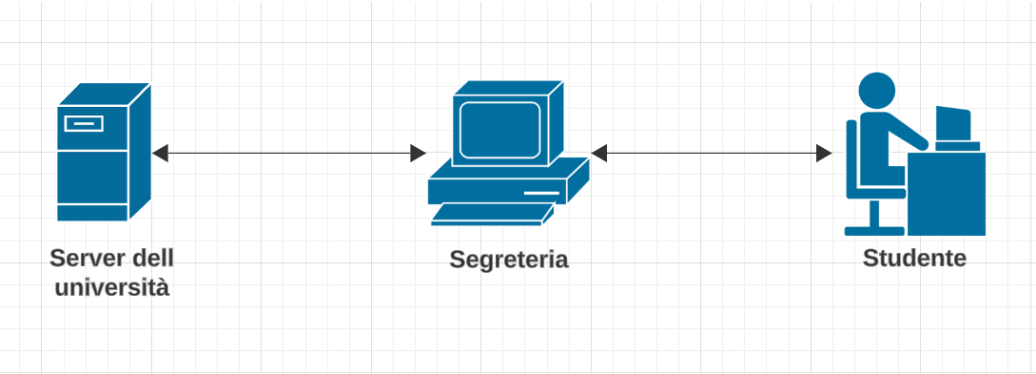
Capitolo 2

Descrizione e Schema dell'Architettura

Il sistema di gestione esami è composto da tre componenti principali: il server universitario, la segreteria e il client studente. Ogni componente comunica con gli altri tramite socket TCP/IP. Di seguito viene fornita una descrizione dettagliata di ciascun componente:

- **Server Universitario**: Questo componente gestisce la lista degli esami disponibili e le prenotazioni degli studenti. Riceve richieste dalla segreteria per aggiungere nuovi esami e per gestire le prenotazioni.
- **Segreteria**: La segreteria funge da intermediario tra gli studenti e il server universitario. Riceve richieste dagli studenti per le date degli esami e per le prenotazioni, e inoltra queste richieste al server universitario.
- **Client Studente**: Questo componente permette agli studenti di richiedere le date degli esami e di prenotare gli esami. Comunica direttamente con la segreteria.

2.1 Schema dell'Architettura



Capitolo 3

Dettagli Implementativi del Client/Server

3.1 Server

Il server è implementato nel file `Server_uni.py`. Utilizza il modulo `socket` per la comunicazione di rete e `pickle` per la serializzazione degli oggetti.

Listing 3.1: Creazione del socket del server

```
server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server_socket.bind(ADDR)
server_socket.listen(5)
```

3.2 Segreteria

La segreteria è implementata nel file `Segreteria.py`. Si connette al server universitario e gestisce le richieste degli studenti.

Listing 3.2: Connessione al server universitario

```
segreteria_socket.connect(ADDR)
```

3.3 Studente

Il client studente è implementato nel file `Studente.py`. Si connette alla segreteria per richiedere le date degli esami e prenotare esami.

Listing 3.3: Connessione alla segreteria

```
studente_socket.connect(ADDR)
```

Capitolo 4

Parti Rilevanti del Codice Sviluppato

4.1 Server

Listing 4.1: Funzione per gestire le richieste dei client

```python
def gestisci_client(conn, addr):
    print(f"Connessione tramite porta {addr}")
    try:
        while True:
            # Riceve la richiesta dal client
            richiesta = conn.recv(1024).decode(FORMAT)
            if richiesta == "DATE_ESAMI":
                # Gestisce la richiesta di date degli esami
                nome_esame = conn.recv(1024).decode(FORMAT)
                esame_trovato = None
                for esame in Lista_esami:
                    if esame.nome_esame == nome_esame:
                        esame_trovato = esame
                        break
                if esame_trovato:
                    date_disponibili = ', '.join(esame_trovato.date_disponibili)
                    conn.sendall(date_disponibili.encode(FORMAT))
                else:
                    conn.sendall(f"Esame {nome_esame} non trovato.".encode(FORMAT))
            elif richiesta == "INSERISCI_ESAME":
                # Gestisce l'inserimento di un nuovo esame
                esame_serializzato = conn.recv(1024)
                esame = pickle.loads(esame_serializzato)
                Lista_esami.append(esame)
                salva_esami()
                conn.sendall(f"Esame {esame.nome_esame} aggiunto con successo.".encode(FORMAT))
            elif richiesta == "PRENOTAZIONE_ESAME":
                # Gestisce la prenotazione di un esame
                prenotazione_serializzata = conn.recv(1024)
                prenotazione = pickle.loads(prenotazione_serializzata)
                # Logica per gestire la prenotazione
                conn.sendall(f"Prenotazione per {prenotazione['nome_esame']} ricevuta.".encode(FORMAT))
            elif richiesta == DISCONNESSIONE:
                # Gestisce la disconnessione del client
                print(f"Disconnessione da {addr}")
                break
    except socket.error as e:
        print(f"Errore di connessione: {e}")
    finally:
        conn.close()
        print(f"Connessione chiusa con {addr}")
```

4.2 Segreteria

Listing 4.2: Funzione per gestire le richieste degli studenti

```python
def gestisci_richieste_studenti(conn, addr):
    global connessioni_attive
    connessioni_attive += 1
    print(f"Connessione stabilita con {addr}. Connessioni attive: {connessioni_attive}")
    try:
        while True:
            richiesta = conn.recv(1024).decode(FORMAT)
            if richiesta == "PRENOTAZIONE_ESAME":
                # Gestisce la prenotazione di un esame
                prenotazione = conn.recv(5000)
                segreteria_socket.sendall("PRENOTAZIONE_ESAME".encode(FORMAT))
                segreteria_socket.sendall(prenotazione)
                risposta = segreteria_socket.recv(5000)
                conn.sendall(risposta)
                print(f"Prenotazione effettuata")
            elif richiesta == "DATE_ESAMI":
                # Gestisce la richiesta delle date degli esami
                nome_esame = conn.recv(1024).decode(FORMAT)
                print(f"Richiesta date esami per {nome_esame}")
                segreteria_socket.sendall("DATE_ESAMI".encode(FORMAT))
                segreteria_socket.sendall(nome_esame.encode(FORMAT))
                date_disponibili = segreteria_socket.recv(1024).decode(FORMAT)
                conn.sendall(date_disponibili.encode(FORMAT))
            elif richiesta == DISCONNESSIONE:
                # Gestisce la disconnessione del client
                print(f"Disconnessione da {addr}")
                break
    except socket.error as e:
        print(f"Errore di connessione: {e}")
    finally:
        conn.close()
        connessioni_attive -= 1
        print(f"Connessione chiusa con {addr}. Connessioni attive: {connessioni_attive}")
```

4.3 Studente

Listing 4.3: Funzione per richiedere le date degli esami

```
def richiesta_esami():
    try:
        studente_socket.sendall("DATE_ESAMI".encode(FORMAT))
        nome_esame = input("Inserisci il nome dell'esame: ")
        nome_esame = nome_esame.upper()
        if not nome_esame:
            print("Il nome dell'esame non può essere vuoto.")
            return

        studente_socket.sendall(nome_esame.encode(FORMAT))

        date_disponibili = studente_socket.recv(1024).decode(FORMAT)
        print(f"Le date disponibili per {nome_esame} sono: {date_disponibili}")
    except socket.error as e:
        print(f"Errore durante la richiesta degli esami: {e}")
```

Capitolo 5

Manuale Utente

5.1 Compilazione e Esecuzione

5.1.1 Avviare il terminale

Apri il terminale sul tuo sistema operativo, assicurandoti di avere tutti i privilegi per eseguire i comandi.

5.1.2 Avvia tre diverse finestre nel terminale

Apri tre finestre sul terminale cosi da poter avviare i tre file python separati

5.1.3 Avvia in ordine nelle tre finestre i file python

Esegui in ordine nelle tre diverse finestre i seguenti comandi:

```
python Server_uni.py
```

```
python Segreteria.py
```

```
python Studente.py
```