Università degli Studi
di Napoli Parthenope

Documentazione del Progetto di Reti Di Calcolatori: Server universitario

Gaetano Maisto: 0124002569

A.A. 2023/2024

Indice

Capitolo 1

Descrizione del Progetto

1.1 Introduzione

Il progetto è basato sulla progettazione e lo sviluppo di un applicazione client-server per la gestione di esami universitari. L' obbiettivo è quello di creare un sistema efficiente che permetta la comunicazione tra la segreteria, gli studenti e il server universitario, cosi da avere una gestione ottimale degli esami universitari. Ho sviluppato il progetto utilizzando python.

1.2 Obiettivi

I principali obiettivi del progetto sono:

Interfaccia intuitiva per lo studente: Creare un interfaccia intuitiva per lo studente in modo da poter richiedere le date o prenotarsi a un esame

Creare una segreteria che faccia da tramite: Creare una segreteria che riesca a collegare il server universitario a uno o più studenti contemporaneamente

Implementare un server Universitario: Implementare un server che sia in grado di registrare gli esami ricevuti dalla segreteria e poter gestire le richieste derivanti dallo studente

Capitolo 2

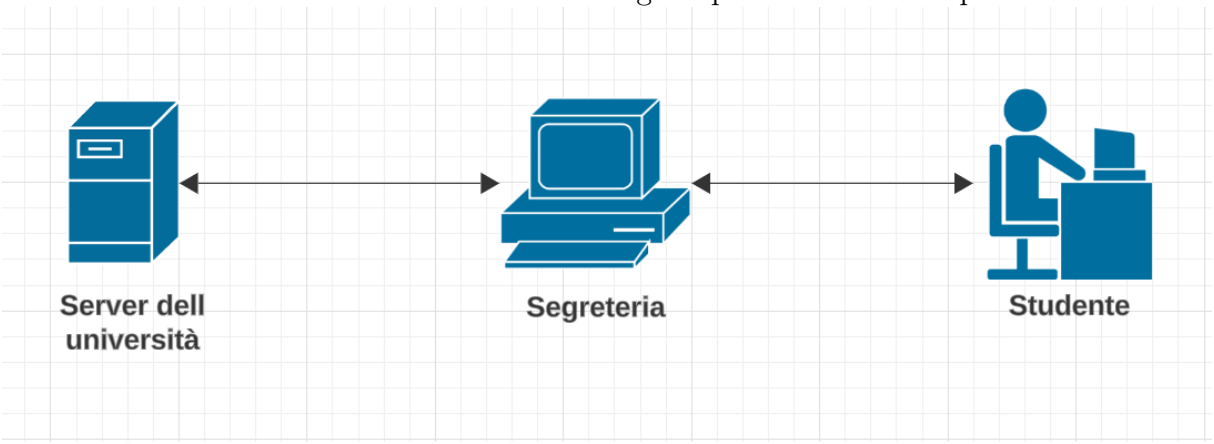
Descrizione e Schema dell'Architettura

L'architettura del progetto è composta da tre elementi che come detto prima interagiscono tra di loro con un sistema client/server tramite delle socket TCP/IP. I tre elementi sono il server universitario, la segreteria e il client studente. Di seguito viene fornita una descrizione dettagliata di ciascun componente:

- **Client Studente**: Questo componente permette agli studenti di effettuare due operazioni, richiedere le date degli esami inserendo il nome dell'esame o di fare una prenotazione a un esame fornendo il proprio id studente, il nome dell esame e la data a cui vuole prenotarsi. Comunica col server universitario utilizzando la segreteria come tramite collegandosi a una socket creata dalla segreteria.
- **Segreteria**: Questo componente ha due principali funzioni, l'immissione di un nuovo esame fornendo nome e date disponibili al server oppure gestisce le richieste dello studente. Comunica sia con il server universitario connettendosi a una socket creata dal server, sia con lo studente creando una socket a cui lo studente potrà collegarsi.
- **Server Universitario**: Questo componente ha diverse funzioni, avviandosi stampa nel server stesso gli esami salvati, riceve dalla segreteria la richiesta di aggiunta di un esame salvando le informazioni fornite dalla segreteria in un file PKL, ricevendo il nome di uno specifico esame restituisce le date disponibili, riceve le richieste di prenotazione. Comunica con la segreteria creando una socket alla quale la segreteria si collegherà.

Diagramma dell' architettura

Per avere una comprensione visiva dell'architettura il diagramma seguente mostra come i tre componenti sono collegati. Ogni componente è rappresentata in modo distinto mostrando come le informazioni vengono passate tra le componenti.



Capitolo 3

Dettagli Implementativi del Client/Server

3.1 Primo passaggio

Al primo passaggio la segreteria sceglie se aggiungere un esame o rimanere in attesa per una connessione da parte dello studente. Se si vuole aggiungere un esame manderà le informazioni necessarie al server, il server salverà le informazioni e al prossimo avvio lo avrà inserito nella lista degli esami disponibili.

3.2 Secondo passaggio

Al secondo passaggio lo studente manderà una richiesta di connessione, se la segreteria sarà in attesa lo accetterà e lo studente potrà inviare la richiesta delle date disponibili per un esame o la prenotazione a un esame.

Capitolo 4

Parti Rilevanti del Codice Sviluppato

Server Universitario: La parte rilevante del server è la gestione delle richieste da parte della segreteria o degli studenti inoltrati sempre dalla segreteria.

```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}")
```

Segreteria: La parte rilevante della segreteria è la gestione delle richieste da parte dello studente, che inoltrerà al server universitario per poi dopo aver ricevuto le informazioni le ripasserà allo studente

```python
# Funzione per gestire le richieste degli studenti
def gestisci_richieste_studenti(conn, addr):
    global connessioni_attive
    # Incrementa il contatore delle connessioni attive
    connessioni_attive += 1
    # Stampa un messaggio di conferma della connessione
    print(f"Connessione stabilita con {addr}. Connessioni attive: {connessioni_attive}")

    # Invia la conferma della connessione allo studente
    conn.sendall("CONNESSIONE_ACCETTATA".encode(FORMAT))

    try:
        while True:
            # Riceve una richiesta dallo studente
            richiesta = conn.recv(1024).decode(FORMAT)

            if richiesta == "PRENOTAZIONE_ESAME":
                # Gestisce la prenotazione di un esame
                # Riceve i dati della prenotazione dallo studente
                prenotazione = conn.recv(5000)
                # Invia un comando al server per indicare che si sta effettuando una prenotazione
                segreteria_socket.sendall("PRENOTAZIONE_ESAME".encode(FORMAT))
                # Invia i dati della prenotazione al server
                segreteria_socket.sendall(prenotazione)
                # Riceve la risposta dal server
                risposta = segreteria_socket.recv(5000)
                # Invia la risposta allo studente
                conn.sendall(risposta)
                # Stampa un messaggio di conferma
                print(f"Richiesta prenotazione")

            elif richiesta == "DATE_ESAMI":
                # Gestisce la richiesta delle date degli esami
                # Riceve il nome dell'esame dallo studente
                nome_esame = conn.recv(1024).decode(FORMAT)
                # Stampa un messaggio di richiesta delle date degli esami
                print(f"Richiesta date esami per {nome_esame}")
                # Invia un comando al server per richiedere le date degli esami
                segreteria_socket.sendall("DATE_ESAMI".encode(FORMAT))
                # Invia il nome dell'esame al server
                segreteria_socket.sendall(nome_esame.encode(FORMAT))
                # Riceve le date disponibili dal server
                date_disponibili = segreteria_socket.recv(1024)
                # Invia solo le date degli esami allo studente
                conn.sendall(date_disponibili)
            elif richiesta == DISCONNESSIONE:
                # Gestisce la disconnessione del client
                print(f"Disconnessione da {addr}")
                break
    except socket.error as e:
        # Gestisce eventuali errori di connessione e stampa un messaggio di errore
        print(f"Errore di connessione: {e}")
    finally:
        # Chiude la connessione con lo studente
        conn.close()
        # Decrementa il contatore delle connessioni attive
        connessioni_attive -= 1
        # Stampa un messaggio di conferma della chiusura della connessione
        print(f"Connessione chiusa con {addr}. Connessioni attive: {connessioni_attive}")
```

Studente: La parte rilevante dello studente è formata da due funzioni, quella della richiesta degli esami e quella della richiesta di prenotazione. Lo studente dopo essersi collegato alla segreteria potrà utilizzare una delle due funzioni e mandando le informazioni necessarie alla segreteria riceverà le informazioni richieste.

```python
# Funzione per richiedere le date degli esami
def richiesta_esami():
    try:
        # Invia una richiesta per ottenere le date degli esami
        studente_socket.sendall("DATE_ESAMI".encode(FORMAT))

        # Richiede all'utente di inserire il nome dell'esame
        nome_esame = input("Inserisci il nome dell'esame: ")
        # Converte il nome dell'esame in maiuscolo
        nome_esame = nome_esame.upper()

        # Invia il nome dell'esame alla segreteria
        studente_socket.sendall(nome_esame.encode(FORMAT))

        # Riceve le date disponibili dalla segreteria
        date_disponibili = studente_socket.recv(1024)
        date = pickle.loads(date_disponibili)
        if date == "Esame non trovato":
            print("Esame non trovato.")
        else:
            print(f"Le date disponibili per {date.nome_esame} sono: {date.date_disponibili}")
    except socket.error as e:
        print(f"Errore durante la richiesta degli esami: {e}")

# Funzione per prenotare un esame
def prenotazione_esame():
    try:
        # Invia una richiesta per prenotare un esame
        studente_socket.sendall("PRENOTAZIONE_ESAME".encode(FORMAT))

        # Richiede all'utente di inserire il proprio ID studente
        studente_id = input("Inserisci il tuo id studente: ")

        # Richiede all'utente di inserire il nome dell'esame
        nome_esame = input("Inserisci il nome dell'esame: ")
        # Converte il nome dell'esame in maiuscolo
        nome_esame = nome_esame.upper()

        # Richiede all'utente di inserire la data dell'esame
        data_esame = input("Inserisci la data dell'esame (dd-mm-yyyy): ")

        # Verifica che tutti i campi siano stati compilati
        if not studente_id or not nome_esame or not data_esame:
            print("Tutti i campi sono obbligatori.")
            return

        prenotazione = {
            'studente_id': studente_id,
            'nome_esame': nome_esame,
            'data_esame': data_esame
        }
        prenotazione_serializzata = pickle.dumps(prenotazione)
        studente_socket.send(prenotazione_serializzata)

        risposta = studente_socket.recv(5000).decode(FORMAT)
        print(f"Risposta dalla segreteria: {risposta}")
    except socket.error as e:
        print(f"Errore durante la prenotazione dell'esame: {e}")
```

Capitolo 5

Manuale Utente

Compilazione e Esecuzione

5.1 Avviare il terminale

Apri il terminale sul tuo sistema operativo, assicurandoti di avere tutti i privilegi per eseguire i comandi. Sul terminale apri tre finestre cosi da poter avviare i tre file python separati

```
python Server_uni.py
```

```
python Segreteria.py
```

```
python Studente.py
```

5.2 Aggiungere un esame

Nel file segreteria premere il tasto 1 e inserire i dati richiesti. Assicurarsi di riavviare il server per assicurarsi del salvataggio dell esame nel file pkl

5.3 Richiesta esame

Nel file studente premere il tasto 1 e inserire i dati richiesti.

5.4 Prenotazione esame

Nel file studente premere il tasto 2 e inserire i dati richiesti.