Università degli Studi
di Napoli Parthenope

Progetto Reti Di Calcolatori

'Università'

Farinato Marco 0124002621

Docente: Prof.Emanuel Di Nardo

Giugno 2024

Contents

1 Introduction

Il progetto si focalizza sulla progettazione e sviluppo di un'applicazione client/server parallelo per la gestione di esami universitari. L'obiettivo principale è quello di creare un sistema efficiente che faciliti la comunicazione tra la segreteria, gli studenti, il server universitario e il server segreteria, consentendo una gestione ottimale degli esami universitari. Per sviluppare ed implementare il codice ho utilizzato il linguaggio di programmazione Phyton. Nel contesto universitario, la gestione degli esami è cruciale per garantire un flusso ordinato di informazioni tra la segreteria e gli studenti. Il progetto mira a fornire una soluzione pratica ed affidabile, per soddisfare le esigenze sia da parte del client che del server.

2 Obiettivi

Gli obiettivi fondamentali del progetto sono molteplici e comprendono:

- **Terminale per la Segreteria**: Creare un terminale per la segreteria, consentendo un facile inserimento e gestione delle informazioni sugli esami.
- **Verifica e Prenotazione per gli Studenti**: Fornire agli studenti la possibilità di verificare la disponibilità degli esami e di effettuare prenotazioni in modo chiaro e intuitivo.
- **Implementazione di un Server Universitario e un Server Segreteria**: Creare un server universitario e un server segreteria, in grado di gestire con efficienza e affidabilità le richieste provenienti sia dalla segreteria che dagli studenti.

3 Architettura del Sistema

Il cuore del progetto risiede nell'architettura del sistema, concepita con cura per garantire un ambiente fluido e interattivo. Il modello adottato è basato su un'architettura client/server che sfrutta le **socket**, promuovendo la connettività e la collaborazione tra le diverse componenti. In particolare, la

segreteria, gli studenti, il server universitario e il server segreteria sono i principali attori coinvolti, ciascuno con ruoli distinti ma interconnessi.

Lo schema architetturale è basato su quattro entità:

- Il server dell'università;
- Il server della segreteria;
- La segreteria;
- Lo studente;

4 Dettagli implementativi dei client/server

Server Universitario ('Server.py'): Il server universitario è implementato in Python utilizzando il modulo **'socket'** per la comunicazione di rete. Parla "solo" con la segreteria (clientSegreteria.py).

In particolare può ricevere dalla segreteria due richieste:

1. Aggiunta degli esami con date del primo e del secondo appello con il formato 'nome esame' ['primo appello', 'secondo appello'], ad esempio Math ['08.04.2024' , '05.05.2024']. Gli esami inviati dalla segreteria al server, verranno memorizzati dal server università nel file exams.txt
2. Prenotazione degli esami raccolti dalla segreteria. In particolare legge dal file prenotazioni.txt e memorizza le prenotazioni in ConfirmedReservation.txt

Client Segreteria ('ClientSegreteria.py'): Il client segreteria è implementato in Python e comunica le richieste al server università.

Server Segreteria ('ServerSegreteria.py'): Il server universitario è implementato in Python utilizzando il modulo **'socket'** per la comunicazione di rete. Il server segreteria memorizza le prenotazioni da inoltrare al server università nel file prenotazioni.txt e fornisce allo studente le date per un esame scelto.

Client Studente ('StudentClient.py'): Il client studente è implementato in Python. Chiede alla segreteria le date per un certo esame oppure invia alla segreteria la richiesta di prenotazione ad un dato esame.

5 Parti rilevanti del codice sviluppato

- **'Server'**: Contiene il codice per gestire le richieste della segreteria relative agli esami universitari.

```
def save_exam(self,exam_name, exam_date):
    print("in save_exam "+str(exam_name))
    with open('exams.txt', 'a') as file:
        file.write(f'{exam_name} {exam_date}\n')

def save_reservation(self):
    print ('In save_reservation\n')
    with open('confirmedReservation.txt', 'a') as file:
        with open('prenotazioni.txt', 'a+') as temp_f:
            temp_f.seek(0)
            datafile = temp_f.readlines()
            for line in datafile:
                file.write(f'{line}\n')
                print (line)

        temp_f.close()
    file.close()
    with open('prenotazioni.txt', 'a+') as temp_f:
        temp_f.seek(0)
        temp_f.truncate()


def handle_reservation(self, exam_name):
    available_exams = [exam for exam in self.exams if exam['name'] == exam_name]
    if available_exams:
        return available_exams[0]['dates']
    else:
        return "Exam not found."
```

```python
def main():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((host, port))
        s.listen()

        print(f"University server listening on {host}:{port}")

        while True:
            conn, addr = s.accept()
            with conn:
                print('Connected by', addr)
                data = conn.recv(1024)
                if not data:
                    break
                data = pickle.loads(data)
                if data['action'] == 'add_exam':
                    print('request from secretary...')
                    exam=data['exam']
                    exam_name=exam['name']
                    exam_date=exam['dates']
                    server.save_exam(exam_name,exam_date)
                    conn.send("Exam added successfully.".encode('utf-8'))
                    print('sent respons to secretary for exam added...')
                elif data['action'] == 'reserve_exam':
                    print('request from secretary to reserve all student request...')
                    server.save_reservation()
                    conn.send("Reservation done...".encode('utf-8'))
                    print('sent respons to secretary for all reservation...')
```

- **'ServerSegreteria'**: Gestisce le richieste degli studenti per prenotare esami e restituisce le date disponibili per un dato esame.

```python
def save_reservation(self, exam_name, exam_date, student_name, student_surname, student_matricola):
    print("in reservation_secretary "+str(student_name))
    with open('prenotazioni.txt', 'a') as file:
        file.write(f'{exam_name} {exam_date} {student_name} {student_surname} {student_matricola}\n')

def handle_reservation(self, exam_name):
    with open('exams.txt') as temp_f:
      datafile = temp_f.readlines()
    available_exams = '00.00.0000'
    for line in datafile:
      #print("leggo "+line)
      if exam_name in line:
        available_exams = line  # The string is found
        print (available_exams)
        break

    if available_exams:
        return available_exams
    else:
        return "Exam not found."
```

- **'ClientSegreteria'**: Fornisce un'interfaccia per la segreteria per aggiungere esami e prenotare esami per gli studenti.

```python
def add_exam(self, exam):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((self.host, self.port))
        data_exams = {'action': 'add_exam', 'exam': exam}
        s.sendall(pickle.dumps(data_exams))
        response = s.recv(1024)
        print(response)

def reserve_all_exam(self):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((self.host, self.port))
        data = {'action': 'reserve_exam'}
        s.sendall(pickle.dumps(data))
        response = s.recv(1024).decode("utf-8", "ignore")
        print(response)
```

```python
def main():


   secretaryclient = SecretaryClient('127.0.0.1', 12345)
   scelta=-1
   while(scelta != 0):
       print("******Secretary section******")
       print("      1. Add exam")
       print("      2. reserve student exams")
       print("      3. quit")
       scelta = int(input("Insert number of your choice: "))
       if scelta == 1:
       # Esempio di utilizzo:
           esame_da_inserire = str(input("Digit exam to insert: "))
           data1=str(input("First date  "))
           data2=str(input("Second date "))
           #exam = {'name': 'Math', 'dates': ['2024-03-10', '2024-03-15']}
           exam = {'name': esame_da_inserire, 'dates': [data1, data2]}
           secretaryclient.add_exam(exam)
       elif scelta == 2:
           #exam_name = str(input("Digit exam to reserve: "))
           #exam_name = 'Math'
           secretaryclient.reserve_all_exam()
       elif scelta == 3:
           print("*** Bye....")
           break
       else:
           print("Not valid choice. Insert 1 or 2 or 3.")
```

- **'StudentClient'**: Permette agli studenti di consultare le date degli esami e di prenotare esami.

```python
def inquire_exam_dates(self, exam_name):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((self.host, self.port))
        data = {'action': 'inquire_exam_dates', 'exam_name': exam_name}
        s.sendall(pickle.dumps(data))
        response = s.recv(1024)
        print(response)

def reserve_exam(self, reservation):
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((self.host, self.port))
        data_reservation = {'action': 'save_reservation', 'reservation': reservation}
        s.sendall(pickle.dumps(data_reservation))
        response = s.recv(1024)
        print(response)
```

```python
def main():
    student = StudentClient('127.0.0.1', 54321)

    scelta=-1
    while(scelta != 0):
        print("""
        1.Inquire exam date
        2.Reserve exam
        3.Exit/Quit
        """)
        scelta = int(input("Insert number of your choice: "))

        if scelta==1:
            exam_name=str(input("Exam to search: "))
            available_dates = student.inquire_exam_dates(exam_name)

        elif scelta==2:
            print("\n Reserve exam")
            exam_name=str(input("Exam Name: "))
            date=str(input("Exam Date: "))
            name=str(input("Name: "))
            surname=str(input("Surname: "))
            matricola=str(input("Matricola: "))
            reservation = {'exam': exam_name, 'date': date, 'studentN': name, 'studentS': surname, 'Matricola': matricola}
            student.reserve_exam(reservation)
        elif scelta==3:
            print("\n Goodbye")
            break
        else:
            print("\n Not Valid Choice Try again")
```

6 Manuale Utente

Compilazione ed Esecuzione:

- Assicurarsi di avere Python installato sul proprio sistema;
- Salvare i file del client e del server con estensione '.py';
- Aprire un terminale o prompt dei comandi;
- Per eseguire il server universitario (server.py), aprire l'apposito terminale ed eseguire il comando python3 server.py;
- Per eseguire il server segreteria (ServerSegreteria.py), aprire l'apposito terminale ed eseguire il comando python3 ServerSegreteria.py;

- Per eseguire il client segreteria (ClientSegreteria.py), aprire l'apposito terminale ed eseguire il comando python3 ClientSegreteria.py;
- Per eseguire il client studente (StudentClient.py). aprire l'apposito terminale ed eseguire il comando python3 StudentClient.py;

Iterazione con l'Applicazione:

- - Dopo aver avviato il server universitario, il server segreteria e i client segreteria e studente, seguire le istruzioni stampate nel terminale.
- Utilizzare i client segreteria/studente per, rispettivamente, aggiungere esami o prenotare date di esami.

Con queste istruzioni, gli utenti saranno in grado di compilare ed eseguire con successo il sistema client/server per la gestione degli esami universitari.