Università degli Studi di Napoli "Parthenope"
Facoltà di Scienze e Tecnologie
Corso di laurea in Informatica

Università degli Studi
di Napoli Parthenope

Corso di Reti di Calcolatori

Traccia - Università

Docente
Prof. Emanuel Di Nardo

Candidato
Tullio Ciricillo 0124002530

Anno Accademico 2023-2024

Descrizione del Progetto

Il progetto è composto da tre classi principali : il `ServerUniversitario`, la `Segreteria` e lo `Studente`. Il `ServerUniversitario` gestisce la connessione con il database `MySQL` e fornisce metodi per l'inserimento di nuovi esami e il recupero di informazioni su esami inseriti. La `Segreteria` funge da intermediario tra lo `Studente` e il `ServerUniversitario` inoltre può aggiungere nuovi esami, mentre lo `Studente` interagisce direttamente con la `Segreteria` per richiedere le date di un esame e prenotare esami.

Tecnologie

Le tecnologie usate per lo sviluppo di questa applicazione sono : protocollo TCP/IP per la comunicazione che avviene tramite Socket e ServerScoket nel linguaggio Java e connessione ad un server locale MySQL.

Architettura del Sistema

L'architettura del sistema è basata su una comunicazione client-server. Il `ServerUniversitario` è in ascolto su una porta specifica e gestisce le connessioni della `Segreteria`. La `Segreteria` a sua volta è in ascolto su una porta diversa e si collega allo `Studente` oltre a connettersi alla socket creata dal `ServerUniversitario`. Le comunicazioni avvengono tramite socket. Di seguito viene illustrata l'architettura del sistema.

(ServerUniversitario) ¡–[Connessione: 12345]– (Segreteria) –[Connessione: 12346]–¿ (Studente)

- ServerUniversitario accetta connessioni sulla porta 12345.
- Segreteria si connette a ServerUniversitario sulla porta 12345 e accetta connessioni sulla porta 12346.
- Studente si connette a Segreteria sulla porta 12346

Implementazione Client/Server

Il `ServerUniversitario` utilizza un loop infinito per accettare connessioni dalla `Segreteria`. Una volta stabilita la connessione, gestisce le richieste inviate dalla `Segreteria`. La `Segreteria` ha due thread distinti uno per l'inserimento di nuovi esami e l'altro per l'inoltro delle richieste degli studenti al server universitario. Lo `Studente` si collega alla `Segreteria` per richiedere informazioni e prenotare esami.

Connessioni delle socket tra le classi

1. ServerUniversitario:
   (a) Apre un ServerSocket in ascolto sulla porta 12345.
   (b) Accetta connessioni in entrata dai client tramite serverSocket.accept().
   (c) Per ogni connessione accettata, crea un socket (segreteriaSocket) dedicato per comunicare con la Segreteria.
2. Segreteria:
   (a) Crea un socket (universitarioSocket) per connettersi al ServerUniversitario sulla porta 12345.
   (b) Apre un ServerSocket in ascolto sulla porta 12346 per accettare connessioni dagli Studenti.
   (c) Per ogni connessione accettata da uno `Studente`, crea un socket dedicato (studenteSocket) per comunicare con lo Studente.
   (d) Invia messaggi al ServerUniversitario attraverso universitarioSocket.
   (e) Riceve messaggi dagli Studenti attraverso i rispettivi socket.
3. Studente:
   (a) Crea un socket (serverSegreteriaSocket) per connettersi alla Segreteria sulla porta 12346.
   (b) Invia messaggi alla Segreteria attraverso segreteriaWriter.
   (c) Riceve messaggi dalla Segreteria attraverso segreteriaReader

Scambio di Messaggi

Lo scambio di messaggi avviene principalmente attraverso i flussi di input/output (InputStream/OutputStream) dei socket tra le varie componenti del sistema: ServerUniversitario, Segreteria e Studente. In ogni componente, l'uso di *println* e *readLine* sugli oggetti *PrintWriter* e *BufferedReader* consente di inviare e ricevere interi messaggi di testo tra le varie parti del sistema.

```
// Nel codice del ServerUniversitario
BufferedReader segreteriaReader = new BufferedReader(new InputStreamReader(
    segreteriaSocket.getInputStream()));
PrintWriter segreteriaWriter = new PrintWriter(segreteriaSocket.getOutputStream(),
    true);

// Nel codice della Segreteria
BufferedReader serverUniversitarioReader = new BufferedReader(new InputStreamReader(
    universitarioSocket.getInputStream()));
PrintWriter universitarioWriter = new PrintWriter(universitarioSocket.getOutputStream
    (), true);

// Nel codice dello Studente
BufferedReader segreteriaReader = new BufferedReader(new InputStreamReader(
    serverSegreteriaSocket.getInputStream()));
PrintWriter segreteriaWriter = new PrintWriter(serverSegreteriaSocket.getOutputStream
    (), true);
```

Gestione delle Richieste

Server Universitario

Il Server Universitario è responsabile della gestione del database degli esami e delle prenotazioni. Utilizza delle serverSocket per accettare connessioni dalla Segreteria. Gestisce le richieste della Segreteria e interagisce con il database MySQL.

```
// Gestisci la richiesta di date per un esame
case "RICHIEDI_DATE_ESAME":
    String[] infoDateEsame = recuperaEsamePerNomeEsame(conn, argomenti);
    if (infoDateEsame != null) {
        String idEsame = infoDateEsame[0];
        String nomeEsame = infoDateEsame[1];
        String dataEsame = infoDateEsame[2];
        segreteriaWriter.println("ID_ESAME:" + idEsame + ",NOME_ESAME:" + nomeEsame
                + ",DATE_ESAME:" + dataEsame);
    } else {
        segreteriaWriter.println("Esame non trovato");
    }
    break;

// Gestisci la richiesta di prenotazione di un esame
case "PRENOTA_ESAME":
    String[] argomentiPrenotazione = argomenti.split(",");
    if (argomentiPrenotazione.length == 2) {
        String idEsamePrenotazione = argomentiPrenotazione[0].trim();
        String matricolaStudente = argomentiPrenotazione[1].trim();
        String rispostaPrenotazione = prenotaEsame(conn, idEsamePrenotazione,
                matricolaStudente);
        segreteriaWriter.println(rispostaPrenotazione);
    } else {
        segreteriaWriter.println("Comando non valido");
    }
    break;

// Gestisci la richiesta di aggiunta di un nuovo esame
case "NUOVO_ESAME":
    String[] argomentiNuovoEsame = argomenti.split(",");
    if (argomentiNuovoEsame.length == 2) {
        String nomeNuovoEsame = argomentiNuovoEsame[0].trim();
        String dataNuovoEsame = argomentiNuovoEsame[1].trim();

        // Esegui l'operazione di aggiunta del nuovo esame
        String rispostaAggiuntaEsame = aggiungiNuovoEsame(conn, nomeNuovoEsame,
                dataNuovoEsame);
        segreteriaWriter.println(rispostaAggiuntaEsame);
    } else {
        segreteriaWriter.println("Comando NUOVO_ESAME non valido");
    }
    break;
```

Segreteria

La Segreteria si connette al Server Universitario e inoltra le richieste degli studenti. Implementa un menu da terminale per l'inserimento di nuovi esami. Utilizza thread per gestire contemporaneamente l'inserimento di nuovi esami e l'inoltro delle richieste degli studenti.

Nel seguente codice vediamo la logica della Segreteria quando deve inoltrare le richieste dallo Studente al Server Universitario.

```
private static void inoltraRichieste(Socket universitarioSocket, ServerSocket
    studenteServerSocket) {
    try {
        // Accettazione della connessione dallo Studente
        Socket studenteSocket = studenteServerSocket.accept();
        System.out.println(
                "\n(*) Connessione accettata da 'Studente' su indirizzo : "
                        + studenteSocket.getInetAddress().getHostAddress());

        try (
                BufferedReader studenteReader = new BufferedReader(
                        new InputStreamReader(studenteSocket.getInputStream()));
                PrintWriter studenteWriter = new PrintWriter(studenteSocket.
    getOutputStream(), true);
                BufferedReader serverUniversitarioReader = new BufferedReader(
                        new InputStreamReader(universitarioSocket.getInputStream()));
                PrintWriter universitarioWriter = new PrintWriter(universitarioSocket
    .getOutputStream(), true)) {
            while (true) {
                // Leggi la richiesta dello studente
                String richiestaStudente = studenteReader.readLine();
                System.out.println("Richiesta dallo Studente: " + richiestaStudente);

                // Inoltra la richiesta al ServerUniversitario
                universitarioWriter.println(richiestaStudente);

                // Leggi la risposta dal ServerUniversitario e inoltrala allo
    Studente
                String rispostaServer = serverUniversitarioReader.readLine();
                System.out.println("Risposta dal ServerUniversitario: " +
    rispostaServer);
                studenteWriter.println(rispostaServer);
            }
...
```

Nel seguente codice invece viene mostrata la logica della Segreteria per aggiungere un nuovo esame. Viene inviato al Server Universitario un messaggio formattato che inizia per NUOVO_ESAME così il server sa quale richiesta deve soddisfare. Il messaggio sarà così formattato : NUOVO_ESAME:nomeEsame,dataEsame

```
private static void inserisciNuoviEsami(Socket universitarioSocket) {
    try (BufferedReader serverUniversitarioReader = new BufferedReader(
            new InputStreamReader(universitarioSocket.getInputStream()));
            PrintWriter universitarioWriter = new PrintWriter(universitarioSocket.
    getOutputStream(), true);
            Scanner scanner = new Scanner(System.in)) {

        while (true) {
            // Inserimento del nuovo esame
            System.out.print("Inserisci il nome del nuovo esame: ");
            String nomeNuovoEsame = scanner.nextLine();
            System.out.print("Inserisci la data del nuovo esame (yyyy-MM-dd): ");
            String dataNuovoEsame = scanner.nextLine();

            // Invia la richiesta di aggiunta del nuovo esame al ServerUniversitario
            universitarioWriter.println("NUOVO_ESAME:" + nomeNuovoEsame + "," +
    dataNuovoEsame);

            // Leggi la risposta dal ServerUniversitario
            String rispostaServer = serverUniversitarioReader.readLine();
            System.out.println("Risposta dal ServerUniversitario: " + rispostaServer)
    ;
            break;
...
```

Studente

Lo Studente si connette alla Segreteria e interagisce con un menu da terminale. Può richiedere date disponibili per esami o prenotare un esame specifico. A seconda della scelta effettuata verrà passato alla Segreteria un messaggio formattato.

1. Se la scelta è RICHIEDI_DATE_ESAME il messaggio sarà formattato così : RICHIEDI_DATE_ESAME:nomeEsame
2. Se la scelta è PRENOTA_ESAME il messaggio sarà formattato così : PRENOTA_ESAME:idEsame,matricolaStudente

```
case "1":
// Richiedi date disponibili per un esame scelto
System.out.print("--> Inserisci il nome dell'esame: ");
String nomeEsameRichiestaDate = userInputReader.readLine();

// Invia la richiesta al server della Segreteria
segreteriaWriter.println("RICHIEDI_DATE_ESAME:" + nomeEsameRichiestaDate);

// Leggi la risposta dal server della Segreteria e stampa le date disponibili
String rispostaDate = segreteriaReader.readLine();
System.out.println();
// System.out.println("\nDate disponibili per l'esame di " +
// nomeEsameRichiestaDate + ":");

// Splitta la stringa usando la virgola come delimitatore
String[] informazioni = rispostaDate.split(",");

// Stampa ogni informazione su una nuova riga
for (String informazione : informazioni) {
    System.out.println(informazione);
}
break;

case "2":
// Prenota un esame
System.out.print("--> Inserisci l'ID dell'esame: ");
String idEsamePrenotazione = userInputReader.readLine();

System.out.print("--> Inserisci la tua matricola: ");
String matricolaStudente = userInputReader.readLine();

// Invia la richiesta al server della Segreteria
segreteriaWriter.println("PRENOTA_ESAME:" + idEsamePrenotazione + "," +
    matricolaStudente);

// Leggi la risposta dal server della Segreteria e stampa il risultato della
// prenotazione
String rispostaPrenotazione = segreteriaReader.readLine();
System.out.println(rispostaPrenotazione);
break;
```

Esempio di Esecuzione

È evidente come il Server Universitario sia in attesa di stabilire una connessione con la Segreteria e comunica immediatamente il successo di tale connessione non appena viene stabilita. Inoltre, è possibile esaminare da vicino i messaggi inviati dal Server Universitario quando riceve comunicazioni dalla Segreteria.

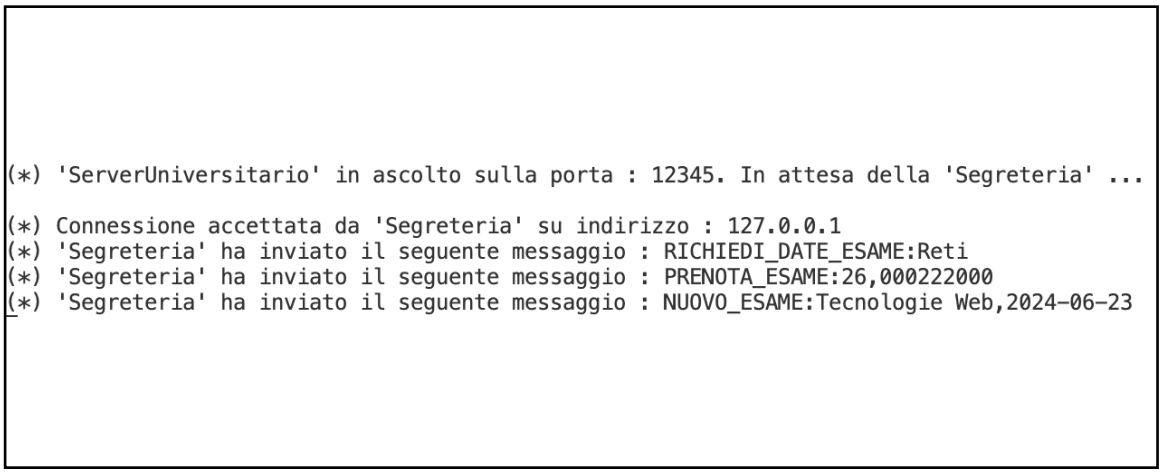

Figure 1: Terminale Server Universitario

Nel seguente contesto, è evidente come la Segreteria comunica al Server Universitario il messaggio di connessione e la sua disponibilità ad ascoltare connessioni in arrivo da parte degli Studenti. Quando uno Studente stabilisce una connessione, viene visualizzato il relativo messaggio. A questo punto, la Segreteria si prepara ad accogliere e inoltrare i messaggi provenienti dallo Studente al Server Universitario.

Il flusso di messaggi tra lo Studente, la Segreteria e il Server Universitario diventa chiaramente evidente, specialmente per quanto riguarda le richieste di informazioni sulle date degli esami e le prenotazioni. Inoltre, è rilevante notare la capacità della Segreteria di agire come client nei confronti del Server Universitario, ad esempio, quando effettua una richiesta di aggiunta di un nuovo esame.

Nella presente illustrazione, è evidente il messaggio di connessione tra lo Studente e la Segreteria. Vengono successivamente eseguite le operazioni di richiesta di una data per un esame e di prenotazione dello stesso, operazioni che vengono inoltrate dalla Segreteria al Server Universitario. La figura mostra chiaramente anche le risposte del server, evidenziate dai messaggi di successo.

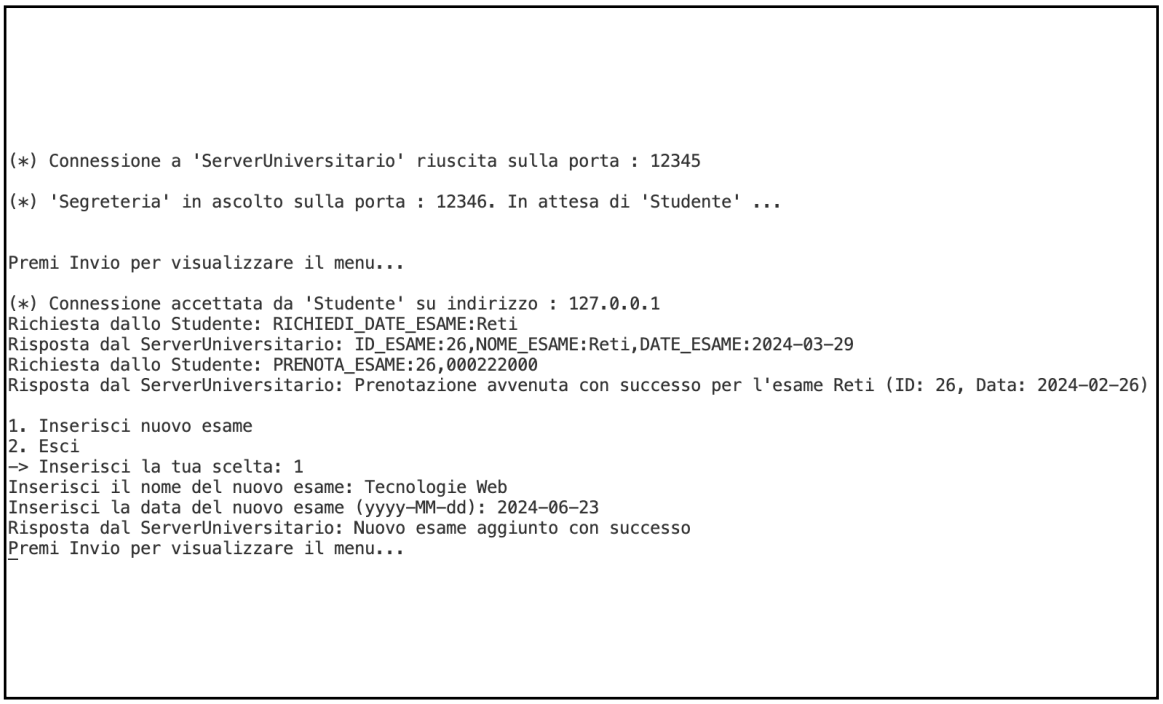

Figure 2: Terminale Segreteria

```
(*) Connessione a 'Segreteria' riuscita sulla porta : 12346


Menu:
1. Richiedi date disponibili
2. Prenota un esame
3. Esci
-> Inserisci la tua scelta: 1

--> Inserisci il nome dell'esame: Reti

ID_ESAME:26
NOME_ESAME:Reti
DATE_ESAME:2024-03-29


Menu:
1. Richiedi date disponibili
2. Prenota un esame
3. Esci
-> Inserisci la tua scelta: 2

--> Inserisci l'ID dell'esame: 26
--> Inserisci la tua matricola: 000222000
Prenotazione avvenuta con successo per l'esame Reti (ID: 26, Data: 2024-02-26)
```

Figure 3: Terminale Studente