

UNIVERSITÁ DEGLI STUDI DI NAPOLI “PARTHENOPE”  
FACOLTÀ DI SCIENZE E TECNOLOGIE  
CORSO DI LAUREA IN INFORMATICA (PERCORSO GENERALE)



PROGETTO ESAME RETI DI CALCOLATORI

Università

DOCENTE  
Emanuel Di Nardo

STUDENTE  
Francesco Picone  
Matr.: 0124/1779

Anno Accademico 2023-2024

# Indice

<b>1</b>	<b>Descrizione del progetto</b>	<b>1</b>
1.1	Traccia . . . . .	1
1.2	Note di sviluppo . . . . .	2
<b>2</b>	<b>Descrizione e schema dell'architettura</b>	<b>3</b>
2.1	Schema Client/Server . . . . .	3
2.2	Client Studente . . . . .	4
2.2.1	Strutture dati utilizzate . . . . .	5
2.3	Client/Server Segreteria . . . . .	6
2.3.1	Strutture dati utilizzate . . . . .	7
2.4	Server universitario . . . . .	8
2.4.1	Strutture dati utilizzate . . . . .	10
2.5	Protocolli di comunicazione . . . . .	12
2.5.1	Comunicazione Segreteria/Server Universitario . . . . .	12
2.5.2	Comunicazione Client studente/Server segreteria . . . . .	16
<b>3</b>	<b>Screenshots</b>	<b>17</b>
3.1	Client studente . . . . .	17
3.2	Client/Server Segreteria . . . . .	19
3.3	Server Universitario . . . . .	21
<b>4</b>	<b>Manuale utente</b>	<b>22</b>
4.1	Struttura dei sorgenti . . . . .	22
4.1.1	Download dei sorgenti . . . . .	22
4.2	Istruzioni per la compilazione . . . . .	23
4.3	Istruzioni per l'esecuzione . . . . .	23

# Capitolo 1

## Descrizione del progetto

### 1.1 Traccia

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

*Segreteria:*

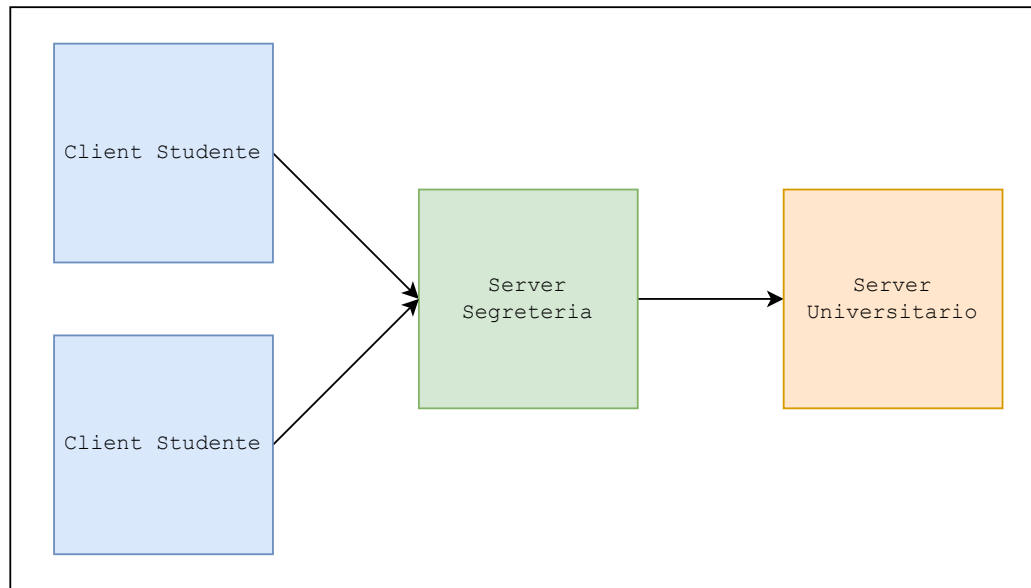
- Inserisce gli esami sul server dell'università (salvare in un file o conservare in memoria il dato)
- Inoltra la richiesta di prenotazione degli studenti al server universitario
- Fornisce allo studente le date degli esami disponibili per l'esame scelto dallo studente

*Studente:*

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

*Server universitario:*

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



## 1.2 Note di sviluppo

Il progetto deve essere sviluppato secondo le seguenti linee:

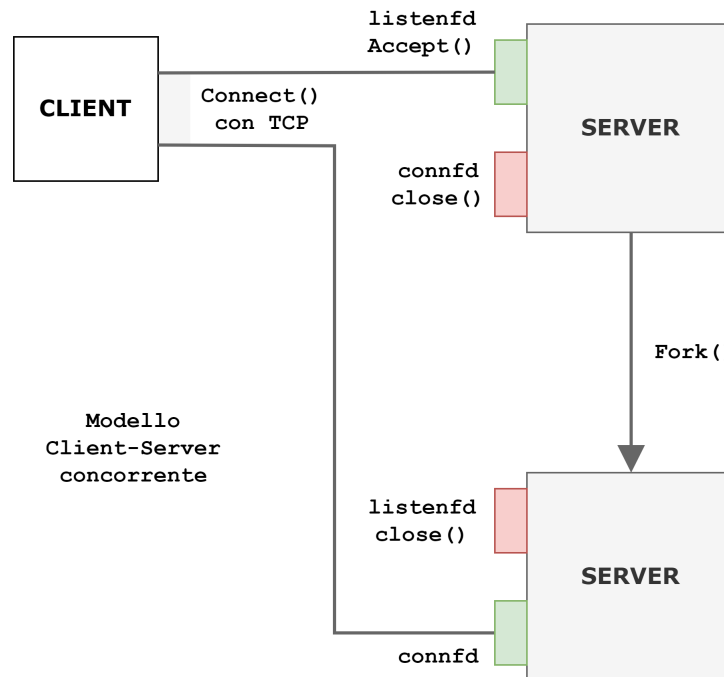
- utilizzare un linguaggio di programmazione a scelta (C, Java, Python, etc...)
- utilizzare una piattaforma **Unix-like**;
- utilizzare le **socket**;
- inserire sufficienti **commenti**;

# Capitolo 2

## Descrizione e schema dell'architettura

### 2.1 Schema Client/Server

Nel progetto implementato é stato adottato un modello di programmazione Client-Server, che é un'architettura comune nella progettazione di applicazioni distribuite. Nell'ambito di questo modello, un server é responsabile di fornire servizi a piú client contemporaneamente, quindi, il server é stato implementato come un processo concorrente. Un aspetto chiave di questa implementazione é l'uso della **system call fork()**. Quando un client richiede una connessione al server, il server chiama `fork()` per creare un nuovo processo figlio. Questo processo figlio é un duplicato esatto del processo padre e puó operare in modo indipendente. Il numero di processi figli creati corrisponde al numero di connessioni dei client che il server é in grado di gestire simultaneamente. In altre parole, ogni volta che un nuovo client si connette, viene creato un nuovo processo figlio per gestire quella specifica connessione. Il processo padre é responsabile di gestire un descrittore chiamato `listenfd`, esso é utilizzato per accettare le richieste di connessione dai client. Quando un client richiede una connessione, il processo padre la accetta tramite `listenfd` ed eseguirá la funzione `connect()` la quale crea un nuovo descrittore chiamato `connfd` che rappresenta la connessione effettiva con il client, esso verrá gestito dal figlio per offrire servizi al client connesso.



## 2.2 Client Studente

Il client studente permette di cercare un appello per un determinato corso ed effettuare la prenotazione. Esso si collega al server della segreteria, chiede allo studente di inserire la propria matricola e la invia per autenticarsi. Riceve un acknowledge dal server della segreteria:

### Se l'autenticazione riesce:

riceve la lista di tutti i corsi con appelli disponibili (riceve una struttura dati di tipo CORSO), ricevuta questa lista permette allo studente di scegliere un corso (attraverso l'inserimento del nome) e prenotarsi per uno specifico appello di quel corso (attraverso l'inserimento dell'ID dell'appello). Riceve infine dal server della segreteria un acknowledge dell'avvenuta prenotazione.

### Se l'autenticazione fallisce:

Riceve dal server della segreteria un acknowledge di autenticazione fallita, stampa

il messaggio, chiude la connessione ed esce.

La prenotazione fallisce se:

- Lo studente é già prenotato per quello specifico appello
- Errore da parte del server segreteria / universitario

### 2.2.1 Strutture dati utilizzate

CORSO		
<b>ID</b> [INT]	<b>Nome</b> [CHAR]	<b>Crediti</b> [INT]

**CORSO** é una struttura dati che contiene le informazioni di un determinato corso. É formata da:

- **ID** del corso
- **NOME** del corso
- **CREDITI** del corso

ESAME		
<b>ID</b> [INT]	<b>Corso</b> [CORSO]	<b>Data</b> [DATE]

**ESAME** é una struttura dati che contiene le informazioni di un determinato appello. É formata da:

- **ID** dell'appello
- **CORSO**, struttura che contiene le info sul corso al quale fa riferimento l'appello
- **Data** dell'appello

DATE		
<b>Day</b> [INT]	<b>Month</b> [INT]	<b>Year</b> [INT]

**DATE** é una struttura dati utilizzata per memorizzare una data

- **Day**, memorizza il giorno
- **Month**, memorizza il mese
- **Year**, memorizza l'anno

## 2.3 Client/Server Segreteria

L'applicazione segreteria funge sia da client che da server. Quando si avvia l'applicazione sono generati 2 processi:

- **Processo server**, accetta connessioni dal client studente al quale fornisce le date degli appelli disponibili per il corso scelto e inoltra la richiesta di prenotazione per un appello al server universitario.

Quando lo studente si connette riceve un messaggio di benvenuto dal server della segreteria ed effettua la login attraverso la propria matricola. Il Server della segreteria una volta ricevuta la matricola controlla se lo studente esiste nel file *"studenti.txt"*.

studenti.txt		
Matricola	Nome	Cognome
⋮	⋮	⋮

### Se l'utente esiste:

invia un acknowledge di avvenuta autenticazione, richiede la lista delle materie con appelli disponibili al server universitario e la inoltra al client studente.

Successivamente riceve dal client studente una richiesta contenente il nome del corso per il quale vuole conoscerne le date degli appelli. Il server della segreteria elabora questa richiesta richiedendo la lista degli appelli al server universitario ed inoltrandola al client studente.

Successivamente se lo studente decide di prenotarsi, il server riceve dal client studente una richiesta con l'ID dell'appello. Elabora poi questa richiesta inoltrando l'ID dell'appello e la matricola dello studente al server universitario il quale provvederà a memorizzare la prenotazione.

Infine il server della segreteria riceve un acknowledge di avvenuta prenotazione o di prenotazione fallita dal server universitario il quale sarà inoltrato al client studente.



**N.B:** Il server della segreteria prima di ogni richiesta al server universitario invia un byte iniziale per indentificare il tipo di richiesta. Ciò é specificato nel paragrafo 2.4.

#### Se l'utente non esiste:

Il server della segreteria invia un acknowledge di autenticazione fallita e chiude la connessione.

Questo processo crea ulteriori processi per rispondere a piú richieste contemporaneamente.

- **Processo client**, permette di inserire nuovi appelli di un determinato corso sul server universitario. Una volta che sono stati inseriti tutti i dati dell'appello (Corso, CFU e data) viene generata una struttura dati di tipo ESAME la quale é inviata al server universitario che provvederá ad inserire il nuovo appello.

Infine riceve un acknowledge di avvenuto inserimento.

### 2.3.1 Strutture dati utilizzate

CORSO		
<b>ID</b> [INT]	<b>Nome</b> [CHAR]	<b>Crediti</b> [INT]

**CORSO** é una struttura dati che contiene le informazioni di un determinato corso. É formata da:

- **ID** del corso
- **NOME** del corso
- **CREDITI** del corso

ESAME		
<b>ID</b> [INT]	<b>Corso</b> [CORSO]	<b>Data</b> [DATE]

**ESAME** é una struttura dati che contiene le informazioni di un determinato appello. É formata da:

- **ID** dell'appello
- **CORSO**, struttura che contiene le info sul corso al quale fa riferimento l'appello
- **Data** dell'appello (struttura dati DATE)

STUDENTE		
<b>Matricola</b> [CHAR]	<b>Nome</b> [CHAR]	<b>Cognome</b> [CHAR]

**STUDENTE** é una struttura dati che contiene le informazioni di un determinato appello. É formata da:

- **Matricola** dello studente
- **Nome** dello studente
- **Cognome** dello studente

DATE		
<b>Day</b> [INT]	<b>Month</b> [INT]	<b>Year</b> [INT]

**DATE** é una struttura dati utilizzata per memorizzare una data

- **Day**, memorizza il giorno
- **Month**, memorizza il mese
- **Year**, memorizza l'anno

## 2.4 Server universitario

Il server universitario accetta connessioni dal client della segreteria. Esso :

- Riceve l'aggiunta di un nuovo appello per un determinato corso (struttura dati di tipo ESAME)
- Riceve la prenotazione di un'appello da parte di uno studente

- Invia la lista dei corsi con appelli disponibili al client della segreteria (struttura dati di tipo CORSO)
- Invia la lista degli appelli per un determinato corso al client della segreteria (struttura dati di tipo ESAME)

Il server universitario riceve dal client della segreteria un *byte iniziale* che identifica il tipo di richiesta:

- ***byte iniziale* = 1**, il client della segreteria vuole inviare l'aggiunta di un nuovo appello. Il server universitario riceve un pacchetto contenente una struttura dati di tipo ESAME la quale contiene tutte le informazioni relative all'appello. L'appello viene aggiunto al file "*esami.txt*" come una nuova riga.

esami.txt			
ID	Nome Corso	Crediti	Data appello
ID	Nome Corso	Crediti	Data appello
⋮	⋮	⋮	⋮

Infine il server universitario invia un messaggio di acknowledge al client della segreteria per confermare l'aggiunta.

- ***byte iniziale* = 2**, il client della segreteria sta richiedendo la lista dei corsi per il quale sono disponibili appelli. Il server universitario prepara una struttura dati di tipo CORSO contenente questi dati e la invia al client della segreteria il quale poi la inoltrerá al client studente.
- ***byte iniziale* = 3**, il client della segreteria sta richiedendo la lista degli appelli per un determinato corso. Il server universitario riceve quindi il nome del corso, prepara una struttura dati di tipo ESAME contenente tutti gli appelli e relative info del corso e la invia al client della segreteria il quale la inoltrerá al client dello studente.
- ***byte iniziale* = 4**, il client della segreteria vuole inoltrare la prenotazione di un'appello da parte di uno studente. Il server universitario riceve l'ID

del corso per il quale deve memorizzare la prenotazione e la matricola dello studente.

Controlla nel file *"prenotazioni.txt"* se lo studente é già prenotato per quel determinato appello, in caso negativo salva la prenotazione nel file *"prenotazioni.txt"* aggiungendo una nuova riga con *matricola* e *ID* dell'appello altrimenti invia un acknowledge di prenotazione fallita (= 0).

prenotazioni.txt	
Matricola	ID appello
Matricola	ID appello
⋮	⋮

### 2.4.1 Strutture dati utilizzate

CORSO		
ID [INT]	Nome [CHAR]	Crediti [INT]

**CORSO** é una struttura dati che contiene le informazioni di un determinato corso. É formata da:

- **ID** del corso
- **NOME** del corso
- **CREDITI** del corso

ESAME		
<b>ID</b> [INT]	<b>Corso</b> [CORSO]	<b>Data</b> [DATE]

**ESAME** é una struttura dati che contiene le informazioni di un determinato appello. É formata da:

- **ID** dell'appello
- **CORSO**, struttura che contiene le info sul corso al quale fa riferimento l'appello
- **Data** dell'appello (struttura dati DATE)

DATE		
<b>Day</b> [INT]	<b>Month</b> [INT]	<b>Year</b> [INT]

**DATE** é una struttura dati utilizzata per memorizzare una data

- **Day**, memorizza il giorno
- **Month**, memorizza il mese
- **Year**, memorizza l'anno

## 2.5 Protocolli di comunicazione

### 2.5.1 Comunicazione Segreteria/Server Universitario

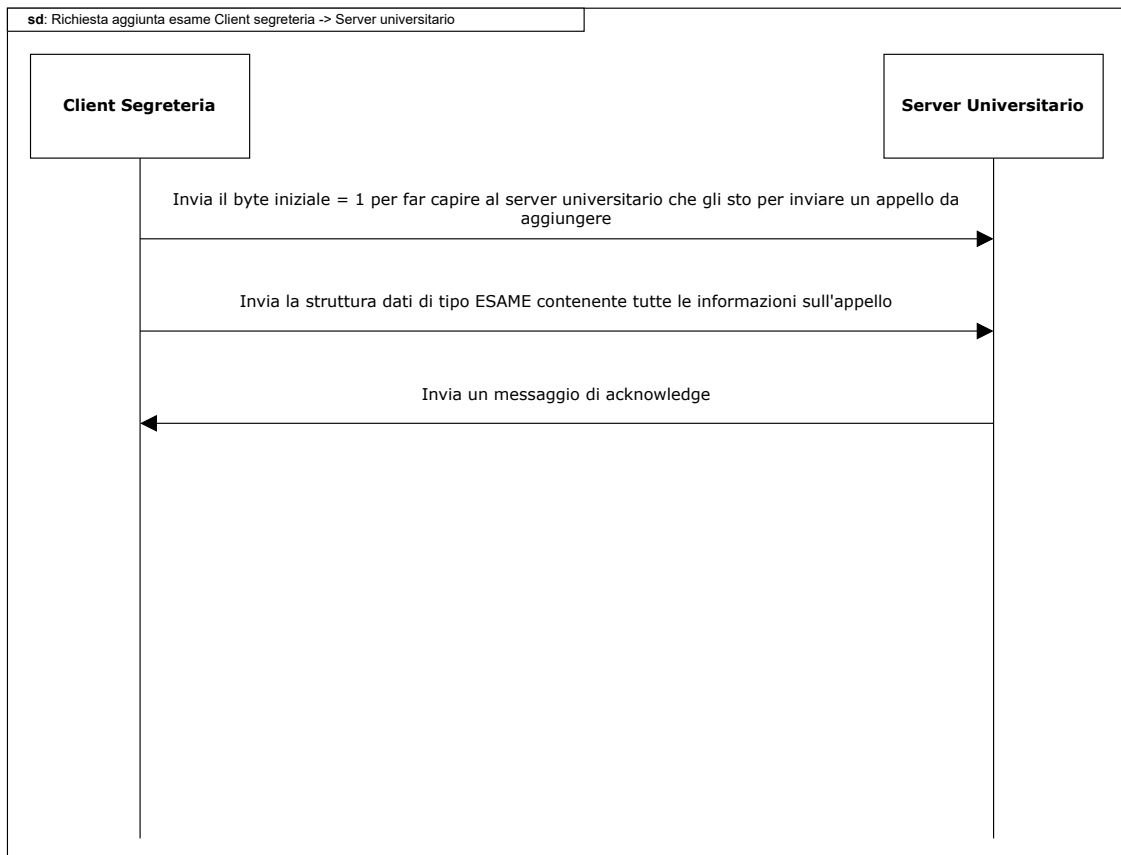


Figura 2.1: Richiesta di inserimento di un nuovo esame effettuata dal client della segreteria al server universitario.

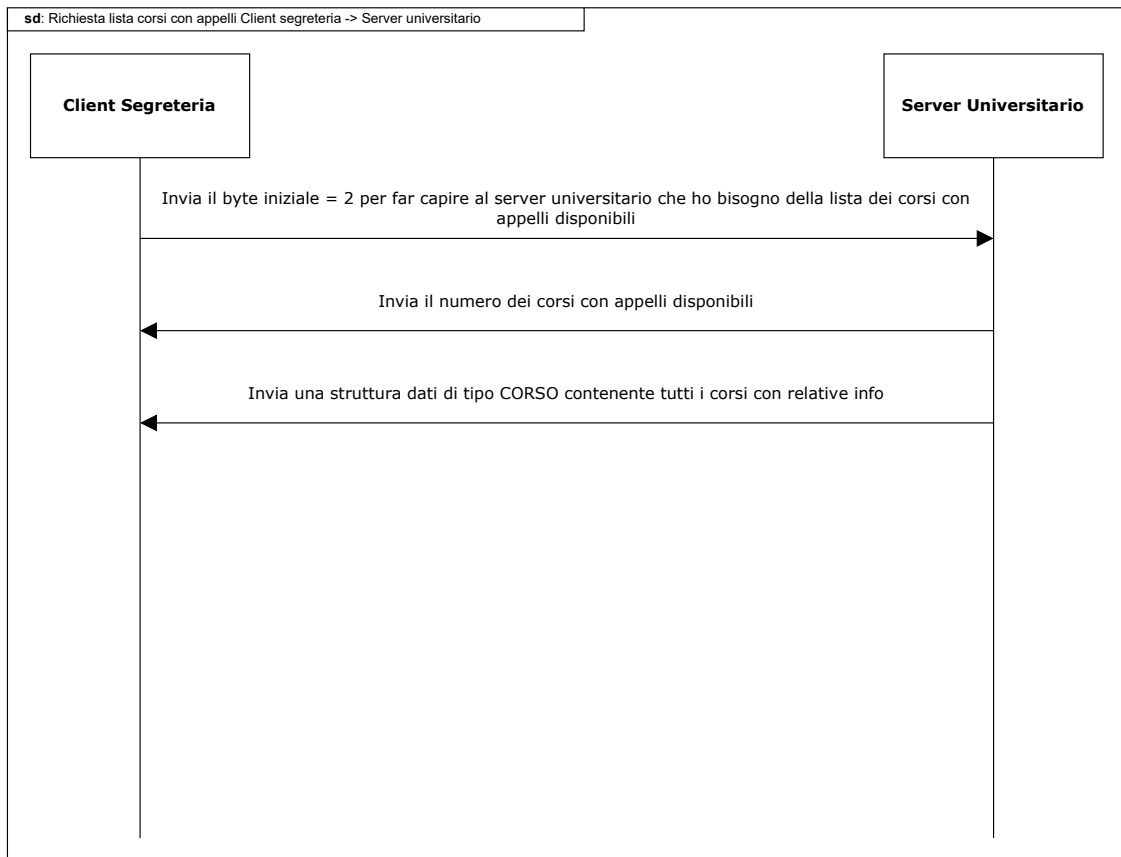


Figura 2.2: Richiesta della lista dei corsi con appelli disponibili dal client segreteria al server universitario.

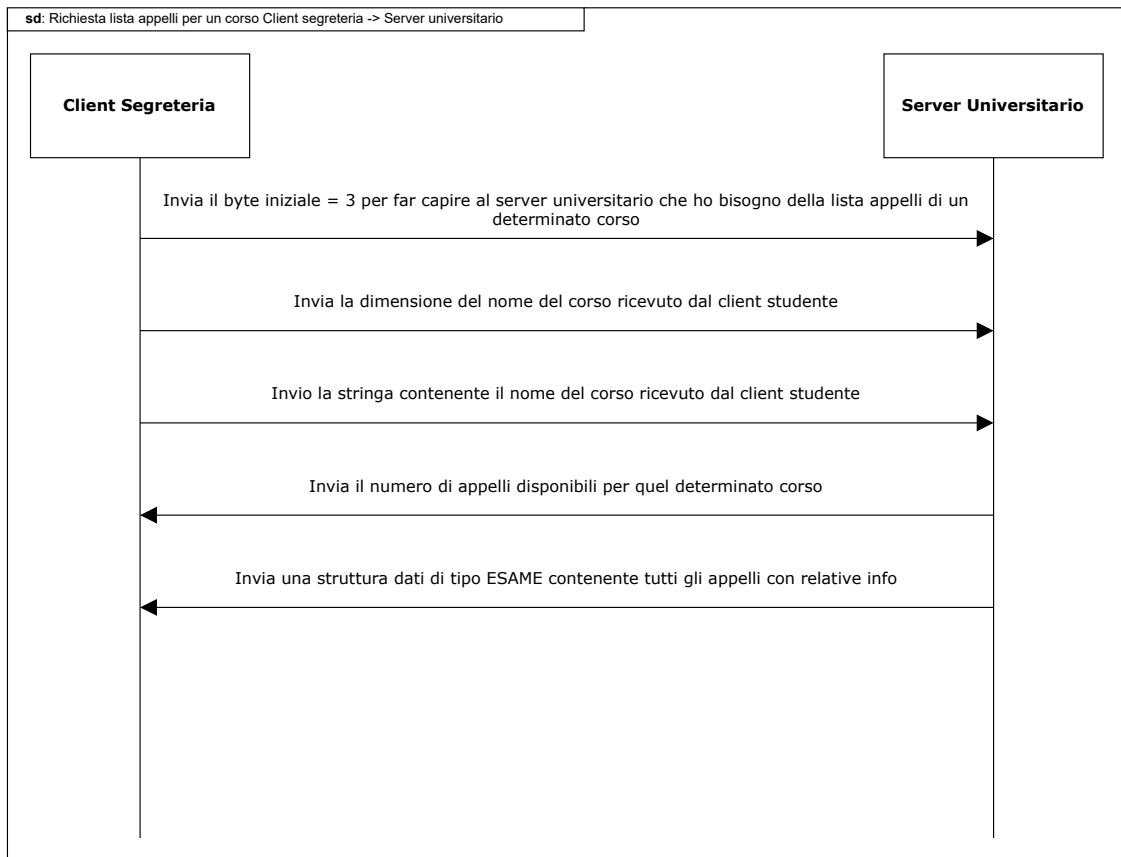


Figura 2.3: Richiesta della lista appelli di un determinato corso dal client segreteria al server universitario.



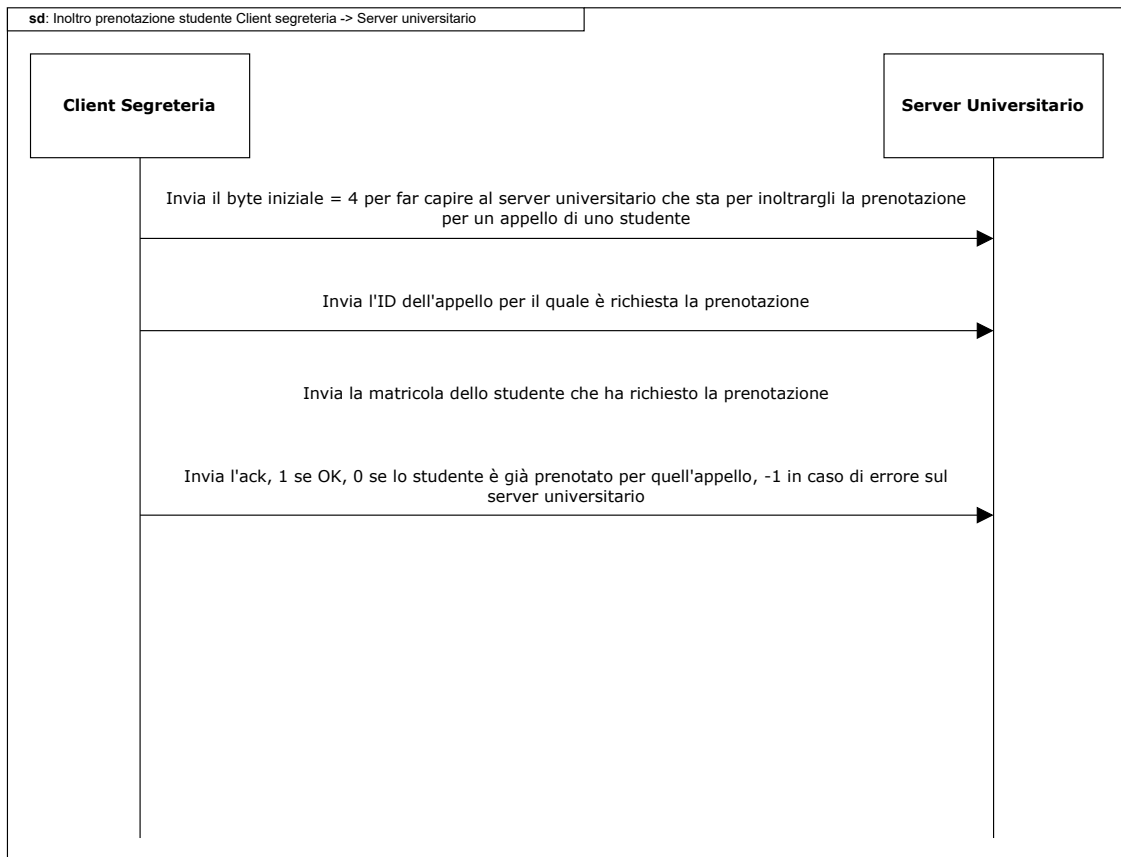


Figura 2.4: Inoltro della prenotazione di uno studente dal client segreteria al server universitario.

### 2.5.2 Comunicazione Client studente/Server segreteria

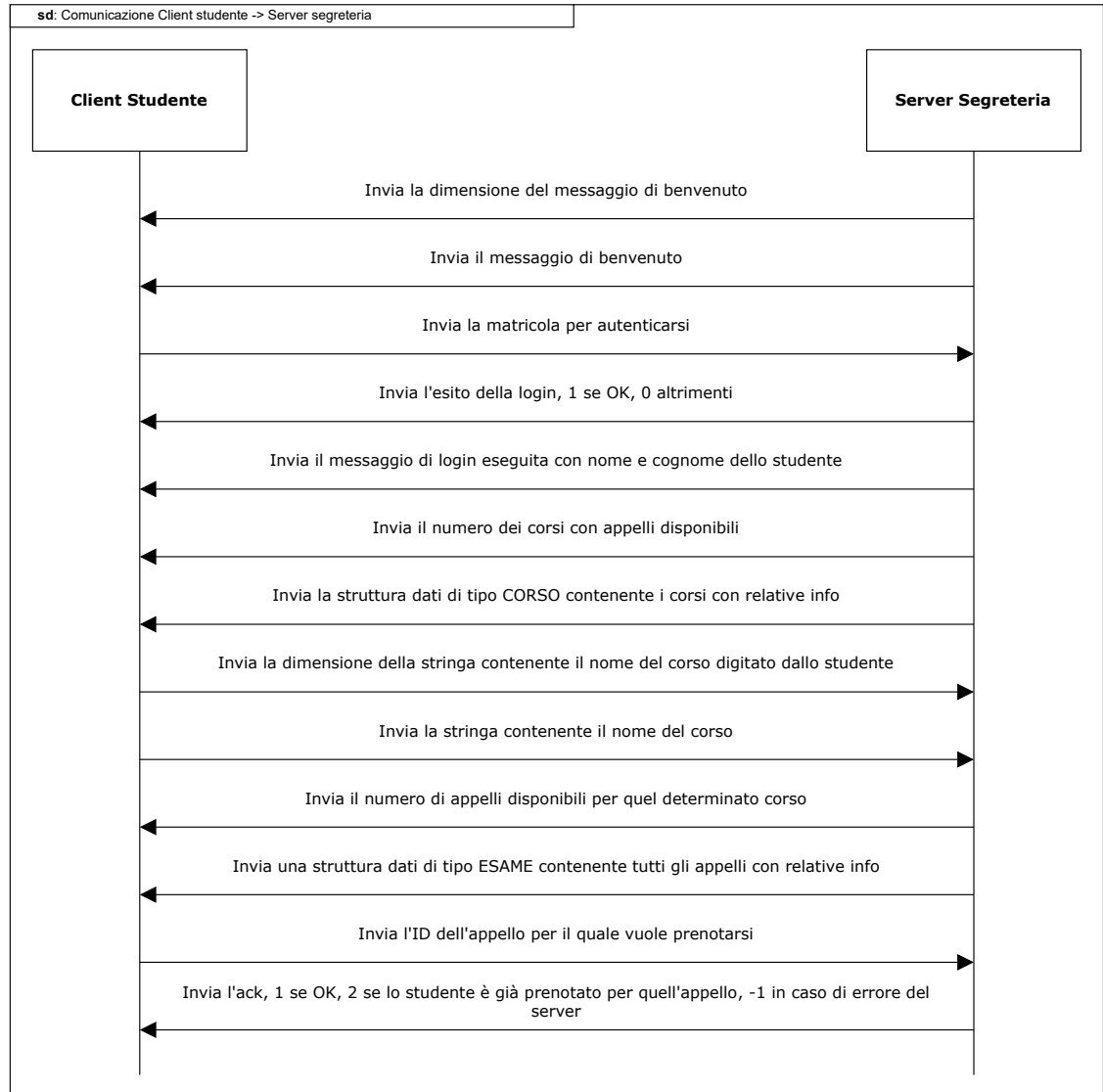
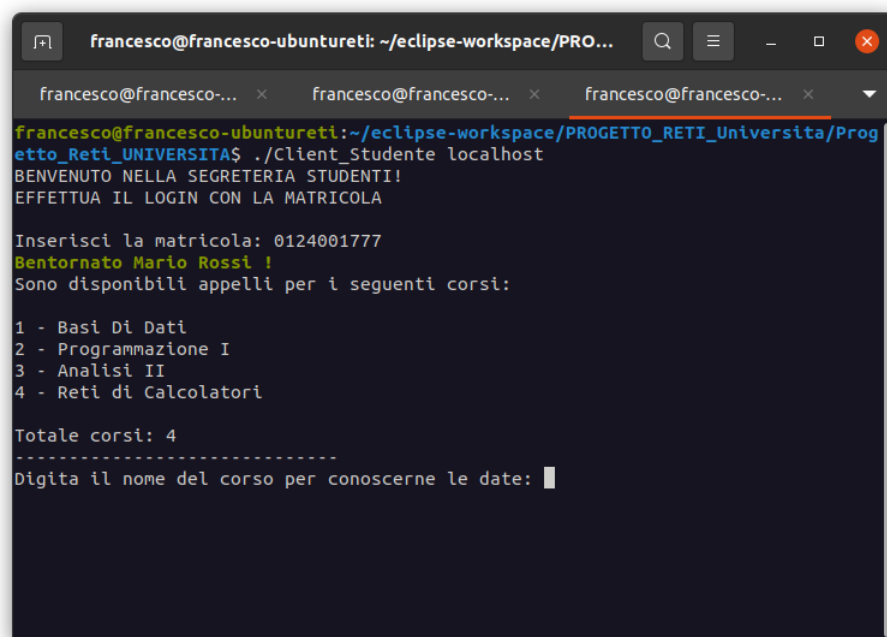


Figura 2.5: Connessione del client studente con il server della segreteria. Autenticazione, ricerca appelli e richiesta di prenotazione.

# Capitolo 3

## Screenshots

### 3.1 Client studente



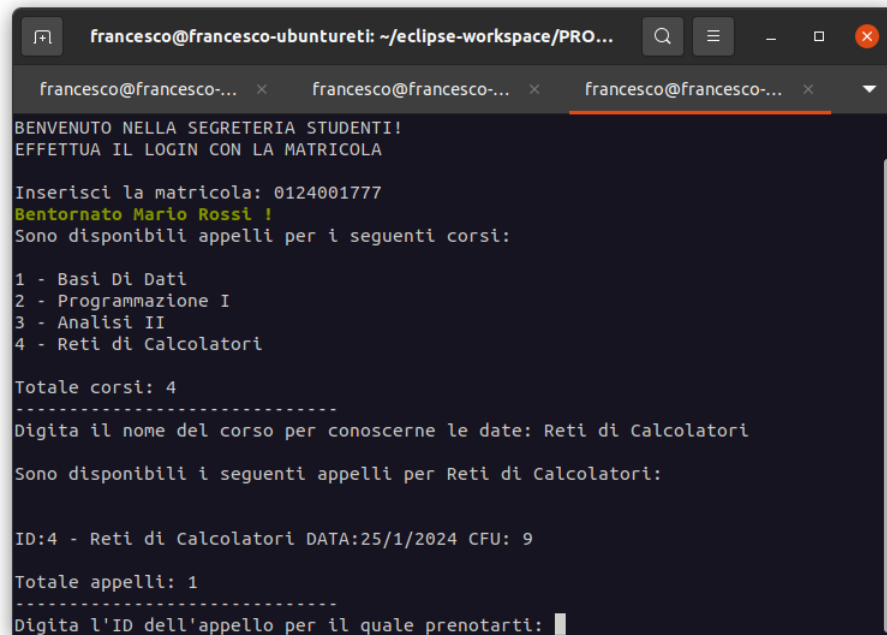
```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PROGETTO_RETI_Universita/Progetto_Reti_UNIVERSITA$ ./Client_Studente localhost
BENVENUTO NELLA SEGRETERIA STUDENTII!
EFFETTUA IL LOGIN CON LA MATRICOLA

Inserisci la matricola: 0124001777
Bentornato Mario Rossi !
Sono disponibili appelli per i seguenti corsi:

1 - Basi Di Dati
2 - Programmazione I
3 - Analisi II
4 - Reti di Calcolatori

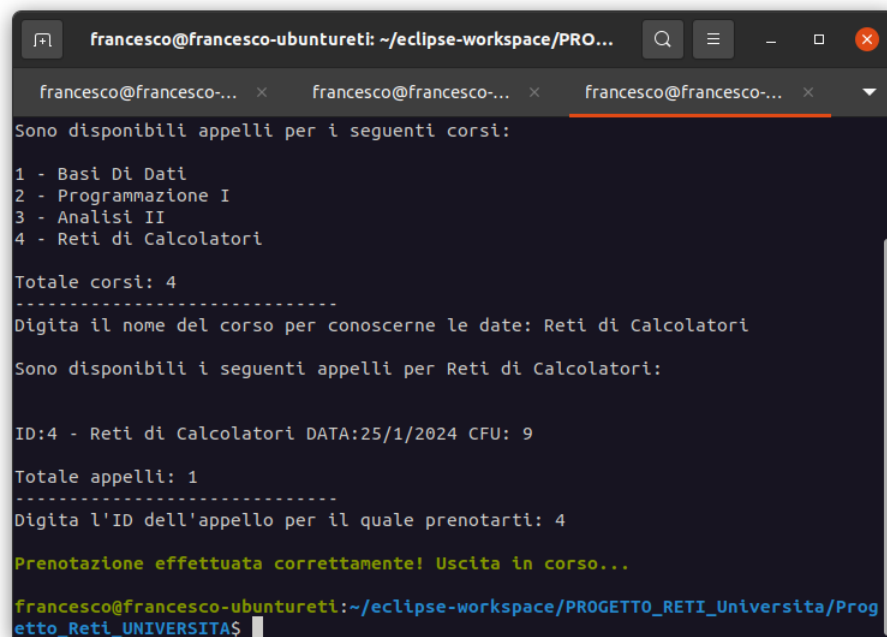
Totale corsi: 4
-----
Digita il nome del corso per conoscerne le date: 
```

Figura 3.1: Client studente. Una volta eseguita l'autenticazione viene visualizzata la lista degli esami con appelli disponibili ricevuta dal server della segreteria.



```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PRO...  
francesco@francesco-... x francesco@francesco-... x francesco@francesco-... x  
BENVENUTO NELLA SEGRETERIA STUDENTI!  
EFFETTUA IL LOGIN CON LA MATRICOLA  
  
Inserisci la matricola: 0124001777  
Bentornato Mario Rossi !  
Sono disponibili appelli per i seguenti corsi:  
  
1 - Basi Di Dati  
2 - Programmazione I  
3 - Analisi II  
4 - Reti di Calcolatori  
  
Totale corsi: 4  
-----  
Digita il nome del corso per conoscerne le date: Reti di Calcolatori  
  
Sono disponibili i seguenti appelli per Reti di Calcolatori:  
  
ID:4 - Reti di Calcolatori DATA:25/1/2024 CFU: 9  
  
Totale appelli: 1  
-----  
Digita l'ID dell'appello per il quale prenotarti: 
```

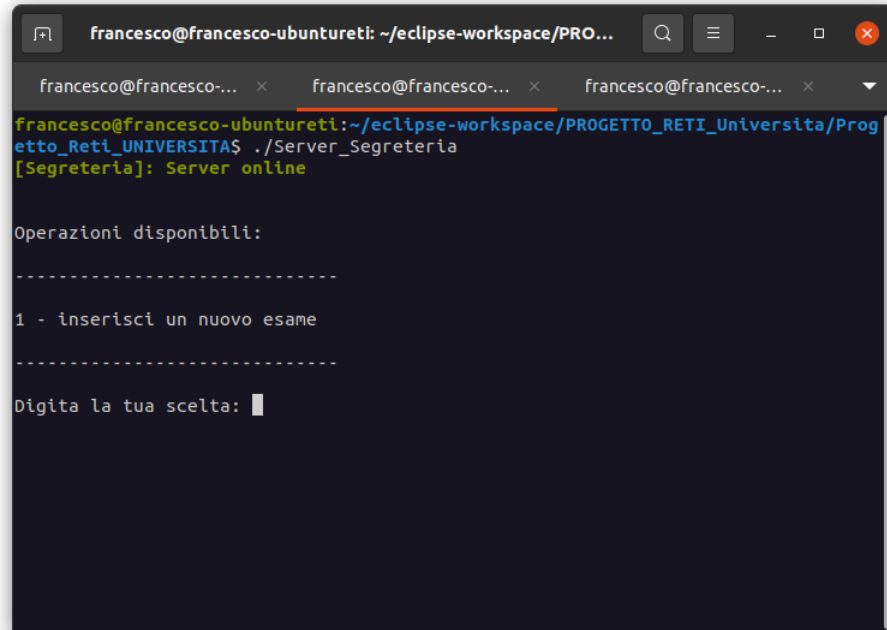
Figura 3.2: Lo studente richiede le date di appello per il corso Reti di Calcolatori. Il client le riceve dal server della segreteria e le stampa a schermo.



```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PRO...  
francesco@francesco-... x francesco@francesco-... x francesco@francesco-... x  
Sono disponibili appelli per i seguenti corsi:  
  
1 - Basi Di Dati  
2 - Programmazione I  
3 - Analisi II  
4 - Reti di Calcolatori  
  
Totale corsi: 4  
-----  
Digita il nome del corso per conoscerne le date: Reti di Calcolatori  
  
Sono disponibili i seguenti appelli per Reti di Calcolatori:  
  
ID:4 - Reti di Calcolatori DATA:25/1/2024 CFU: 9  
  
Totale appelli: 1  
-----  
Digita l'ID dell'appello per il quale prenotarti: 4  
  
Prenotazione effettuata correttamente! Uscita in corso...  
francesco@francesco-ubuntureti:~/eclipse-workspace/PROGETTO_RETI_Universita/Prog  
etto_Reti_UNIVERSITA$ 
```

Figura 3.3: Lo studente digita l'ID dell'appello per il quale vuole prenotarsi. Il client invia la richiesta al server della segreteria e riceve un ack di conferma.

## 3.2 Client/Server Segreteria

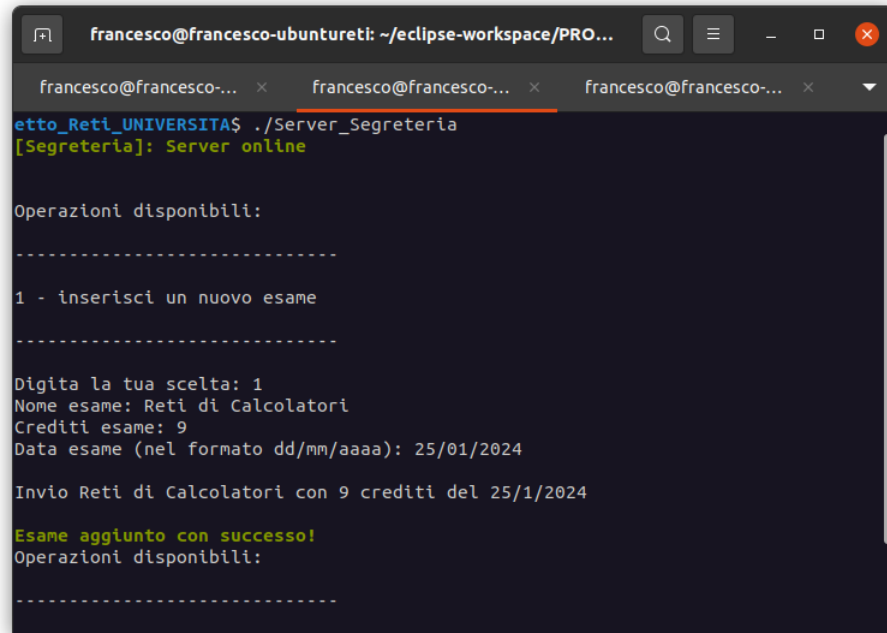


```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PRO...
francesco@francesco-... x francesco@francesco-... x francesco@francesco-... x
francesco@francesco-ubuntureti:~/eclipse-workspace/PROGETTO_RETI_Universita/Progetto_Reti_UNIVERSITA$ ./Server_Segreteria
[Segreteria]: Server online

Operazioni disponibili:
-----
1 - inserisci un nuovo esame
-----

Digita la tua scelta: █
```

Figura 3.4: Una volta avviata l'applicazione della segreteria un messaggio mostra che il processo server é stato creato ed il server é online. Successivamente compare il menú con le operazioni disponibili, in questo caso l'unica disponibile é l'inserimento di un nuovo esame sul server universitario.



```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PRO...
francesco@francesco-... x francesco@francesco-... x francesco@francesco-... x
etto_Reti_UNIVERSITA$ ./Server_Segreteria
[Segreteria]: Server online

Operazioni disponibili:
-----
1 - inserisci un nuovo esame
-----

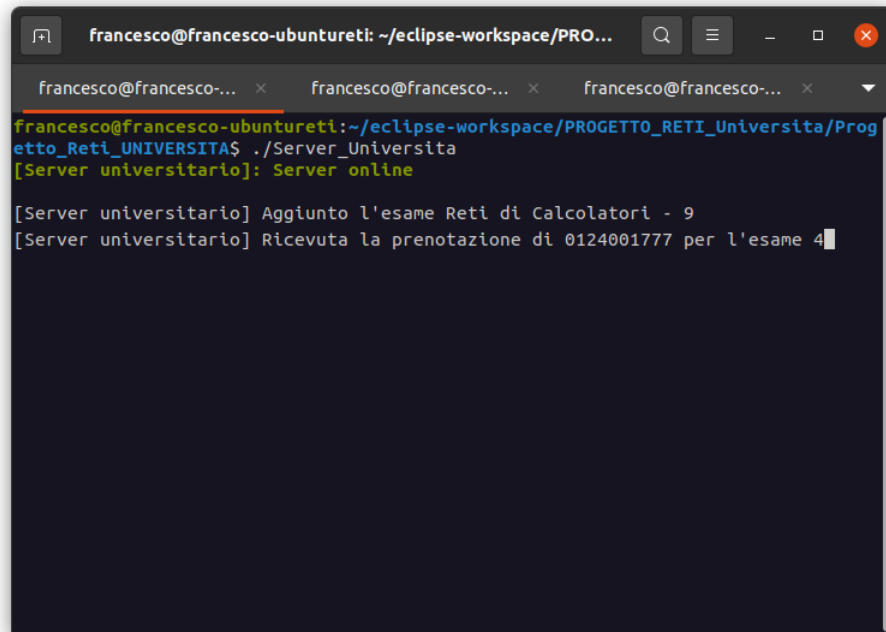
Digita la tua scelta: 1
Nome esame: Reti di Calcolatori
Crediti esame: 9
Data esame (nel formato dd/mm/aaaa): 25/01/2024

Invio Reti di Calcolatori con 9 crediti del 25/1/2024

Esame aggiunto con successo!
Operazioni disponibili:
-----
```

Figura 3.5: Scegliendo l'opzione 1 del menù l'applicazione chiederà all'utente il nome del corso, i crediti e la data dell'appello. Una volta inseriti i dati richiederà al server universitario di inserirlo, successivamente riceverà un'ack che conferma o meno l'inserimento.

### 3.3 Server Universitario



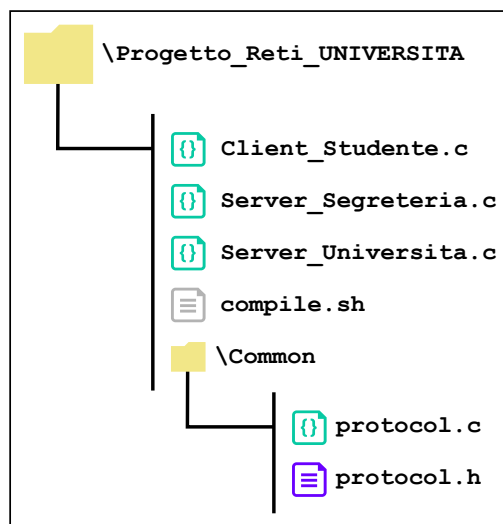
```
francesco@francesco-ubuntureti: ~/eclipse-workspace/PRO...  
francesco@francesco-... x francesco@francesco-... x francesco@francesco-... x  
francesco@francesco-ubuntureti:~/eclipse-workspace/PROGETTO_RETI_Universita/Prog  
etto_Reti_UNIVERSITA$ ./Server_Universita  
[Server universitario]: Server online  
  
[Server universitario] Aggiunto l'esame Reti di Calcolatori - 9  
[Server universitario] Ricevuta la prenotazione di 0124001777 per l'esame 4
```

Figura 3.6: Il server universitario stampa a video tutte le operazioni eseguite. In questo caso specifico vediamo la richiesta di inserimento di un appello da parte della segreteria, e la ricezione di una prenotazione di uno studente per l'appello con  $ID = 4$

# Capitolo 4

## Manuale utente

### 4.1 Struttura dei sorgenti



I sorgenti *protocol.h* e *protocol.c* presenti nella cartella *Common* contengono le funzioni wrapped `FullRead`, `FullWrite`, `Socket`, `Bind`, `Listen`, `Accept` e `Connect`.

Tutti i client/servers includono nei loro sorgenti l'header *protocol.h* e fanno uso delle funzioni considerate in precedenza.

Il file *compile.sh* è uno script shell che esegue la compilazione di tutti i sorgenti C e produce gli eseguibili delle 3 applicazioni.

#### 4.1.1 Download dei sorgenti

I sorgenti sono salvati in una repository **Github**  e possono essere visionati al seguente link :

[https://github.com/francescopicone/Progetto\\_Reti\\_UNIVERSITA](https://github.com/francescopicone/Progetto_Reti_UNIVERSITA)



## 4.2 Istruzioni per la compilazione

É possibile eseguire la compilazione in 2 modi:

- Con l'esecuzione dello script shell "compile.sh". In questo caso basta eseguire il terminale, muoversi verso la cartella contenente i sorgenti e digitare il comando :

**`./compile.sh`**

Tutti i sorgenti saranno compilati e si potrà passare all'esecuzione.

- Compilando i sorgenti uno alla volta.

Per compilare il client studente:

**`gcc Client_Studente.c Common/protocol.c -o Client_Studente`**

Per compilare il server della segreteria:

**`gcc Server_Segreteria.c Common/protocol.c -o Server_Segreteria`**

Per compilare il server universitario:

**`gcc Server_Universita.c Common/protocol.c -o Server_Universita`**

## 4.3 Istruzioni per l'esecuzione

É consigliato eseguire le applicazioni nel seguente ordine:

1. **`./Server_Universitario`** per eseguire il server universitario
2. **`./Server_Segreteria`** per eseguire il server della segreteria
3. **`./Client_Studente localhost`** per eseguire il client studente