Università degli Studi di Messina

Scienze Informatiche

Francesco Montecucco MAT 531620
Francesco Bartolomeo MAT 528405

Progetto Laboratorio di Reti e Sistemi Distribuiti 23/24
Gestione di un sistema bibliotecario

5 Luglio 2024

Indice

Stato dell'arte

Introduzione

Nell'era digitale moderna, la gestione efficiente delle risorse è diventata un elemento cruciale per molte istituzioni, tra cui le biblioteche. Con l'avvento di Internet e delle tecnologie di rete, è ora possibile amministrare sistemi complessi da remoto, rendendo i processi più fluidi e accessibili.

Paradigma REST

Il paradigma REST (Representational State Transfer) è uno stile architettonico utilizzato per la progettazione di sistemi distribuiti.

- Le risorse sono identificate da URI
- Le operazioni su queste risorse sono eseguite tramite metodi standard come GET, POST, PUT e DELETE, attraverso richieste HTTP

Multithreading in Python

Il multithreading è una tecnica che consente l'esecuzione simultanea di più thread all'interno di un singolo processo.

- La libreria threading fornisce strumenti per creare e gestire thread

Particolarmente utile per le applicazioni client/server, dove è necessario gestire più connessioni simultanee.

Database MySQL

MySQL è un sistema di gestione di database relazionali (RDBMS) open source, ampiamente utilizzato in applicazioni web.

- Supporta il linguaggio SQL per la gestione dei dati
- Offre funzionalità avanzate come transazioni, integrità referenziale e procedure memorizzate

Descrizione del problema

Descrizione

Realizzare in Python un programma multithreading client/server per la gestione da remoto di un sistema bibliotecario secondo il paradigma REST usando MYSQL.

Obiettivo del progetto

L'obiettivo principale di questo progetto è creare un sistema che permetta agli utenti di interagire con la biblioteca da remoto, consentendo operazioni come:

- Ricerca di libri
- Verifica della disponibilità
- Prestito di un libro
- Restituzione di un libro

Implementazione

URL e Variabili di stato

Il file inizia definendo l'URL base del server locale a cui il client farà riferimento per le richieste HTTP e le variabili di stato per gestire lo stato del login dell'utente, il ruolo e l'id di quest'ultimo.

Menù Principale

Il codice include un loop principale che gestisce le operazioni dell'applicazione a seconda dello stato di login dell'utente. Offre un menu interattivo che permette di selezionare le diverse operazioni disponibili a seconda del ruolo dell'utente.

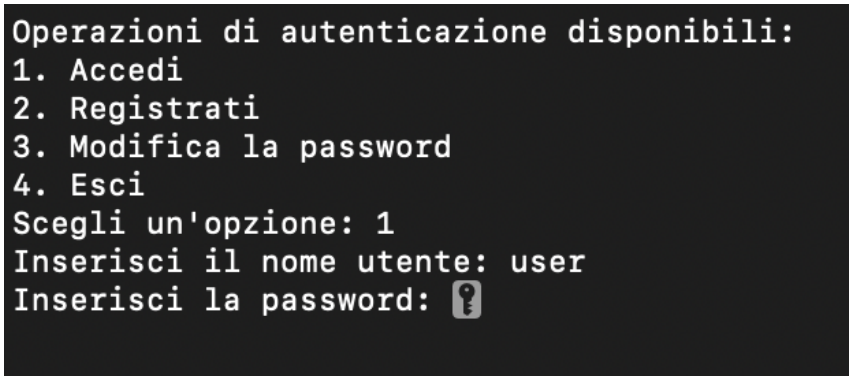
Gestione dell'Autenticazione

- Login (**login()**)
- Registrazione (**register()**)
- Cambio password (**change_password()**)

```
Operazioni di autenticazione disponibili:
1. Accedi
2. Registrati
3. Modifica la password
4. Esci
Scegli un'opzione:
```

Considerazioni di Sicurezza

È stato effettuato l'hashing delle password degli utenti utilizzando la libreria 'bcrypt' prima di essere memorizzate nel database. In fase di login, la password scritta dall'utente, viene nascosta andando a simulare il comportamendo di Linux, utilizzando la libreria 'getpass'.



Gestione della Biblioteca

L'utente può interagire con l'applicazione attraverso una serie di funzioni:

- Mostrare a schermo di tutti i libri in inventario (**get_libri()**)
- Mostrare a schermo i dettagli di un singolo libro (**get_libro()**)
- Aggiunta di un nuovo libro al database (**add_libro()**)
- Aggiornamento informazioni di un libro presente nel database (**update_libro()**)
- Eliminazione di un libro dal database (**delete_libro()**)
- Registrazione di un nuovo prestito (**new_prestito()**)
- Restituzione di un libro (**restituisci_libro()**)
- Recupero della disponibilità di un libro (**get_disponibilita()**)
- Recupero dei propri prestiti (passati/correnti) (**get_user_prestiti()**)

Gestione della Biblioteca

```
----------------------------
Login effettuato con successo
Ruolo dell'utente: Utente
----------------------------

Operazioni disponibili:
1. Visualizza tutti i libri
2. Visualizza un libro
3. Aggiungi un nuovo libro (solo amministratori)
4. Aggiorna un libro esistente (solo amministratori)
5. Elimina un libro (solo amministratori)
6. Registra un nuovo prestito
7. Restituisci un libro
8. Visualizza i tuoi prestiti
9. Esci
Scegli un'opzione:
```

Funzioni Ausiliarie

Vi sono anche funzioni ausiliarie come clear_screen() per pulire la console.

```
def clear_screen():
    os.system('cls' if os.name == 'nt' else 'clear')
```

Panoramica dell'Implementazione

Il server è implementato utilizzando il modulo 'http.server' di Python per gestire le richieste HTTP. Utilizza 'socketserver.ThreadingMixIn' per supportare connessioni concorrenti. I componenti principali includono:

- Configurazione e Connessione al Database
- Meccanismo di Threading e Locking
- Gestione delle Richieste
- Registrazione dei tempi di risposta

Endpoint e Operazioni

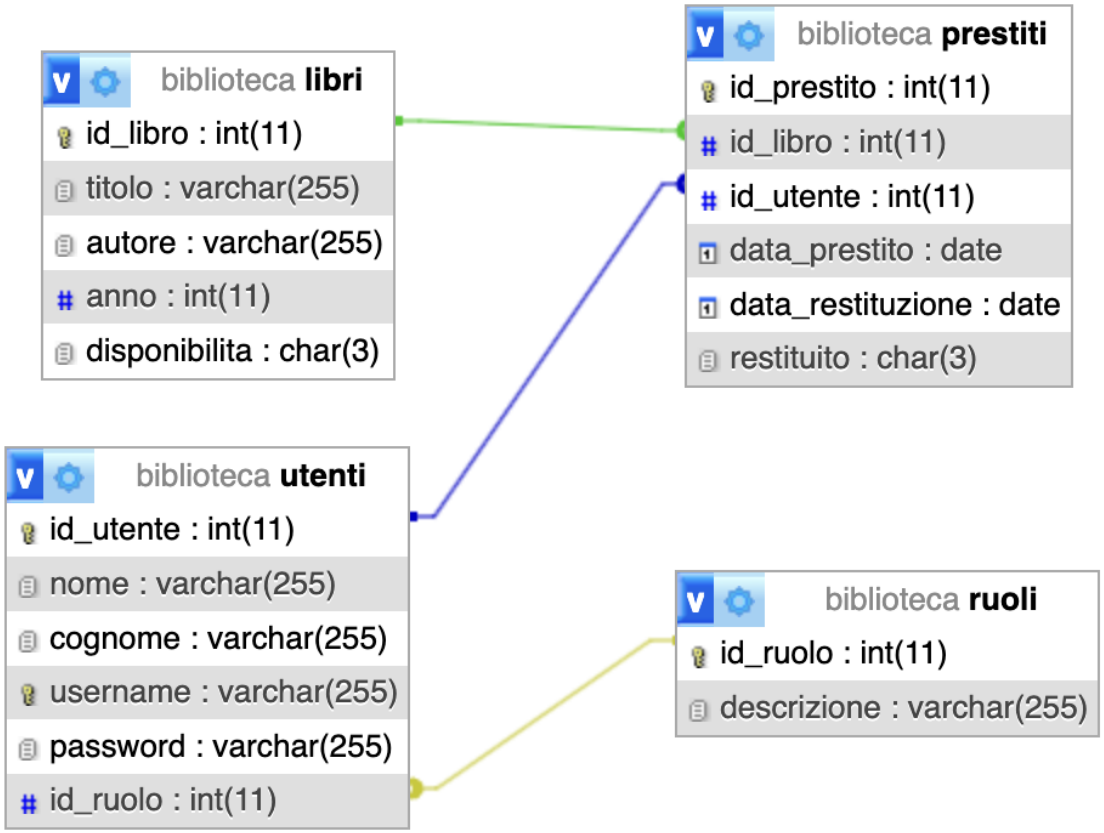
- **Operazioni GET:** Recupero di tutti i libri (‘/**libri**‘), di un libro specifico (‘/**libri/id**‘), prestiti dell'utente (‘/**prestiti**‘).
- **Operazioni POST:** Aggiunta di un libro (‘/**libri**‘), registrazione di un nuovo utente (‘/**register**‘), login dell'utente (‘/**login**‘), cambio password (‘/**change_password**‘), richiesta di un prestito (‘/**prestiti**‘), restituzione di un libro (‘/**restituzioni**‘).
- **Operazioni PUT:** Aggiornamento dei dettagli di un libro (’/**libri/id**’).
- **Operazioni DELETE:** Eliminazione di un libro (’/**libri/id**’).

Creazione del Database

- Il database "biblioteca" e le relative tabelle sono state create attraverso una serie di query in linguaggio SQL
- Attraverso altre query SQL sono stati inseriti libri e ruoli all'interno delle tabelle

Tabelle

- Libri - Contiene le informazioni sui libri
- Ruoli - Amministratore / Utente
- Utenti - Contiene le informazioni degli utenti
- Prestiti - Contiene le informazioni sui prestiti



Gestione del Multithreading

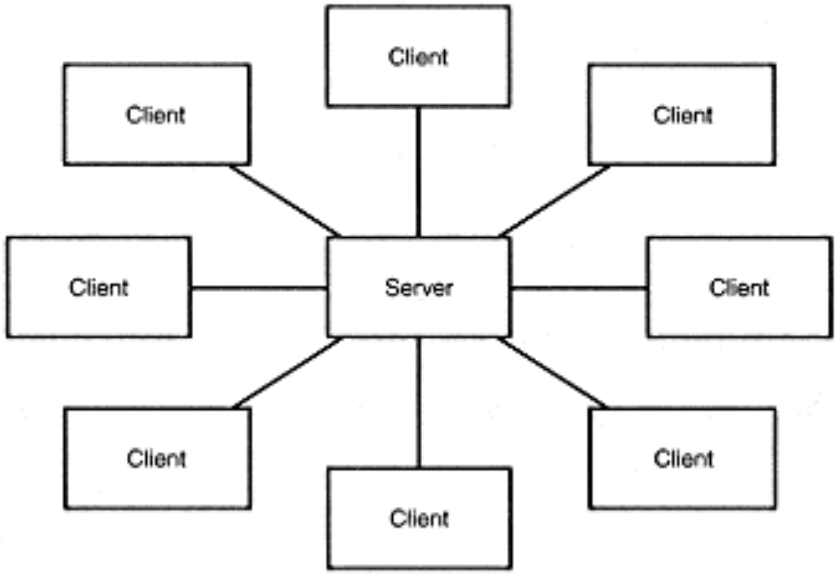
Il multithreading viene gestito attraverso la classe "**ThreadingSimpleServer**", che eredita da due classi: "**socketserver.ThreadingMixIn**" e "**HTTPServer**". Questa classe viene utilizzata per creare un server HTTP che gestisce ogni richiesta in un thread separato, permettendo al server di gestire più connessioni contemporaneamente.

L'accesso al database (risorsa condivisa) viene gestito attraverso l'utilizzo di un lock, implementato utilizzando la libreria **threading**

Risultati sperimentali

Risultati sperimentali

Sono stati effettuati dei test per andare a verificare se il server è in grado di gestire grossi carichi, nel caso in cui una grande quantità di client invii richieste contemporaneamente.



Risultati sperimentali

Per fare questo, è stata creata una versione del client esclusivamente per i test (client_login_automatico.py), dove si vanno ad automatizzare il login, ed una richiesta di visualizzazione di tutti i libri.

```
# Funzione per il login automatico
def automatic_login():
    username = "user"
    password = "pass"
    login(username, password)

# Funzione per le richieste automatiche
def automatic_requests():
    while logged_in:
        time.sleep(5)
        get_libri()
        time.sleep(100000)
```

Run_process_linux

Il codice è progettato per avviare più istanze di un client Python (client_login_automatico.py) simultaneamente, utilizzando il terminale GNOME come ambiente di esecuzione separato per ciascun client. Ciò è particolarmente utile quando si desidera testare o simulare comportamenti multi-client senza dover avviare manualmente ogni istanza.

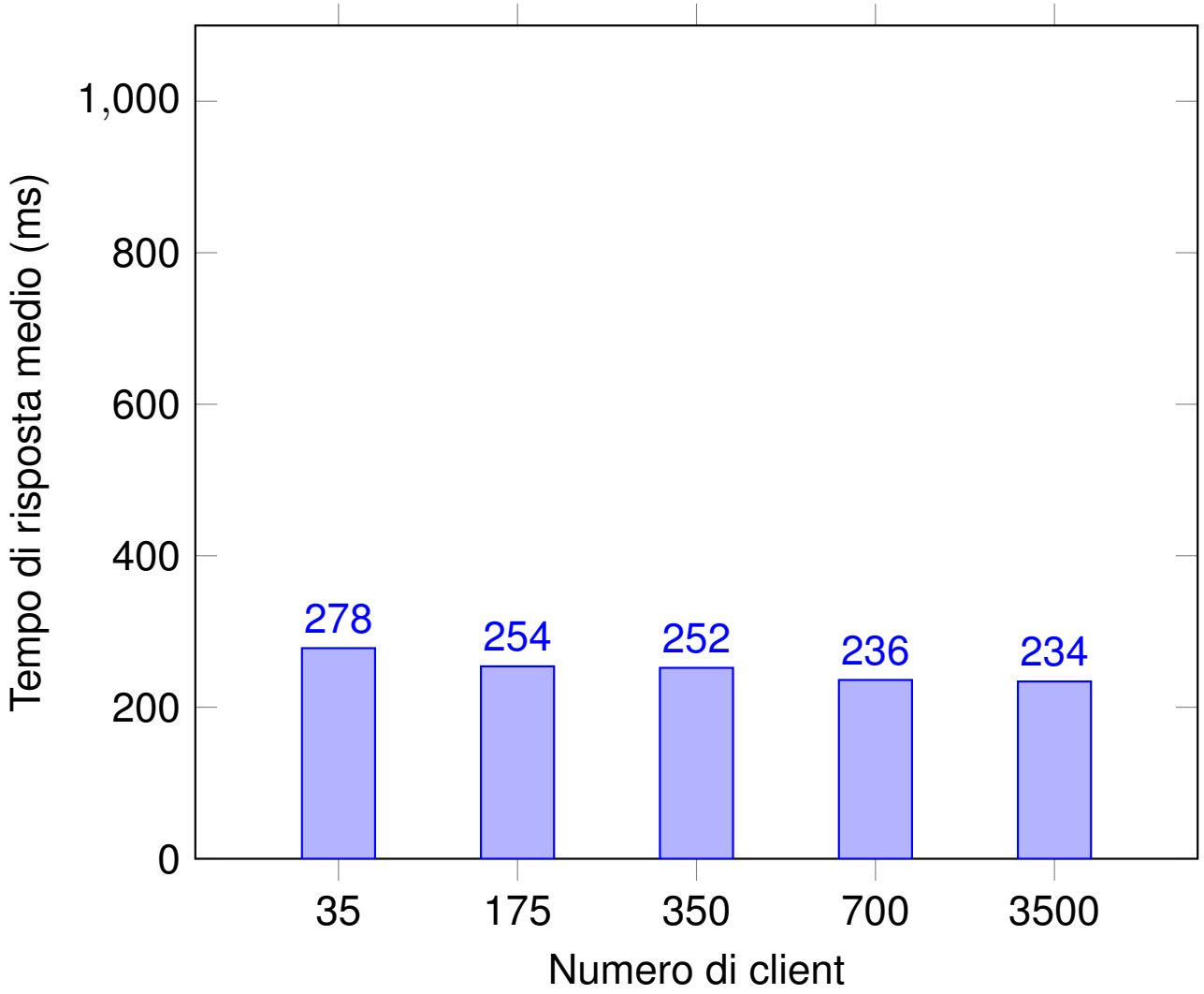
```
subprocess.Popen(["gnome-terminal", "--", "python3", "client_login_automatico.py"])
```

Risultati dei test

```
192.168.128.235 - - [26/Jun/2024 13:44:47] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:48] "GET /libri HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:48] "POST /login HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:48] "GET /libri HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:49] "GET /libri HTTP/1.1" 200 -
192.168.128.237 - - [26/Jun/2024 13:44:49] "GET /libri HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:44:49] "POST /login HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:49] "GET /libri HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:50] "POST /login HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:50] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:50] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:50] "GET /libri HTTP/1.1" 200 -
192.168.128.237 - - [26/Jun/2024 13:44:51] "POST /login HTTP/1.1" 200 -
192.168.128.236 - - [26/Jun/2024 13:44:51] "POST /login HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:52] "POST /login HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:52] "POST /login HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:44:52] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:53] "POST /login HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:53] "GET /libri HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:44:54] "GET /libri HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:44:54] "POST /login HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:44:54] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:54] "GET /libri HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:55] "POST /login HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:55] "GET /libri HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:44:56] "POST /login HTTP/1.1" 200 -
192.168.128.237 - - [26/Jun/2024 13:44:56] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:56] "GET /libri HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:44:56] "GET /libri HTTP/1.1" 200 -
192.168.128.236 - - [26/Jun/2024 13:44:56] "GET /libri HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:44:57] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:57] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:57] "POST /login HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:44:57] "POST /login HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:44:58] "GET /libri HTTP/1.1" 200 -
192.168.128.236 - - [26/Jun/2024 13:44:59] "POST /login HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:44:59] "GET /libri HTTP/1.1" 200 -
192.168.128.235 - - [26/Jun/2024 13:45:00] "GET /libri HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:45:01] "GET /libri HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:45:02] "GET /libri HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:45:02] "GET /libri HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:45:04] "POST /login HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:45:04] "POST /login HTTP/1.1" 200 -
192.168.128.230 - - [26/Jun/2024 13:45:04] "GET /libri HTTP/1.1" 200 -
192.168.128.236 - - [26/Jun/2024 13:45:04] "GET /libri HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:45:08] "GET /libri HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:45:08] "POST /login HTTP/1.1" 200 -
192.168.128.238 - - [26/Jun/2024 13:45:09] "GET /libri HTTP/1.1" 200 -
192.168.128.233 - - [26/Jun/2024 13:45:09] "GET /libri HTTP/1.1" 200 -
192.168.128.231 - - [26/Jun/2024 13:45:13] "GET /libri HTTP/1.1" 200 -
```

Figura: Richieste ricevute dal server da parte dei diversi client in OpenLAB

Risultati dei test



Risultati dei test

Dai test effettuati si può notare come il server sia stato in grado di gestire le numerose richieste da parte dei client senza subire rallentamenti di alcun tipo

Conclusioni e sviluppi futuri

Conclusioni

Il progetto ha dimostrato come l'utilizzo della programmazione multithread possa migliorare significativamente l'efficienza e la reattività di un sistema complesso. L'implementazione ha permesso di gestire simultaneamente più richieste degli utenti, riducendo i tempi di attesa e migliorando l'esperienza complessiva dell'utente.

- Il sistema è stato progettato per essere scalabile, consentendo l'aggiunta di nuove funzionalità e la gestione di un numero crescente di utenti e libri senza comprometterne le prestazioni.

Possibili sviluppi futuri

Ci sono diversi ambiti in cui il progetto potrebbe essere ulteriormente sviluppato e migliorato:

- Aggiungere funzionalità avanzate come la vendita di libri, o la possibilità di recensire i libri letti
- Implementare algoritmi di raccomandazione basati su machine learning per suggerire libri agli utenti in base alle loro letture passate
- Sviluppare un'interfaccia utente più intuitiva e user-friendly
- Implementare l'aggiunta di altri linguaggi per rendere il sistema accessibile ad un numero maggiore di utenti

Saluti finali

GRAZIE PER
L'ATTENZIONE