
MÔ TẢ BÀI TOÁN

Xây dựng hệ thống **"Quản lý tập tin từ xa (Remote File Manager)"** sử dụng công nghệ **Java RMI** theo mô hình Command/Response tương tự giao thức POP3. Hệ thống cho phép người dùng đăng nhập và thực hiện thao tác upload/download file giữa Client và Server.

YÊU CẦU KỸ THUẬT

1. **Giao thức kết nối:** Sử dụng **Java RMI** lắng nghe tại cổng **1099**.
2. **Cơ sở dữ liệu:** Sử dụng **Microsoft Access** (file .mdb hoặc .accdb) để lưu thông tin người dùng.
3. **Thư mục làm việc mặc định:**
    - **Server:** Tự động tạo thư mục E://server khi khởi chạy nếu chưa tồn tại.
    - **Client:** Tự động tạo thư mục E://client khi khởi chạy nếu chưa tồn tại.
4. **Quản lý trạng thái:** Hệ thống phải xử lý theo phiên (Session). Server cần phân biệt được trạng thái của từng Client (Chưa đăng nhập / Đã đăng nhập).

CHI TIẾT CHỨC NĂNG

Sinh viên thực hiện viết chương trình bao gồm 2 phần: **Server** và **Client** đáp ứng các tập lệnh sau:

I. TRẠNG THÁI TRƯỚC KHI ĐĂNG NHẬP (PRE-LOGIN STATE)

Ở trạng thái này, Client chỉ được phép thực hiện các lệnh xác thực.

1. **Lệnh: UNAME <username>**
    - **Mô tả:** Client gửi tên đăng nhập lên Server.
    - **Xử lý (Server):** Kiểm tra trong CSDL.
    - **Phản hồi:**
        - Nếu tồn tại: Trả về "OK User accepted". Chuyển sang chờ mật khẩu.
        - Nếu không tồn tại: Trả về "ERR User not found".
2. **Lệnh: PASS <password>**
    - **Mô tả:** Chỉ hoạt động nếu đã gửi UNAME hợp lệ trước đó. Client gửi mật khẩu.
    - **Xử lý (Server):** Sử dụng **JDBC** kiểm tra mật khẩu ứng với username đã nhập.
    - **Phản hồi:**
        - Đúng mật khẩu: Trả về "OK Welcome <username>". **Chuyển trạng thái sang Đã đăng nhập**.
        - Sai mật khẩu: Trả về "ERR Wrong password". Reset về trạng thái chưa nhập user.
3. **Lệnh: EXIT**
    - **Mô tả:** Đóng kết nối ứng dụng Client.

II. TRẠNG THÁI SAU KHI ĐĂNG NHẬP (POST-LOGIN STATE)

Sau khi đăng nhập thành công, Client có thể thực hiện các lệnh thao tác file. Lưu ý: Thư mục hiện hành (current directory) của phiên làm việc bắt đầu tại thư mục mặc định (E://server và E://client).

4. **Lệnh: SS_DIR <folder_name> (Set Server Directory)**
    - **Mô tả:** Yêu cầu Server chuyển đổi thư mục làm việc hiện tại **của phiên đó** sang thư mục con.
    - **Ví dụ:** Đang ở E://server, lệnh SS_DIR data sẽ chuyển sang E://server/data.
    - **Yêu cầu:** Sử dụng lớp java.io.File để kiểm tra thư mục có tồn tại không.
    - **Phản hồi:** "OK Dir changed" hoặc "ERR Directory not found".
5. **Lệnh: SC_DIR <folder_name> (Set Client Directory)**
    - **Mô tả:** Thay đổi thư mục làm việc hiện tại ở phía **Client** (Local).
    - **Xử lý:** Việc này xử lý cục bộ tại Client, không cần gọi RMI, nhưng phải kiểm tra tính hợp lệ của đường dẫn trên máy Client.
    - **Thông báo:** In ra màn hình Client đường dẫn mới.
6. **Lệnh: UPLOAD <local_file> <server_file>**
    - **Mô tả:** Client đọc file từ thư mục hiện tại của Client, gửi lên thư mục hiện tại của Server.
    - **Kỹ thuật:**
        - Dùng RandomAccessFile hoặc FileInputStream để đọc file ở Client.
        - Đóng gói dữ liệu (dạng byte[]) gửi qua phương thức RMI.
        - Server ghi dữ liệu ra file đích.
    - **Phản hồi:** "OK Upload success" hoặc "ERR <chi tiết lỗi>".
7. **Lệnh: DOWNLOAD <server_file> <local_file>**
    - **Mô tả:** Ngược lại với UPLOAD. Server đọc file, gửi byte[] về Client, Client ghi ra đĩa.
    - **Phản hồi:** Trả về dữ liệu file hoặc thông báo lỗi nếu file không tồn tại trên Server.
8. **Lệnh: EXIT**
    - **Mô tả:** Đăng xuất, kết thúc phiên làm việc, quay về trạng thái chưa đăng nhập hoặc đóng chương trình.