Universidad Nacional Autónoma de Honduras

Escuela de matemáticas y ciencias de la computación

Departamento de matemática aplicada

Administración de centros de cómputo

Proyecto: **Aplicación distribuida Entrega FINAL**

Sistema operativo: Linux

Fecha de entrega: lunes 12 de julio

Modo de entrega: Teams

Middleware: JavaRMI, Python Jyro, otros remote objects (NO web)

Especificación

PARTE 1

Desarrolle la aplicación cliente/servidor que haga lo siguiente:

Interfaz remota

1. obtenerEstructuraFS(): retorna la estructura de directorio del sistema de archivos.
2. leerArchivo(ruta,nombreArchivo): retorna los datos del archivo y los almacena localmente.
3. escribirArchivo(ruta,nombreArchivo, datos): reemplaza los datos del archivo en el servidor.
4. **borrarArchivo(nombreArchivo): Antes de borrar el archivo, se deben invalidar las copias de los caches de los clientes que lo están accediendo**
5. crearDirectorio(ruta, nombreDirectorio): retorna true si el directorio fue creado correctamente. Se tiene que notificar a los clientes para que obtengan la nueva estructura.
6. **borrarDirectorio(ruta, nombreDirectorio): El servidor elimina el archivo, y notifica a los clientes para que obtengan la estructura del FS nuevamente.**
7. registrarCallback(callback): registra el callback del cliente en el servidor.

Si necesita otras funciones remotas las puede agregar

Cliente

Parte #1

La estructura de directorio se debe almacenar localmente en el directorio /temp/fs11

Si hay múltiples clientes en el mismo host entonces se debe crear un directorio diferente para cada uno de ellos. /temp/fs11, /temp/fs12, /temp/fs13,…

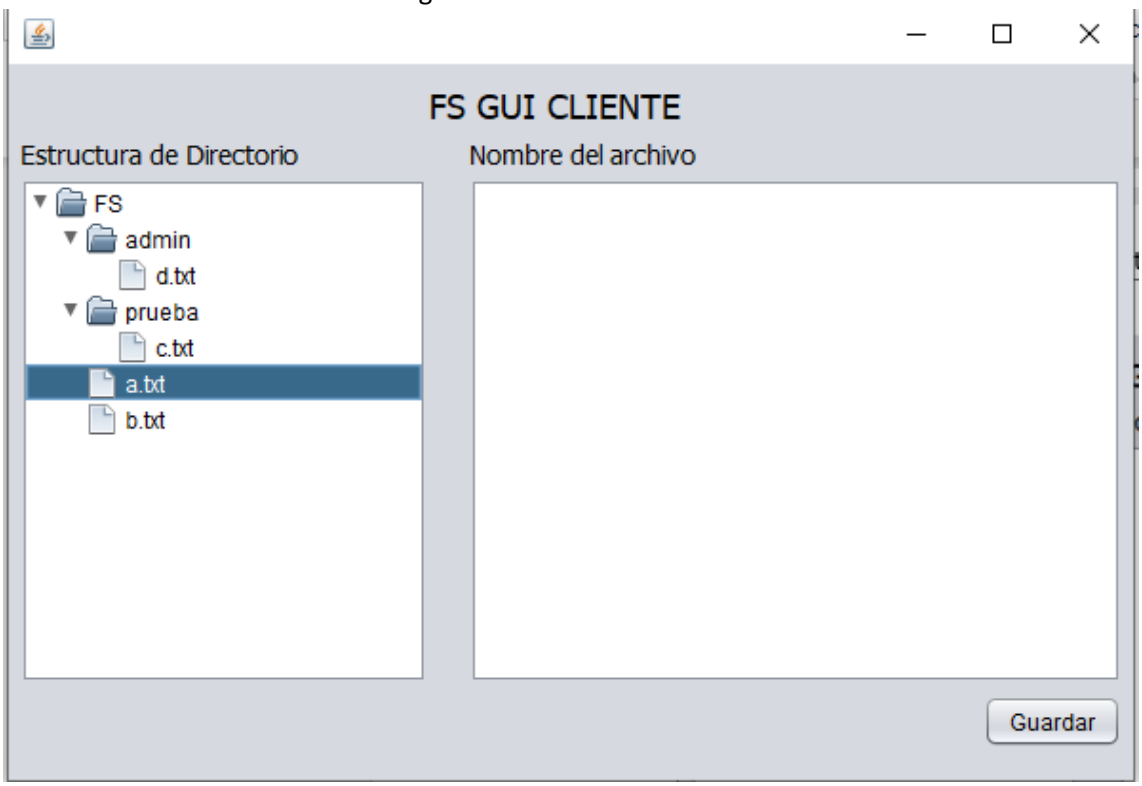
Este directorio debe almacenar:

1. La estructura de directorios: esta estructura se obtiene del servidor. Cuando se trae la estructura SOLAMENTE se obtiene los nombres de los archivos de cada directorio. NO se traen los datos de los archivos. Se debe tener en cada directorio un archivo especial archivos.txt que contiene los nombres de los archivos del directorio. Este archivo no se debe mostrar en la GUI del sistema de archivos.
2. Los datos de los archivos bajados del servidor. Solo los archivos que se acceden están localmente en el cliente. Se tiene que llevar el control de los archivos que están localmente. Además, se debe llevar el control si el archivo es válido o no.

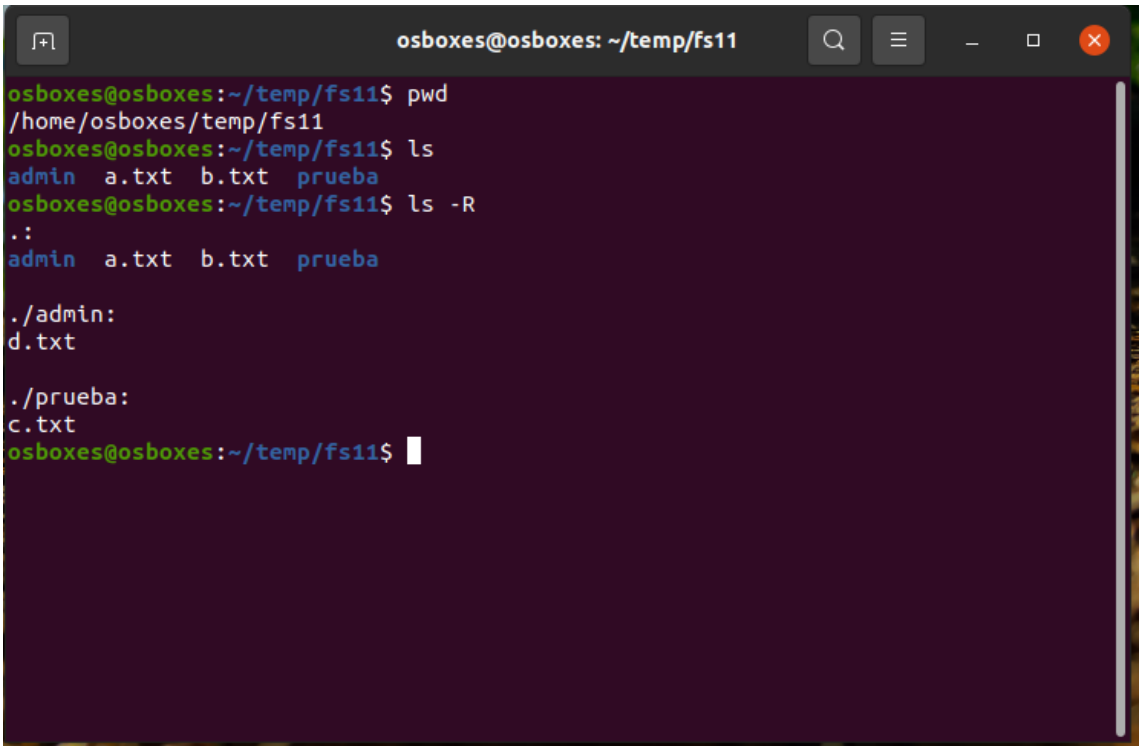
Funcionalidad

1. Se debe crear un explorador de archivos (GUI) que
    1) muestre la estructura de directorios junto con los nombres de los archivos.
    2) Muestre un área de texto editable donde se pueda leer y/o escribir en los archivos. Al seleccionar (click o menú de contexto) un archivo de la estructura de directorio se debe mostrar el contenido del archivo. Se puede modificar el archivo y guardar.
        i. Después de guardar el archivo, se debe escribir el archivo en el servidor.
2. Se debe tener un callback donde recibe las notificaciones de invalidación de archivos y de cambio de estructura. Cuando se modifica un archivo en el servidor que el cliente tiene almacenado localmente, se debe invalidar el archivo por el servidor. No se obtiene la nueva copia del archivo del servidor.
3. El cliente lee y escribe localmente en los archivos, pero debe asegurarse que archivo es válido antes de realizar las operaciones.

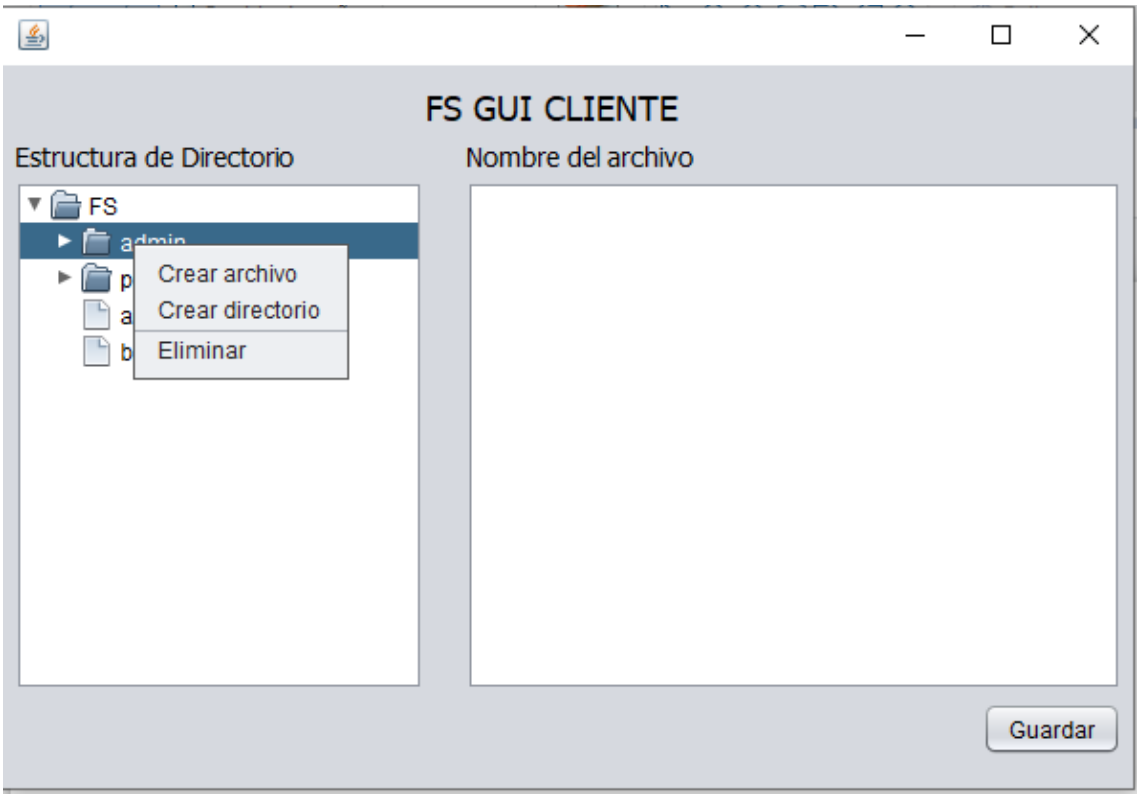
La interfaz se debe mostrar de la siguiente manera:



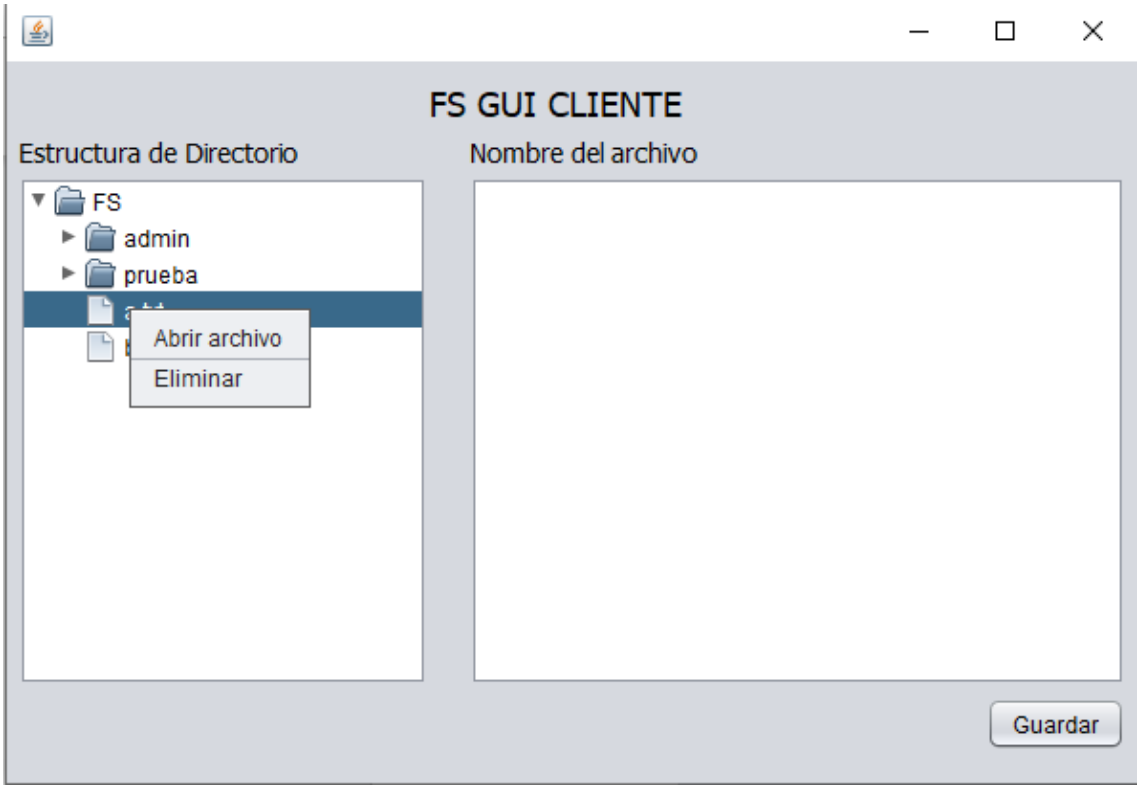
La estructura de directorio debe ser la misma que tiene el servidor.



Para las operaciones sobre los directorios se debe mostrar un menú de contexto



Para los archivos también se debe mostrar un menú de contexto



Servidor

1) Mantiene la estructura de directorio se debe almacenar localmente el directorio /temp/fs11.
2) Procesa las solicitudes del objeto remoto
3) Invalida el archivo local y la estructura en los clientes cuando este es modificado.

PARTE 2

1. Corra el servidor en un contenedor de Linux
2. Corra el RMIregistry en otro contenedor de Linux
3. Corra dos o tres clientes en Linux que accedan al servidor y al RMIRegistry