Logotipo

Descripción generada automáticamenteLogotipo

Descripción generada automáticamenteBenemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Ingeniería en Ciencias de la Computación

Materia: Programación Distribuida Aplicada

Practica 1

Profesor: Gustavo Emilio Mendoza Olguin

Alumnos:

Pérez Flores Ivonne 202141158

Primavera 2025

18 de enero de 2025

# **Introducción**

Como practica uno, se plantea la realización de un programa servidor de nombres de recursos de archivos, el cual debe de cumplir con los siguientes requisitos:

* El proceso debe de obtener la lista de archivos de una carpeta de la computadora (especificada por el usuario) y crear una lista dinámica de objetos que contenga:
* Nombre del archivo y extensión
* Permitir al usuario definir que archivos pueden publicarse y cuales no y el tiempo de actualización (TTL) (Guardar un archivo de configuración).
* Crear un hilo que actualice la lista de archivos cada 5 minutos, si hay un archivo que estaba en la lista anterior, entonces debe preguntar si se podrá publicar, si existe un archivo en la lista que ya no esta en la carpeta, entonces deberá eliminarse (los mensajes de cambio de la lista deben enviarse a un log)
* El archivo de configuración debe actualizarse cada vez que inicie el proceso con los cambios en el sistema de archivos

Asimismo, se requiere de otro hilo que inicie un socket que escuche en el puerto 50000 UDP.

# **Desarrollo**

Para comenzar con el desarrollo de esta práctica, se consideraron varios aspectos para la implementación, los cuales se enlistan a continuación:

* Para la recopilación de los archivos desde la ruta proporcionada por el usuario, se toman en cuenta solo aquellos que tienen una extensión en el nombre del archivo, excluyendo carpetas del listado.
* El almacenamiento de un archivo de configuración se realiza mediante un archivo json para facilitar el acceso a los datos del listado.
* Los mensajes de cambio se almacenan en un archivo de extensión .log.
* El lenguaje de programación utilizado para esta práctica es Python.

Una vez considerando los pasados puntos, vamos a desglosar cada parte del código para comprender su funcionamiento.

Importación de librerías de Python

Texto

Descripción generada automáticamente

Estas librerías cumplen con los siguientes aspectos:

* **Threading.** Librería para gestión y creación de hilos.
* **Time.** Gestión de tiempo en Python, nos permitirá suspender o poner a dormir un hilo el tiempo necesario.
* **Os.** Permite interactuar con el sistema operativo, se utilizará para las consultas de los listados de archivos en el sistema.
* **Json.** Librería para gestionar archivos json.
* **Socket.** Creación y gestión de sockets en Python.
* **Logging.** Librería para generar mensajes de registro para archivos log.

Clase server

Texto

Descripción generada automáticamenteConstructor

Definimos valores globales, tales como la ruta del archivo de configuración “config.json”, el formato de salida para el archivo de extensión .log y el mecanismo de sincronización para establecer una zona critica.

Texto

Descripción generada automáticamenteMétodo para iniciar conexión de socket

El método iniciar\_socket\_udp configura y gestiona un servidor UDP para recibir y procesar mensajes de clientes. Inicializa el socket, lo vincula al puerto 50000 y escucha constantemente los datos entrantes, recibidos en formato json. Al recibir un mensaje, verifica si el cliente es nuevo y lo registra. Según el contenido del mensaje, puede guardar información en un archivo JSON o validar una ruta específica, enviando la respuesta correspondiente al cliente. Cada entrada de cliente es definida por un hilo para seguir su comportamiento.

Texto

Descripción generada automáticamenteValidar ruta

Texto

Descripción generada automáticamente

El método validarRuta comprueba si la ruta proporcionada es válida y devuelve información sobre los archivos disponibles en ella. Inicializa una respuesta predeterminada indicando que la ruta no es válida y que no se deben procesar archivos. Si la ruta no corresponde a un directorio válido, se registra un error y se devuelve la respuesta sin cambios. En caso de que la ruta sea válida, intenta obtener los archivos presentes en el directorio, filtrando aquellos que contienen un punto en el nombre. Luego, verifica si la ruta ya está registrada en el archivo JSON. Si está registrada, compara los archivos existentes con los nuevos, actualizando la respuesta según corresponda. Si no está registrada, agrega todos los archivos encontrados y marca la ruta como válida. El método gestiona errores como permisos denegados o fallos inesperados, registrando los eventos en un log para garantizar un seguimiento adecuado.

Método para guardar y gestionar el uso de archivos de configuración y logs

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

El método guardarEnJsonCliente se encarga de gestionar la actualización de un archivo JSON con la información recibida de los clientes, utilizando un bloqueo (self.lock) para definir una zona crítica y asegurar que múltiples hilos puedan interactuar de manera segura con el archivo.

Si el archivo JSON ya existe, el método lo abre y verifica si la ruta especificada en el mensaje ya está registrada. Si la ruta existe, compara los archivos del mensaje con los ya almacenados, agregando únicamente los nuevos para mantener actualizada la información sin duplicados. En caso de que la ruta no esté registrada, la agrega como una nueva entrada junto con los archivos asociados.

En los casos en que el archivo JSON no existe, el método crea un nuevo archivo e inicializa su contenido con los datos del mensaje recibido. Cualquier evento relevante, como la adición de nuevas rutas o archivos, se registra en un log para facilitar el seguimiento. Asimismo, se maneja cualquier error que pueda ocurrir durante la lectura o escritura del archivo.

Por último, el método incluye un retardo de cinco minutos antes de realizar nuevas actualizaciones, lo que permite optimizar los recursos y reducir la frecuencia de acceso al archivo en entornos concurrentes.

Clase client

Constructor

Texto

Descripción generada automáticamente

El constructor de la clase Client configura un cliente para la comunicación mediante UDP. Inicializa un socket, obtiene el nombre del host local y define el puerto a utilizar para las conexiones.

Texto

Descripción generada automáticamenteEstablecer conexión con el servidor

Texto

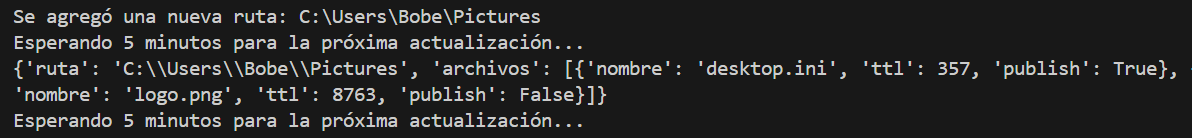
Descripción generada automáticamente

El método establecerConexion establece la conexión con el servidor y gestiona la interacción con el usuario para obtener información sobre una ruta de directorio. Primero, intenta conectar con el servidor usando el socket y, si la conexión es exitosa, pide al usuario que ingrese la ruta del directorio a consultar.

Luego, envía la ruta al servidor y espera la respuesta. Si la ruta es válida, verifica si hay archivos nuevos. En caso afirmativo, solicita al usuario el TTL para cada archivo y si desea publicarlos. Los datos se envían de vuelta al servidor. Si la ruta no es válida, solicita una nueva ruta.

El método maneja posibles errores como la falta de conexión con el servidor o la ausencia de respuesta, mostrando un mensaje adecuado al usuario en cada caso.

Ejecución

Texto

Descripción generada automáticamenteServidor

Texto

Descripción generada automáticamenteClientes

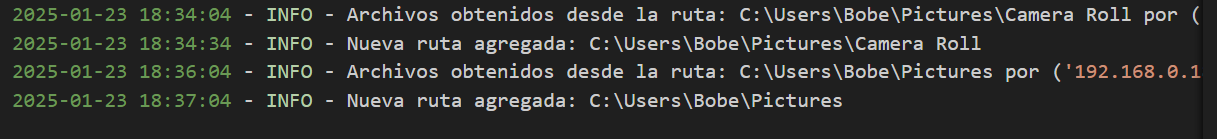
Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteTexto

Descripción generada automáticamenteArchivos de configuración

Archivo de historial .log

Conclusión

En conclusión, el programa establece una comunicación eficiente entre el cliente y el servidor mediante el uso de sockets UDP, permitiendo al cliente enviar solicitudes de validación de rutas y obtener información sobre los archivos disponibles. La interacción se maneja de manera dinámica, permitiendo al usuario ingresar rutas, recibir información sobre archivos y decidir sobre su publicación. Además, el programa está diseñado para manejar errores de conexión y validar la entrada del usuario, asegurando que el proceso sea robusto y estable.

La implementación de mecanismos de sincronización y control de errores proporciona una experiencia de usuario fluida y segura, garantizando que el intercambio de datos entre cliente y servidor se realice sin problemas. Dado el alcance actual, se considera que se han logrado los objetivos planteados inicialmente. En trabajos posteriores, se implementarán mecanismos para interactuar como servidor DNS y las estrategias necesarias para su implementación.