1. Introducción

Este documento describe los requisitos del servidor web desarrollado para la práctica 1 de la asignatura Redes 2. El servidor soporta HTTP 1.0 y 1.1, maneja peticiones GET, POST y OPTIONS, y es capaz de servir archivos estáticos y ejecutar scripts dinámicos.

2. Requisitos Funcionales

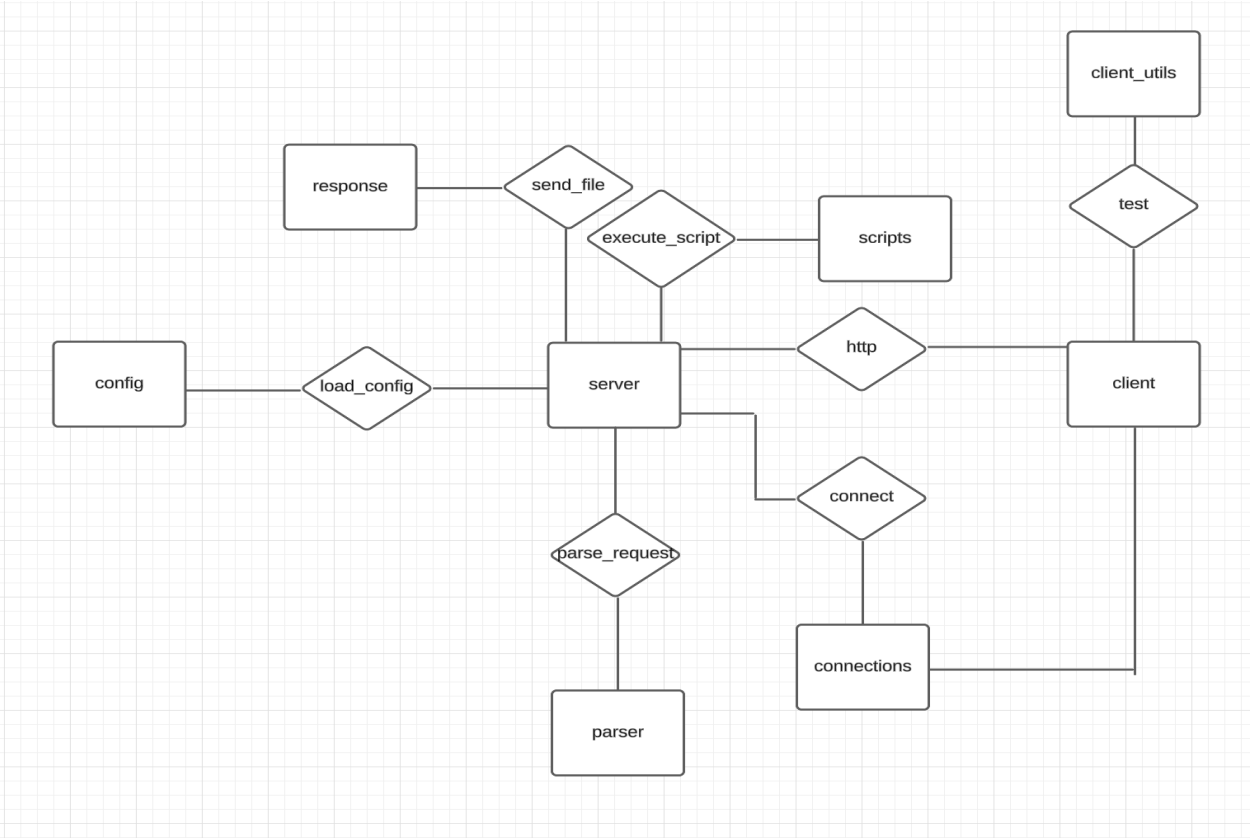
RF-1: El servidor debe aceptar conexiones entrantes en un puerto configurable.
RF-2: Debe procesar solicitudes HTTP de tipo GET, POST y OPTIONS.
RF-3: Debe servir archivos estáticos como HTML e imágenes.
RF-4: Debe ejecutar scripts en Python y PHP en respuesta a peticiones POST y GET.
RF-5: Debe permitir múltiples clientes simultáneamente mediante hilos.
RF-6: Debe cerrar conexiones correctamente cuando se reciba la señal SIGINT (Ctrl+C).
RF-7: Debe manejar correctamente errores como rutas no encontradas o métodos no soportados.
RF-8: Debe manejar tanto peticiones HTTP 1.0 como 1.1

3. Requisitos No Funcionales

RNF-1: El servidor debe estar implementado en C.
RNF-2: Debe ser ejecutable en entornos Linux.
RNF-3: Debe utilizar manejar la concurrencia entre varios clientes.
RNF-4: Debe manejar un límite configurable de clientes simultáneos.
RNF-5: Debe permitir pruebas con archivos multimedia y scripts de prueba.
RNF-6: Debe de no tener fallos de memoria.

Diagrama de Entidad-Relación (E/R)

Este diagrama modela los datos clave manejados por el servidor web:



Explicación del modelo:

- server: servidor que procesa peticiones http.
- client: cliente de prueba provisional.
- client_utils: diversas herramientas de las que se ayuda el cliente.
- connections: se encarga de la creación y conexión de sockets.
- parser: se encarga de parsear (traducir) las peticiones http.
- scripts: scrpits de prueba.
- response: encargado de la carga de imágenes.
- config: encargado de la configuración del servidor.

Casos de Uso

A continuación, se presentan los principales casos de uso del sistema.

CU-1: Servir archivo estático

Actor: Cliente (Navegador o herramienta como curl)
Descripción: Un cliente solicita un archivo estático como una imagen o un HTML.
Flujo principal:

El cliente envía una petición GET /archivo.html HTTP/1.1.
El servidor busca el archivo en el directorio raíz.
Si el archivo existe, el servidor lo devuelve con un código 200 OK.
Si el archivo no existe, el servidor responde con 404 Not Found.

CU-2: Ejecutar un script dinámico

Actor: Cliente
Descripción: El servidor ejecuta un script Python o PHP.
Flujo principal:

El cliente envía una petición GET /script.py?param=valor HTTP/1.1 o un POST con datos en el cuerpo.
El servidor identifica que se trata de un script.
El servidor ejecuta el script con los parámetros de la petición.
El resultado del script se devuelve como respuesta HTTP.

CU-3: Manejar múltiples clientes simultáneamente

Actor: Múltiples clientes
Descripción: El servidor maneja varias conexiones concurrentes.
Flujo principal:

Se establecen múltiples conexiones simultáneamente.
El servidor crea un hilo para cada cliente.
Cada cliente es atendido de forma independiente.
Cuando un cliente cierra la conexión, el servidor libera los recursos.

CU-4: Manejar errores HTTP

Actor: Cliente
Descripción: El servidor gestiona peticiones incorrectas.
Flujo principal:

Un cliente envía una petición con un método no permitido (PUT o DELETE).
El servidor devuelve 405 Method Not Allowed.
Si el archivo no se encuentra, se responde con 404 Not Found.