UNIDAD DE INGENIERÍA EN COMPUTACIÓN
Sede Central Cartago
Principios de Sistemas Operativos
Profesor: Kenneth Obando Rodríguez

TEC | Tecnológico de Costa Rica

Proyecto 1 (%) Fecha de Entrega: 4 de octubre, 2024

Implementación de Servidor HTTP

Descripción General

En este proyecto, los estudiantes deberán implementar un servidor HTTP v1.x funcional desde cero, utilizando uno de los lenguajes de programación propuestos: **Rust, C, C++ o Go**. El servidor debe soportar las principales operaciones HTTP (GET, PUT, POST, DELETE, UPDATE) y deberá manejar múltiples solicitudes concurrentemente utilizando hilos o procesos. Además, se debe implementar un sistema básico de gestión de cookies para manejar la sesión del usuario. Aunque no se requiere implementar HTTPS, se espera que el servidor sea lo suficientemente robusto para manejar varias conexiones simultáneamente y evitar bloqueos o colisiones de datos. Se realizará en los grupos de trabajo ya establecidos.

Objetivos del Proyecto

1. **Desarrollar un servidor HTTP** que soporte las principales operaciones HTTP: GET, PUT, POST, DELETE y UPDATE.
2. **Implementar concurrencia** utilizando hilos o procesos para manejar múltiples solicitudes de clientes de manera eficiente.
3. **Mitigar bloqueos y colisiones de datos**, implementando mecanismos adecuados de sincronización.
4. **Gestionar sesiones de usuarios** utilizando cookies para mantener un estado básico entre solicitudes.
5. Probar el servidor mediante herramientas como `curl` o `Postman`, asegurando su funcionalidad.
6. Documentar el código de manera clara y proporcionar pruebas unitarias que validen el correcto funcionamiento de cada funcionalidad implementada.

Requisitos Técnicos

1. **Lenguajes permitidos**: Rust, C, C++ o Go.
2. **Operaciones HTTP requeridas**:
    - **GET**: Recuperar recursos del servidor.
    - **POST**: Enviar datos para ser procesados o almacenados.

- **PUT**: Actualizar o crear recursos en el servidor.
- **DELETE**: Eliminar recursos específicos.
- **UPDATE**: Modificar parcialmente un recurso existente.

3. **Concurrencia**: El servidor debe manejar múltiples conexiones de clientes simultáneamente. Se debe implementar un modelo multihilo (threads) o multiproceso (processes) para manejar las solicitudes de manera eficiente.
4. **Manejo de cookies**: Se espera que el servidor gestione cookies para mantener un estado simple de sesión del usuario.
5. **No se requiere HTTPS**: Para simplificar el desarrollo, el protocolo seguro HTTPS no es necesario para este proyecto.
6. **Compatibilidad con herramientas de prueba**: El servidor debe ser probado utilizando `curl` o `Postman` para asegurar que las operaciones HTTP son correctas.
7. El servidor solo debe parsear archivos JSON y Texto plano, no es necesario que el servidor HTTP reciba archivos binarios.

Documentación

El proyecto debe ser acompañado por una **documentación detallada** que incluya:

- **Descripción del diseño del servidor**: Explicación de la arquitectura utilizada, cómo se gestionan las conexiones concurrentes y cómo se implementan las distintas operaciones HTTP.
- **Descripción de la implementación de la concurrencia**: Explicación de cómo se manejan los hilos o procesos y las técnicas utilizadas para mitigar bloqueos y evitar condiciones de carrera.
- **Manejo de cookies**: Detalle de cómo el servidor gestiona las cookies, incluyendo cómo se crean, almacenan y eliminan las mismas.
- **Instrucciones para ejecutar y probar** el servidor, indicando qué herramientas se utilizaron (`curl` o `Postman`) y los comandos necesarios para probar cada funcionalidad.
- **Estructura de directorios y archivos**: Explicar la disposición del proyecto, los archivos importantes y sus roles.

Pruebas Unitarias

Es indispensable que el código esté acompañado de pruebas unitarias para validar el correcto funcionamiento de las funcionalidades clave. Las pruebas unitarias deben cubrir y no solo:

- Operaciones HTTP (GET, POST, PUT, DELETE, UPDATE).
- Gestión de sesiones y cookies.
- Manejo concurrente de múltiples solicitudes.

Se espera un porcentaje de cobertura razonable (mínimo 70%) de las funciones principales.

Requerimientos de Entrega

1. **Código fuente** del servidor HTTP en el lenguaje elegido (Rust, C, C++ o Go).
2. **Documentación** detallada del proyecto, que explique la implementación y los pasos para ejecutar el servidor.
3. **Pruebas unitarias** que validen el comportamiento del servidor.
4. **Resultados de pruebas**: Capturas de pantalla o registros que muestren el uso de `curl` o `Postman` para probar las diferentes operaciones del servidor.
5. **Manual de usuario**: Guía para la instalación, ejecución y uso del servidor.

Debe presentar un archivo comprimido en el TecDigital antes de las 10:00pm del día de entrega. La actividad se puede realizar en parejas.

Si la entrega se realiza después de la hora de entrega, se le penalizará con 5 puntos porcentuales que se acumulan cada 24 horas. Por ejemplo si entrega a las 10:05pm su evaluación tendrá una nota base de 95%, si entrega después de las 10:05 p.m. del siguiente día, su nota base será 90%, y así sucesivamente.

Rúbrica de Evaluación

Criterio	Puntos
1. Funcionalidad General	
- Implementación correcta de las operaciones HTTP (GET, POST, PUT, DELETE, UPDATE)	15
- Manejo adecuado de concurrencia con hilos/procesos	15
- Gestión y uso correcto de cookies	5
2. Robustez del Servidor	
- Capacidad para manejar múltiples solicitudes concurrentes sin errores	15
- Mitigación de bloqueos y colisiones de datos	10
3. Documentación	
- Descripción clara y detallada del diseño e implementación	10
- Instrucciones precisas para ejecutar y probar el servidor	5
4. Pruebas Unitarias y Validación	
- Pruebas unitarias completas y bien documentadas	10
- Capturas de pruebas realizadas con `curl` o `Postman`	5
5. Buenas Prácticas de Programación	
- Código limpio y estructurado, siguiendo las convenciones del lenguaje	5
- Uso de técnicas adecuadas para manejar errores y excepciones	5
Total	100

Recomendaciones Finales

- **Uso de bibliotecas estándar**: Asegúrense de usar bibliotecas estándar para la implementación de sockets y manejo de hilos o procesos en el lenguaje que elijan.

- **Verificación continua**: Usen herramientas como `curl` o `Postman` para probar el servidor de manera incremental y asegúrense de que cada operación HTTP funcione correctamente antes de avanzar a la siguiente.