ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

SERVIDOR WEB CON RESPUESTA A PETICIONES HTTP DE IMÁGENES Y ARCHIVOS HTML

PRESENTADO POR:

NICOLÁS OSORIO ARIAS

05 DE SEPTIEMBRE DE 2018

BOGOTÁ D.C.

ARQUITECTURA SERVIDOR WEB

Cuando navegamos por internet, muchas veces omitimos lo increíble que puede llegar a ser su funcionamiento, ésta increíble red capaz de conectar al mundo ha creado un espacio virtual donde cualquier internauta puede explorar, obtener e intentar lo que desee con solo buscarlo en pocos segundos. En primera instancia esto puede sonar muy sencillo, sin embargo, involucra muchos más factores de lo que normalmente se pensaría; y obliga necesariamente a pensar en un orden y diversos protocolos que ayuden a estandarizar cada una de las partes que lo componen.

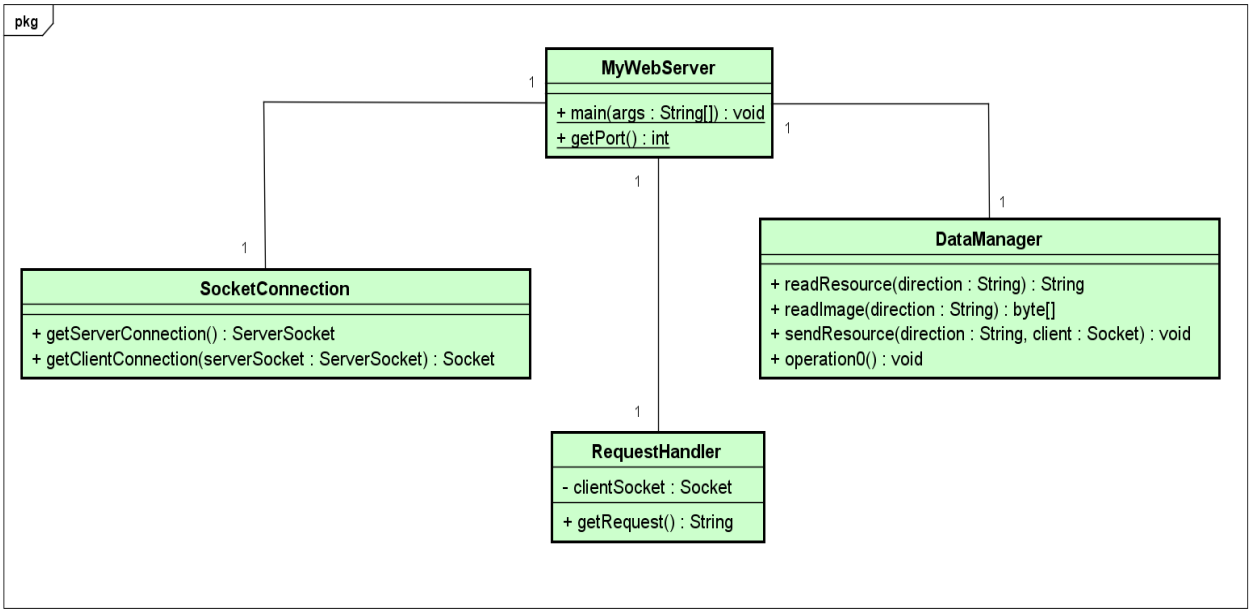
Los servidores web o servidores HTTP son los encargados de establecer una comunicación con los clientes y resolver todas sus peticiones; son los encargados de conectar esto es posible gracias a que normalmente se usa el protocolo HTTP lo cual hace que dos máquinas puedan hablarse entre sí con el mismo lenguaje. Estos servidores son el punto central de la red, ya que es gracias a estos que se distribuyen todos los sitios que se visitan normalmente, inclusive las imágenes, documentos, etc.

Explicando de forma más clara este concepto, se ha construido un Servidor Web en Java, el cual atiende múltiples solicitudes no concurrentes, y es capaz de entregar páginas .html e imágenes de tipo .png a sus clientes, logrando mostrar al público lo que se tenga guardado dentro del servidor.

Toda esta solución es posible gracias al uso de Sockets, los cuales funcionan como túneles en la comunicación que ayuda a que se pueda intercambiar cualquier flujo de datos entre diferentes máquinas.

La arquitectura de este Web Server se divide en tres partes principales, como primera medida, tenemos la encargada de establecer una conexión a un Socket, ya que es necesario para establecer un canal de comunicación entre el servidor y los clientes que los soliciten; en segunda instancia se tiene todo el sistema de obtención de peticiones GET, los cuales reportan al servidor solo aquellos recursos que sus clientes le piden por medio de estas peticiones. Finalmente sabiendo el recurso que está pidiendo el cliente, un administrador de archivos se encarga del proceso de búsqueda y envío del recurso al cliente para que se pueda completar su petición mostrándolo en pantalla.

Diagrama de clases:

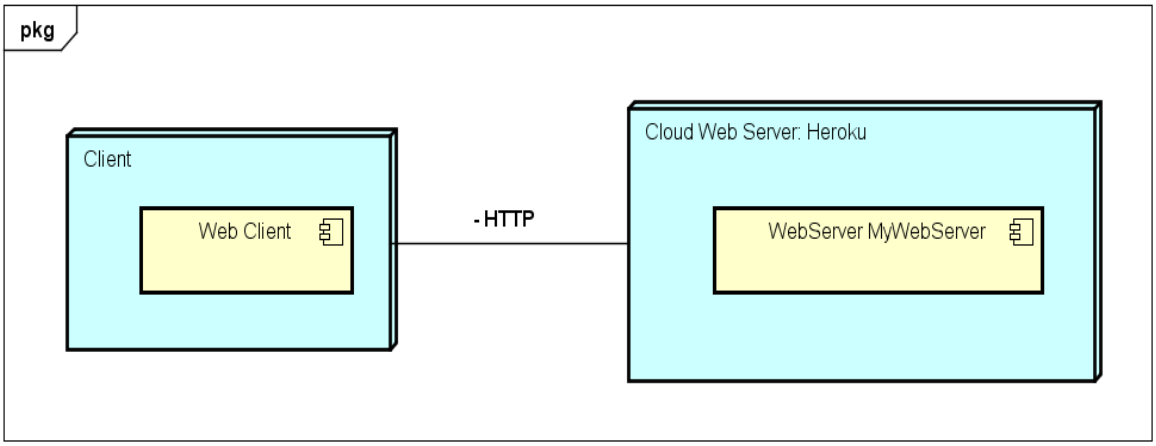


En el diagrama anterior se puede observar de mejor manera la arquitectura que este servidor Web tiene, el controlador principal del servidor es MyWebServer, el cual pone en operación al servidor, y conecta las diversas piezas para que estas puedan lograr un resolver la petición particular.

Lo primero que se hace cuando el servidor empieza a ejecutar es arrancar el proceso de conexión el cual empieza con SocketConnection, encargándose de brindar el socket por el cual el servidor establecerá conexión con sus clientes.

La parte de la recolección de las peticiones es llevada a cabo por RequestHandler, el cual por medio de getRequest retorna el Path de cualquier petición que elabore un cliente, MyWebServer se encarga de brindar esta ruta al DataManager, para que con su método sendResource pueda entrar a buscar dentro del servidor el recurso que el cliente ha pedido, y se lo logre enviar con el encabezado correspondiente.

Diagrama de despliegue:



Para que este servidor web funcione de manera pública y los clientes puedan venir desde lugares externos a los de la propia red es necesario hacer un despliegue del servidor y sea posible llevar el servidor a una dirección publica donde cualquier usuario pueda verla y comunicarse por medio de peticiones HTTP.

En la labor de despliegue, la herramienta utilizada fue Heroku, el cual guarda el servidor en contenedores virtuales que se ejecutan en un entorno de tiempo real, a lo cual llaman Dynos.

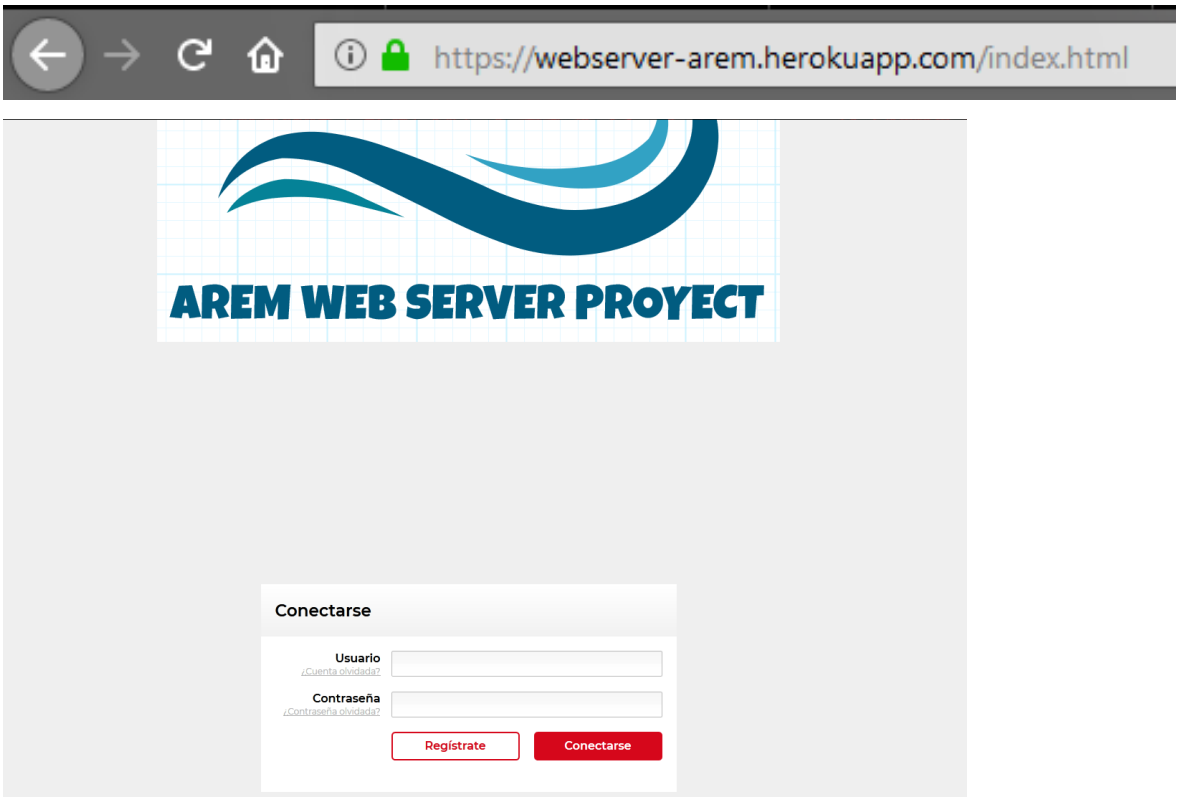
Para poder desplegar todo de manera correcta es necesario que el servidor hable por el socket que le brinda Heroku, y que los archivos que tiene el servidor estén guardados en rutas accesibles.

PRUEBAS ELABORADAS AL SERVIDOR WEB:

La primera prueba elaborada al servidor web fue verificar que el cliente pudiera comunicarse efectivamente y obtener un recurso del servidor, para esto se utiliza un archivo .png el cual al hacer la petición desde un browser logra obtener el siguiente resultado:

Con esto se verifica que la solución a la petición de imágenes se ha elaborado de forma correcta, sin embargo es necesario probar las peticiones a los archivos que se encuentran en texto, para esto, se utiliza una segunda prueba donde se devuelve una indicación al cliente de un error 404 que se encuentra en .html en caso de que el servidor no pueda solucionar su petición.

Con esta prueba se puede verificar que los archivos .html efectivamente pueden ser brindados a los clientes que lo requieran. Para la prueba final es necesario que se pueda evidenciar todo el proceso de petición y respuesta de un recurso .html que tiene una plantilla de Bootstrap la cual tiene relacionados archivos de diferentes formatos, además también se tendrá que mostrar la imagen anteriormente creada.



Finalmente, el servidor logra cumplir con las pruebas realizadas sobre este, permitiendo extenderlo tanto como se desee, aplicando diferentes plantillas personalizadas, añadiendo imágenes o archivos a este.

Como se pudo evidenciar los servidores web tienen puntos claves en la forma en la que se estructuran, uno de estos son los sockets, ya que sin ellos no sería posible efectuar la comunicación entre el cliente desde el browser y el servidor que lo atiende.

Hoy en día existen grandes servidores que hacen posible ver a internet como una red con todas las posibles utilidades que se le quieran dar, servidores como Apache o Microsoft IIS brindan soluciones mucho más avanzadas, pero con el mismo concepto que se hace este Web Server, una comunicación entre el cliente y su servidor, que pueda hacer las peticiones que requiera, y que nunca se preocupe por este proceso que sucede tan rápido, que lo hace sencillamente increíble.

TRABAJO FUTURO:

- Para un trabajo futuro es necesario hacer que el servidor pueda atender múltiples solicitudes concurrentes ya que si se solicitan varias peticiones al mismo tiempo es posible que ocurran problemas con los resultados, o que salgan errores inesperados.

- Se pueden crear nuevas formas para interactuar con el usuario, como carga y descarga de archivos, nuevos formatos disponibles, etc.

- En un futuro se puede llegar a refactorizar el código de una mejor manera para lograr eliminar los ciclos, por medio de hilos.