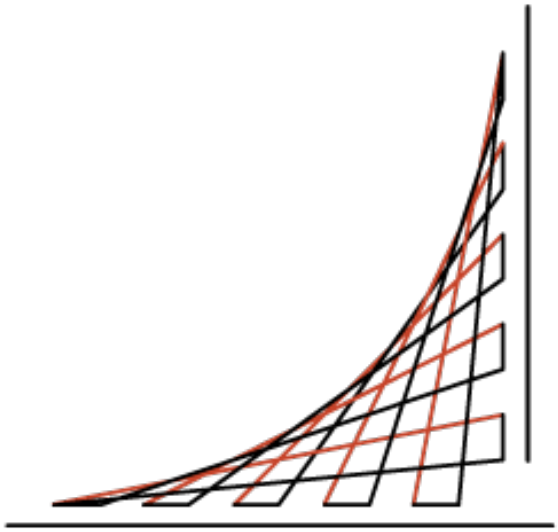

INGENIERÍA DE SISTEMAS - AREP 2020-2

Arquitecturas Empresariales

LABORATORIO 3 : Servidor Web

Luis Daniel Benavides Navarro

Authors:
Juan Camilo Rojas Ortiz

Índice

Resumen

Este artículo presenta la descripción de la teoría y la arquitectura utilizada para la creación de un servidor web capaz de responder solicitudes a archivos alojados en el servidor, tanto texto de extensiones html, css, js como imágenes de extensión jpg y png,además permite la creación de endpoints que respondan a peticions GET y POST, estos endpoints solo piden que el usuario especifique la ruta y la función que se ejecutará cuando se utilicen estos endpoints. Adicionalmente se construyó una aplicación web sencilla para la publicación de mensajes con el objetivo de probar el funcionamiento del servidor construido. Tanto la arquitectura del servidor y de la aplicación construida son explicadas en el documento. Se aporta con un framework que permite al usuario construir aplicaciones web simples con los verbos GET y POST y archivos estáticos comunes, este desarrollo es extensible y permitiría implementar los otros verbos HTTP fácilmente

1. Introducción

Actualmente la infraestructura en la nube ha tomado gran fuerza dado que permite acceder a recursos computacionales muy potentes a precios accesibles y permitiendo que el usuario se evite los esfuerzos de instalación, mantenimiento y compra de estos equipos.Consecuencia de esto, muchas compañías pueden diseñar sus aplicaciones web y llevarlas a un entorno de producción en mucho menos tiempo, sin embargo muchos de los frameworks usados actualmente tienden a ser pesados, por lo que, con el esfuerzo de desarrollo necesario se podría desarrollar un framework que permita solo las operaciones necesitadas por el desarrollador,si bien esta solución permite aligerar el framework e incluir los comportamientos deseados, puede llegar a tomar mucho esfuerzo y tiempo de desarrollo.
El objetivo de este laboratorio es desarrollar la lógica de un servidor web capaz de responder solicitudes a archivos alojados en el servidor, tanto texto de extensiones html,css,js como imágenes de extensión jpg y png,además permite la creación de endpoints que respondan a peticions GET y POST, estos endpoints deben permitir el uso de funciones especificadas por el desarrollador encargado de crear la aplicación web.
Para explicar la arquitectura del sistema se explicarán los componentes usados para el funcionamiento del servidor y luego los componentes de la aplicación desarrollada para probar el servidor, que tiene conexión con una base de datos mongodb y está desplegada en heroku

2. Arquitectura del servidor

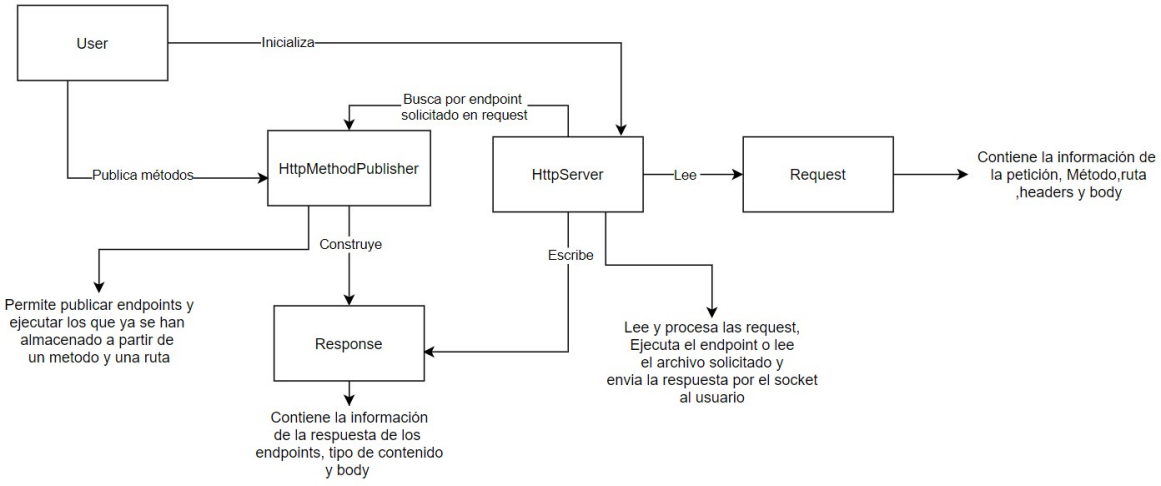

Figura 1: Arquitectura servidor web

En la figura 1 podemos ver la arquitectura del servidor web implementado, a continuación se derá una breve explicación de la función de cada uno de estos componentes.

- User: Es un componente externo, se refiere al programador que va a utilizar el servidor para construir su aplicación web, debe encargarse de inicializar el hilo del servidor y posteriormente publicar los métodos que necesite, para publicar un método debe especificar la ruta y escribir la función lambda que ejecutará el endpoint, esta función cuenta con un objeto Request y un objeto Response como parámetros, y debe retornar un String
- Request: Contiene toda la información de las peticiones, método,ruta y encabezados.
- Response: Permite al usuario establecer el tipo de retorno, el tipo por defecto es text/html, también se encarga de almacenar el valor retornado por la función especificada por el usuario
- HttpMethodPublisher: Encargado de almacenar los endpoints del usuario, permite publicar endpoints de tipo GET y POST, también es el encargado de buscar un endpoint y ejecutarlo dado un request.
- HttpServer: Componente principal, se encarga de crear un socket entre el servidor y el cliente, leer las peticiones que se generen a través de este, luego procesarlas buscando el endpoint solicitado o buscando el archivo estático necesario , y finalmente escribe la respuesta en el socket enviando los encabezados necesarios para que el cliente la pueda recibir satisfactoriamente. Permite múltiples peticiones no concurrentes y puede leer archivos de tipo js,html,css,png y jpg.

3. Arquitectura de la aplicación

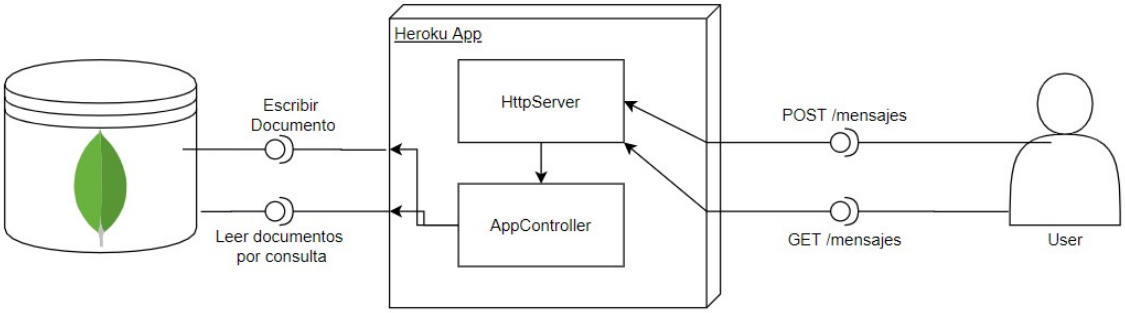

Figura 2: Arquitectura aplicación

La arquitectura general de la aplicación se puede observar en la figura 2 y consta de un controlador que publica los endpoints GET /mensajes y POST /mensajes sobre una instancia del servidor web implementado, estos endpoints se encargan de enviar al cliente los mensajes que se han almacenado hasta ahora y permitirle publicar un nuevo mensaje respectivamente. Para poder realizar esto el controlador usa una capa de servicios que a su vez se apoya en una capa de persistencia,que es implementada por una conexión a una base de datos mongoDB que permite almacenar los mensajes publicados por medio de documentos, cada documento consta de un id, la fecha y el contenido del mensaje. Para comunicar las distintas capas se utilizó el principio de inversión de como se puede ver en la figura 3

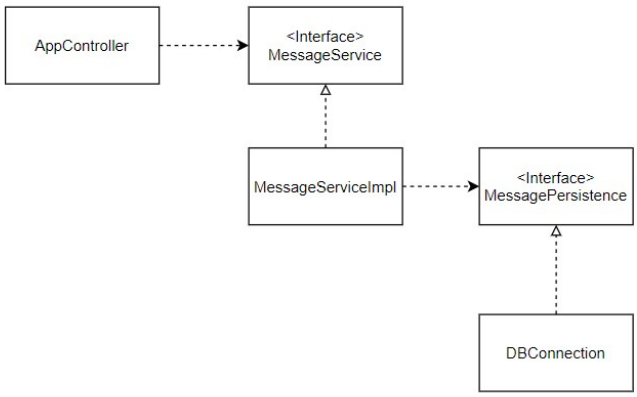

Figura 3: Arquitectura interna

4. Pruebas

Para verificar el correcto funcionamiento del servidor y de la aplicación se realizaron unas peticiones usando Postman.Primero se realiza una petición GET al endpoint /mensajes, el cual retorna una lista de mensajes como se ve en la figura 4 luego se realiza un POST a este mismo endpoint que permite registrar un nuevo mensaje ("mensaje de prueba") como se ve en la figura 5, finalmente se realiza otra petición GET para verificar que el mensaje se publicó correctamente, el resultado se puede observar en la figura 6

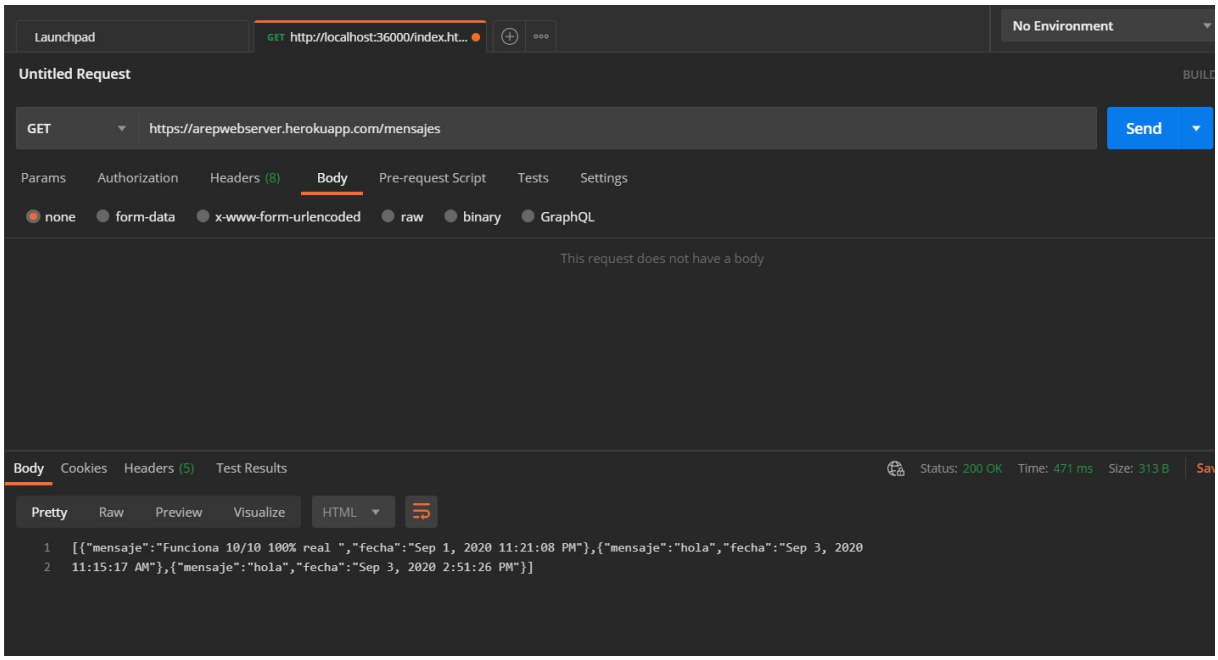

Figura 4: primer GET /mensajes

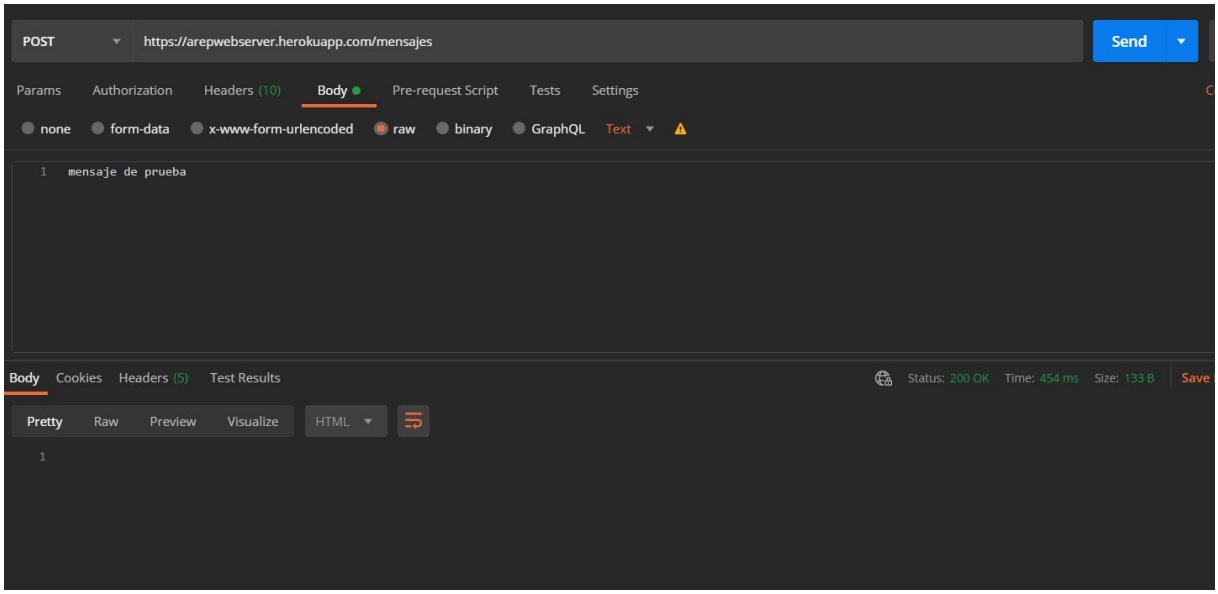

Figura 5: POST /mensajes

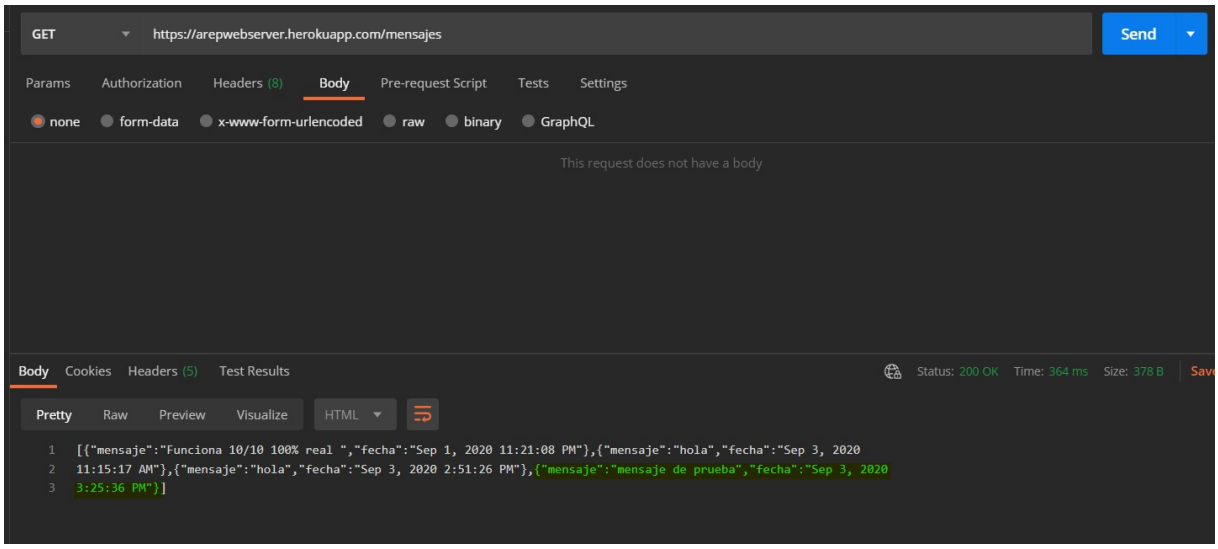

Figura 6: segundo GET /mensajes

Para probar las peticiones a archivos estáticos se repite el ejercicio desde Postman a los archivos index.html y logoesc.png como se ve en las figuras 7 y 8

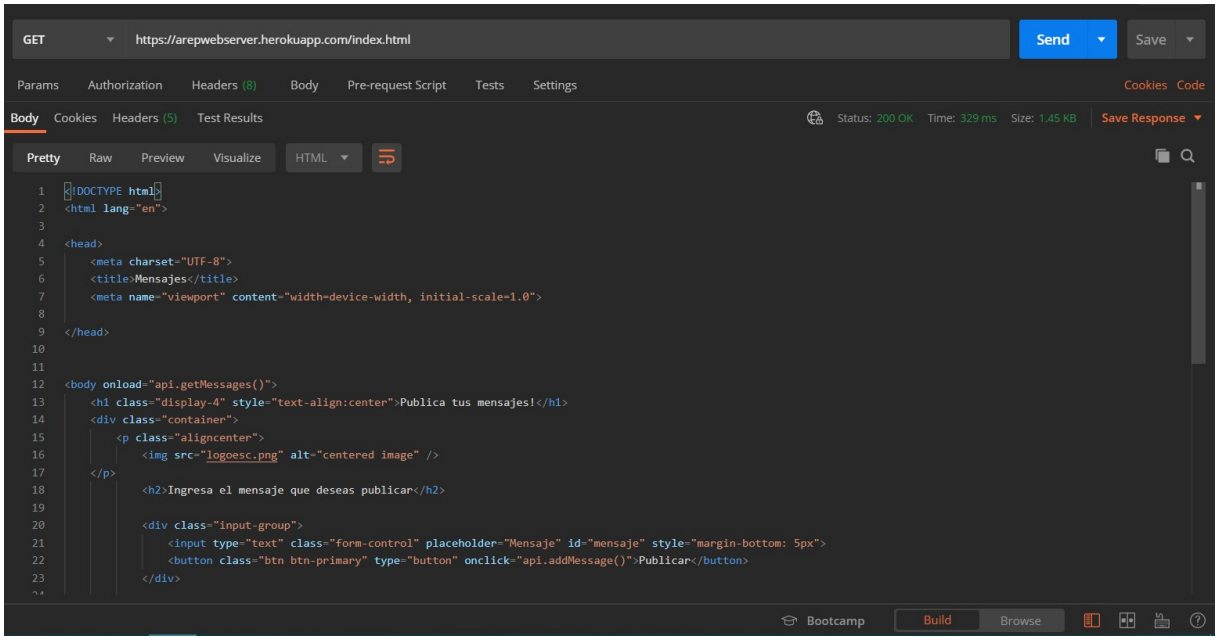

Figura 7: Petición a recurso index.html

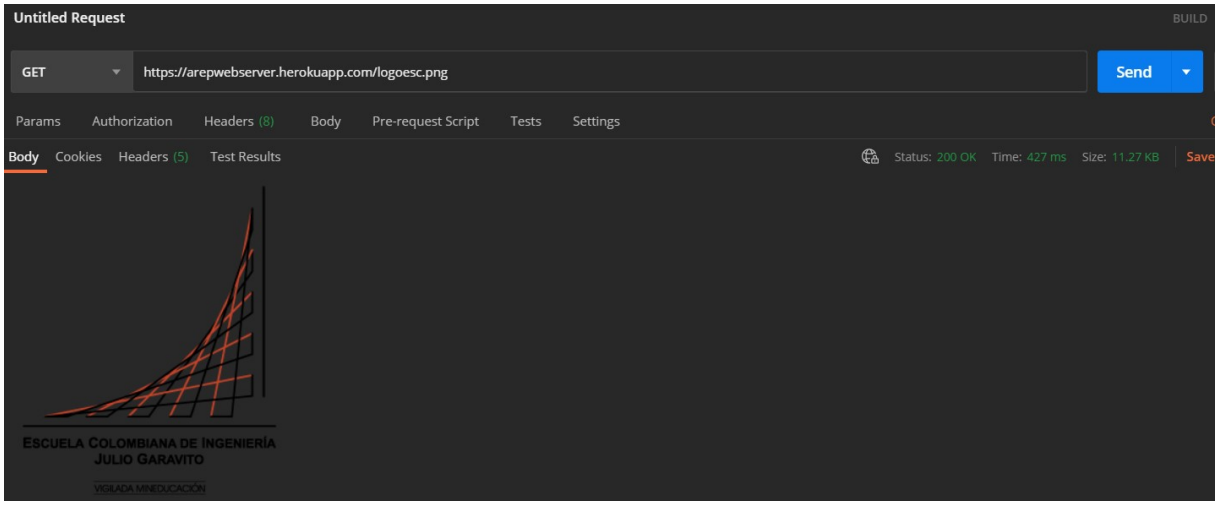

Figura 8: Petición a recurso logoesc.png

5. Resultado

Se entrega una implementación de un servidor web que permite la creación de endpoints de tipo GET y POST ,además de permitir la transferencia de archivos estáticos de tipo .css, .js, html, .jpg y .png

.También se construye una aplicación web a partir del servidor implementado, esta aplicación se apoya en una base de datos MongoDB para permitir la lectura y publicación de mensajes.

6. Referencias