# Construcción de un servidor web en Java (septiembre 2019)

Duran Vivas, Karen Paola

***Resumen*—En el siguiente artículo se evidencia la construcción de un servidor Web (tipo Apache) en Java. El servidor debe ser capaz de entregar páginas html e imágenes tipo PNG. Igualmente, el servidor debe proveer un framework IoC para la construcción de aplicaciones web a partir de POJOS. Usando el servidor se debe construir una aplicación Web de ejemplo y desplegarlo en Heroku. El servidor debe atender múltiples solicitudes no concurrentes.**

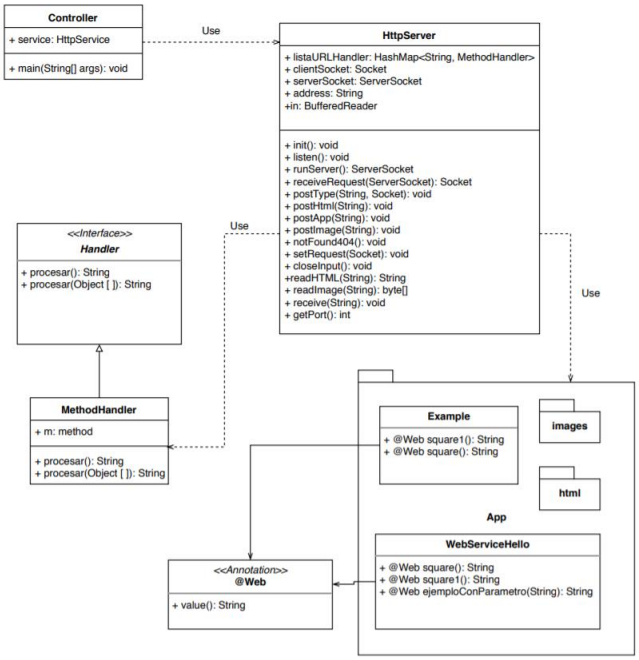
***Palabras Clave*—Servidor web, POJOS**

## I. Introducción

El siguiente artículo describe la arquitectura de la implementación de un servidor no concurrente desarrollado en Java y desplegado en Heroku, el cual es capaz de entregar páginas HTML e imágenes en formato PNG, además cuenta con un framework para la construcción de aplicaciones web a partir de POJOS (Plain Old Java Object). La usabilidad de este software está dada por peticiones HTTP/GET, es decir, se le solicitarán recursos ya predefinidos, y en dado caso que este recurso no exista se le notificará al usuario con una página de error. El ciclo de vida de este proyecto está estructurado y manejado en Maven.

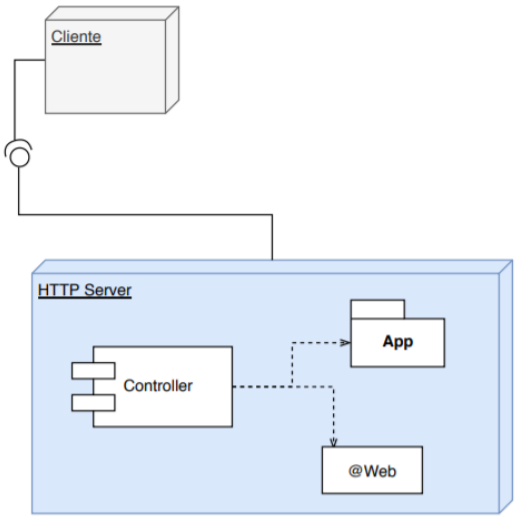
## II. Diagrama de Clases

En el diagrama de clases podemos observar las relaciones que tienen unas clases con otras, para este ejercicio se creó una clase controlador que es la que invoca los métodos de inicialización y escucha al servidor HTTP, este se ejecuta por el puerto 4567 (localhost). Dentro de la clase del servidor HTTP, se tienen métodos específicos de lectura hacia cada una de las clases .java que existen en el paquete /app, esto se hace con el fin de encontrar cada uno de los métodos de estas clases que contengan la anotación @Web y de esta forma poder cargar todos los POJOS. Para poder acceder a cada uno de los recursos ubicados en la carpeta /app, es necesario conocer el nombre del recurso que se quiere obtener; en caso de que no sea ni .HTML, ni .PNG, será necesario escribir en la URL /app/recurso, en caso de que tenga algún parámetro se escribirá de la siguiente manera .../app/recurso/parámetro.




*[1] Diagrama de clases.*

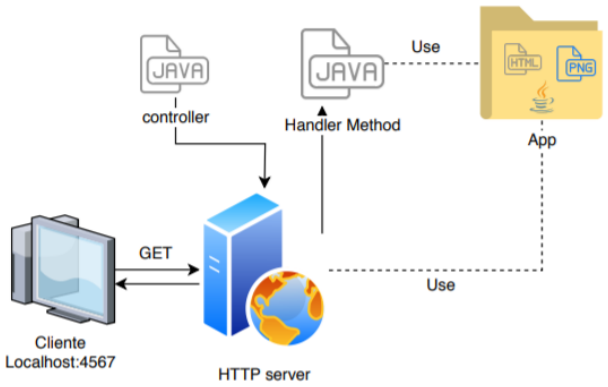
## III. Diagrama de Componentes



*[2] Diagrama de componentes.*

En el diagrama de componentes se puede apreciar como el cliente interactúa con el servidor a través de la interfaz que este provee por medio de peticiones HTTP.

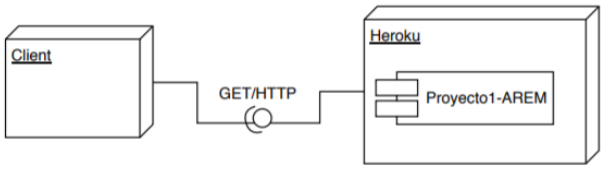
## IV. Diagrama de Arquitectura




*[3] Diagrama de arquitectura.*

En este diagrama podemos observar varios de los componentes anteriormente mencionados y la interacción entre ellos. Tenemos un servidor HTTP que es quien se encarga de gestionar las peticiones que el cliente le realiza solicitándole recursos ya predefinidos, la carpeta app que es donde se encuentran almacenados todos los recursos para este ejercicio, en esta carpeta las clases .java que se almacenan utilizan la anotación @Web en sus métodos de tal forma que se pueda acceder a ellas mediante /app/recurso, y por último también se tiene un manejador de recursos.

## V. Diagrama de Despliegue

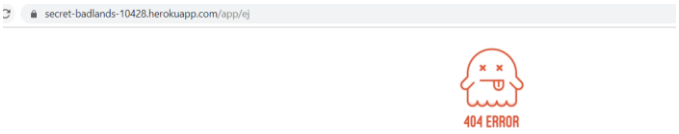



*[4] Diagrama de despliegue.*

El despliegue se realizó en heroku cumpliendo con uno de los requisitos de la entrega del proyecto, en este diagrama se tienen sólo dos componentes, uno de ellos es el servidor de heroku que es en donde se encuentra almacenado el servicio web que se desarrolló y el otro elemento es el cliente que mediante peticiones GET/HTTP solicita recursos.

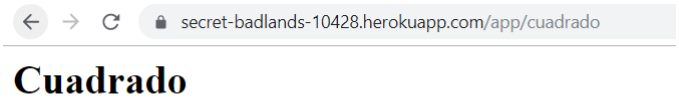
## VI. Resultados

Las imágenes que se observan a continuación son ejemplos de los resultados que usted puede obtener utilizando distintas rutas ya predefinidas, ya sea para obtener un archivo con extensión HTML o PNG o simplemente recursos de /app.

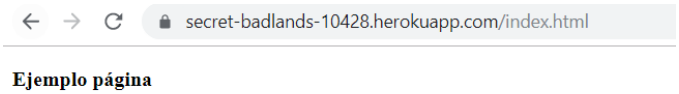



*[5] Resultado de solicitar un recurso inexistente.*

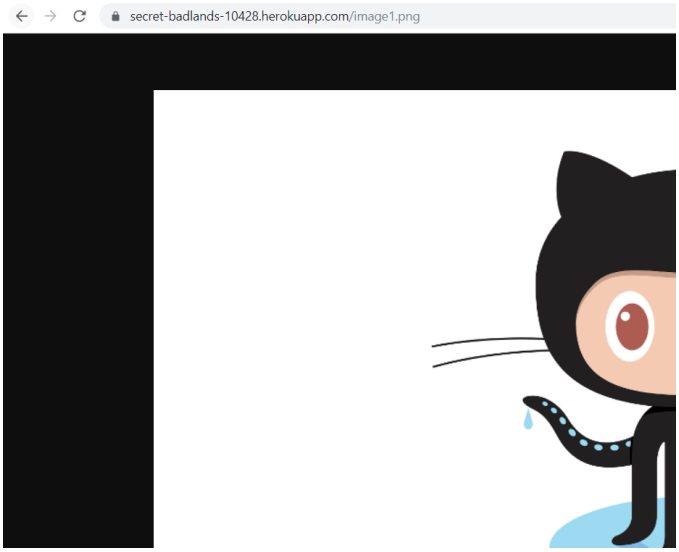

*[6]Resultado del recurso ../app/ejemplo:karen, Karen es un parámetro y por ende puede cambiarse por cualquier otro.*




*[7] Resultado de solicitar el recurso ../app/cuadrado.*




*[8]Resultado de solicitar un HTML.*




*[9] Resultado de solicitar una imagen en formato PNG.*

## VII. Conclusiones

Este proyecto fue útil para aprender a desplegar un servidor web en Java y además desplegarlo en Heroku, se tuvo un acercamiento hacia los que son los POJOS y se aprendió a usarlos durante la construcción del proyecto.