Laboratorio No. 2 - Taller Clientes y Servicios

Angie Tatiana Medina Gil

September 2021

1 Introduction

En el presente documento se explicará el diseño que se siguió para lograr la implementación de un servidor HTTP, este responderá a solicitudes realizadas a través del método GET haciendo que el browser (el cliente web) despliegue el recurso estático consultado (actualmente se manejan archivos html, javascript, css e imágenes)

2 Objetivos

Desarrollar una arquitectura correcta siguiendo el modelo de cliente/servidor Lograr que el servidor HTTP soporte solicitudes seguidas(no concurrentes) Desplegar la aplicación en Heroku para la satisfactoria visualización del producto sin necesidad de almacenarlo en disco Desplegar archivos estáticos con extensiones html, js, css e imágenes

3 Marco Teórico

- **Git:** Es un sistema de control de versión distribuida, lo que quiere decir que la base del código entero y su historial se encuentran disponibles en la computadora de todo desarrollador, lo cual permite un fácil acceso a las bifurcaciones y fusiones.
- **Github:** Es una compañía sin fines de lucro que ofrece un servicio de hosting de repositorios almacenados en la nube. Esencialmente, hace que sea más fácil para individuos y equipos usar Git como la versión de control y colaboración.
- **Maven:** Es un framework de gestión de proyectos de software, que proporciona un modelo estándar de gestión y descripción de proyectos.
- **Java:** Es un lenguaje de programación y una plataforma informática que fue comercializada por primera vez en 1995 por Sun Microsystems. Hay muchas aplicaciones y sitios web que no funcionarán, probablemente, a menos que tengan Java instalado y cada día se crean más.

- **Heroku:** Es esa plataforma. En pocas palabras, es un servicio de los que conocemos como PaaS, Platform as a Service (Plataforma como Servicio en español), ayuda a los desarrolladores, a ejecutar y mantener sus proyectos sin preocuparse mucho por la gestión de la infraestructura incluyendo las bases de datos, seguridad, networking, logging y monitoreo. Esto lo hace a través de un equipo que pone a disposición de quien contrate sus servicios, permitiéndote enfocarte en lo que de verdad te importa y te genera valor, tu proyecto de software.

- **HTTP:** Es un protocolo de la capa de aplicación para la transmisión de documentos hipermedia, como HTML. Fue diseñado para la comunicación entre los navegadores y servidores web, aunque puede ser utilizado para otros propósitos también. Sigue el clásico modelo cliente-servidor, en el que un cliente establece una conexión, realizando una petición a un servidor y espera una respuesta del mismo.

- **GET:** El método GET envía la información codificada del usuario en el header del HTTP request, directamente en la URL. La página web y la información codificada se separan por un interrogante ?, este envía la información en la propia URL, estando limitada a 2000 caracteres, la información es visible por lo que con este método nunca se envía información sensible.

- **Cliente/Servidor:** Se llama cliente al dispositivo que requiere ciertos servicios a un servidor. La idea de servidor, por su parte, alude al equipo que brinda servicios a las computadoras (ordenadores) que se hallan conectadas con él mediante una red. El concepto refiere por lo tanto a un modelo de comunicación que vincula a varios dispositivos informáticos a través de una red. El cliente, en este marco, realiza peticiones de servicios al servidor, que se encarga de satisfacer dichos requerimientos.

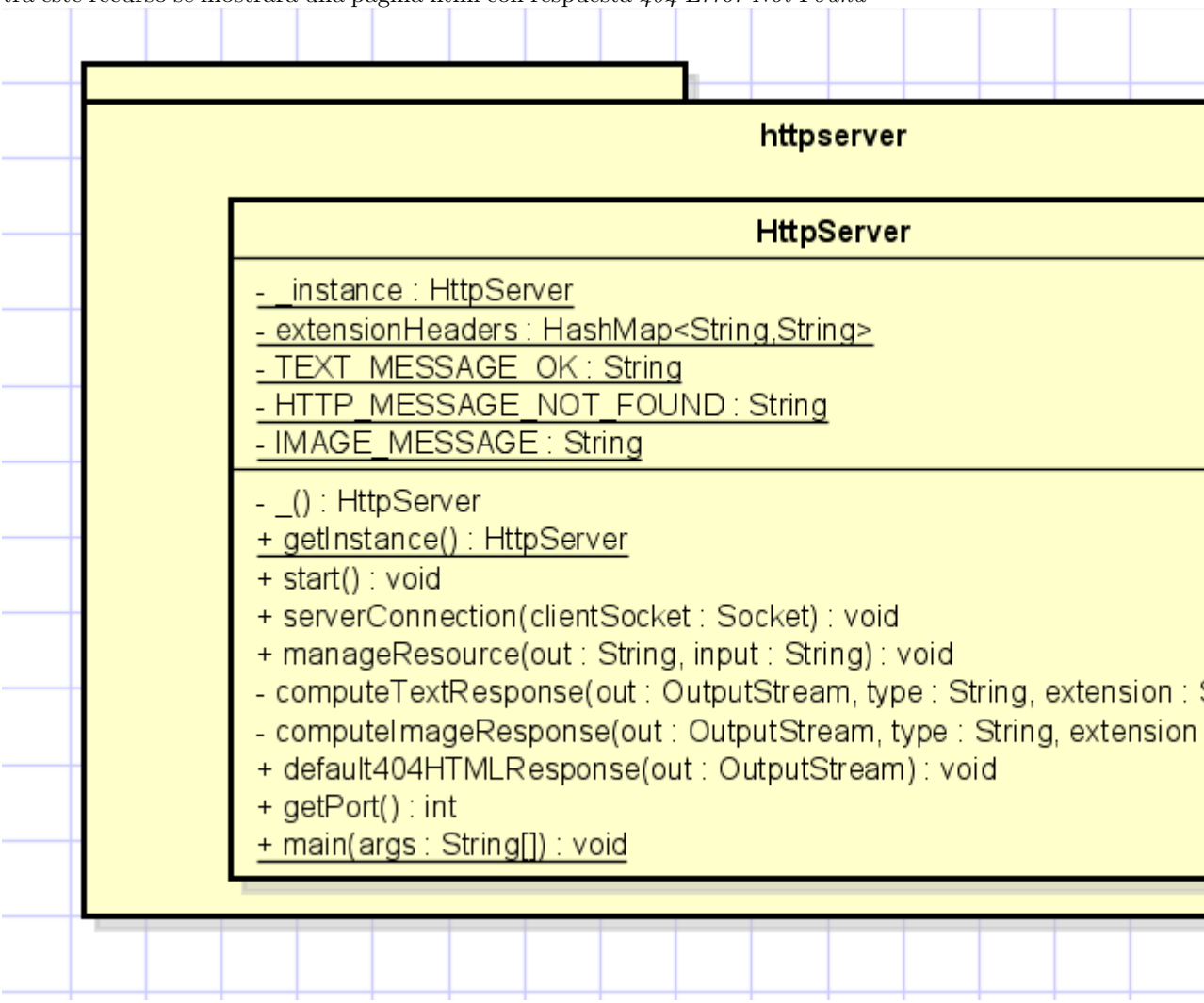
4 Arquitectura

Para el presente laboratorio se siguió un modelo Cliente/Servidor en el cual

4.1 Servidor HTTP

Es la parte central del desarrollo, para este se implementaron a grandes rasgos dos características: la primera de esas consiste en la conexión y gestión de los mensajes del protocolo HTTP lo cual incluye la creación y activación de sockets, recibir e interpretar la información provista por el protocolo. Para la segunda parte una vez identificado la instrucción GET junto con el recurso estático se decide si el encabezado de este debe tener como content-type text o image. En el caso en que se considere que es de tipo text se busca el recurso y en caso de no encontrarlo se mandara al cliente una página html con respuesta *404 Error Not Found* y si por otro lado el recurso existe se pondrá el *content-type*

correspondiente se procederá a leer el archivo para finalmente escribir sobre el Stream. En el caso de que sea una imagen se deberá leer la imagen como un arreglo de bytes para luego escribirlo sobre el Stream el cual junto con el browser se encargara de interpretarlo como una imagen, y así mismo si no se encuentra este recurso se mostrara una página html con respuesta *404 Error Not Found*



5 Recursos

Los recursos estaticos que son provistos actualmene por el servidor son:

- /resources/static/Index.html (Página principal del programa)
- /resources/static/Page.html

- /resources/static/Script.js
- /resources/static/ScriptPage.js
- /resources/static/Styles.css
- /resources/static/StylesPage.css
- /resources/static/img/Eevee.png

6 Conclusiones

Para poder implementar un servidor web con Java puro que sea funcional es necesario tener varios conceptos claros (sobre todo los de red), ya que son herramientas que facilitan el manejo de mensajes que se envían y reciben por la red.

7 Bibliografia

- https://es.wikipedia.org/wiki/Git
- https://kinsta.com/es/base-de-conocimiento/que-es-github/
- http://www.juntadeandalucia.es/servicios/madeja/contenido/recurso/322
- https://openwebinars.net/blog/introduccion-heroku/
- https://diego.com.es/get-y-post-en-php
- https://definicion.de/cliente-servidor/