Taller De Arquitecturas De Servidores De Aplicaciones, Meta Protocolos De Objetos, Patrón IOC y Reflexión

Daniel Felipe Walteros Trujillo

12 de Febrero del 2021

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

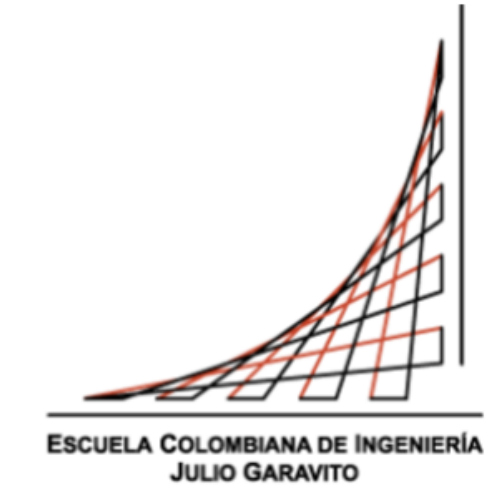


Tabla de Contenido

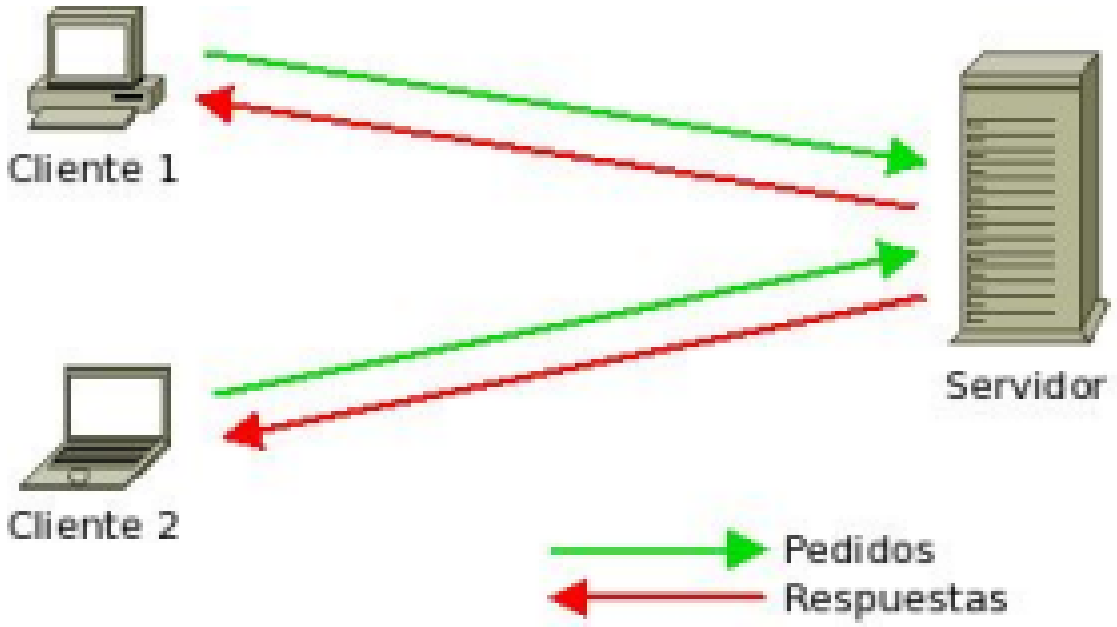
1 Prerrequisitos

Para el desarrollo del programa se utilizó Maven como una herramienta para la gestión del ciclo de vida del software, el código fue desarrollado con el lenguaje de programación Java, por lo tanto para su ejecución se requiere:

- Java versión 8 o superior.
- Maven versión 3.5 o superior.

2 Introducción

El modelo Cliente/Servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Las aplicaciones Clientes realizan peticiones a una o varias aplicaciones Servidores, que deben encontrarse en ejecución para atender dichas demandas.[2]



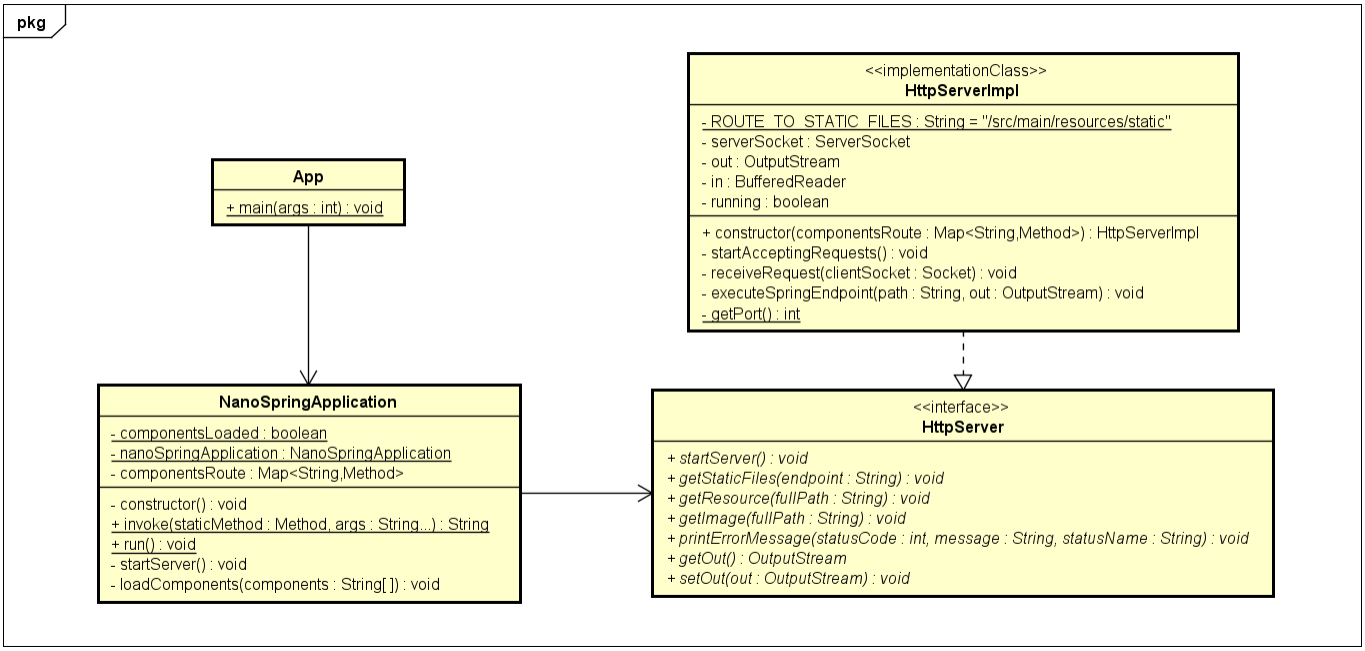
En este trabajo, se desarrolló un servidor Web (tipo Apache) en Java con la capacidad de entregar páginas html e imágenes tipo PNG.

Este servidor utiliza un framework IoC de implementación propia para la construcción de aplicaciones web por medio de POJOS, se encuentra desplegado en Heroku y atiende múltiples solicitudes no concurrentes.

Esta aplicación es un prototipo mínimo que demuestra las capacidades reflexivas de JAVA y permite cargar varios beans (POJOS) de los cuales se derivr una aplicación Web.

3 Diseño

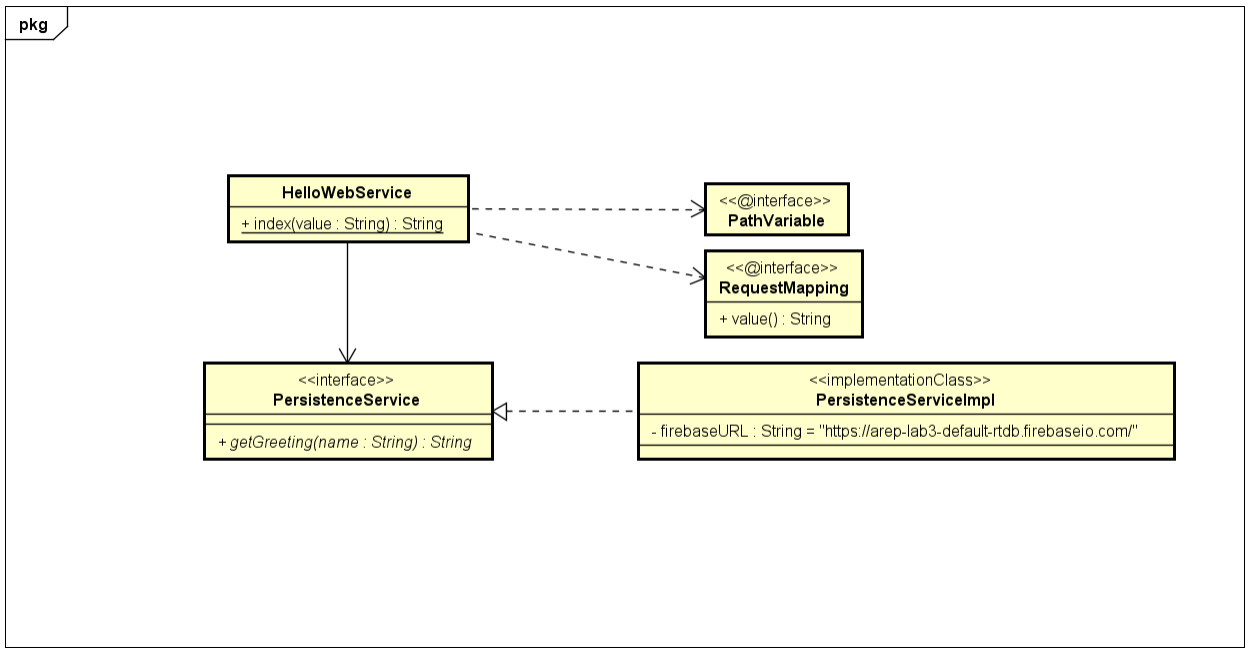
3.1 Diagrama de Clases del Framework



El programa principal utiliza la clase NanoSpringApplication, esta clase por medio de reflexión[3] carga los componentes y su maneja su ejecución por medio de la interfaz HttpServer, su implementación crea por medio de sockets un servidor sobre el cual corre una aplicación web.

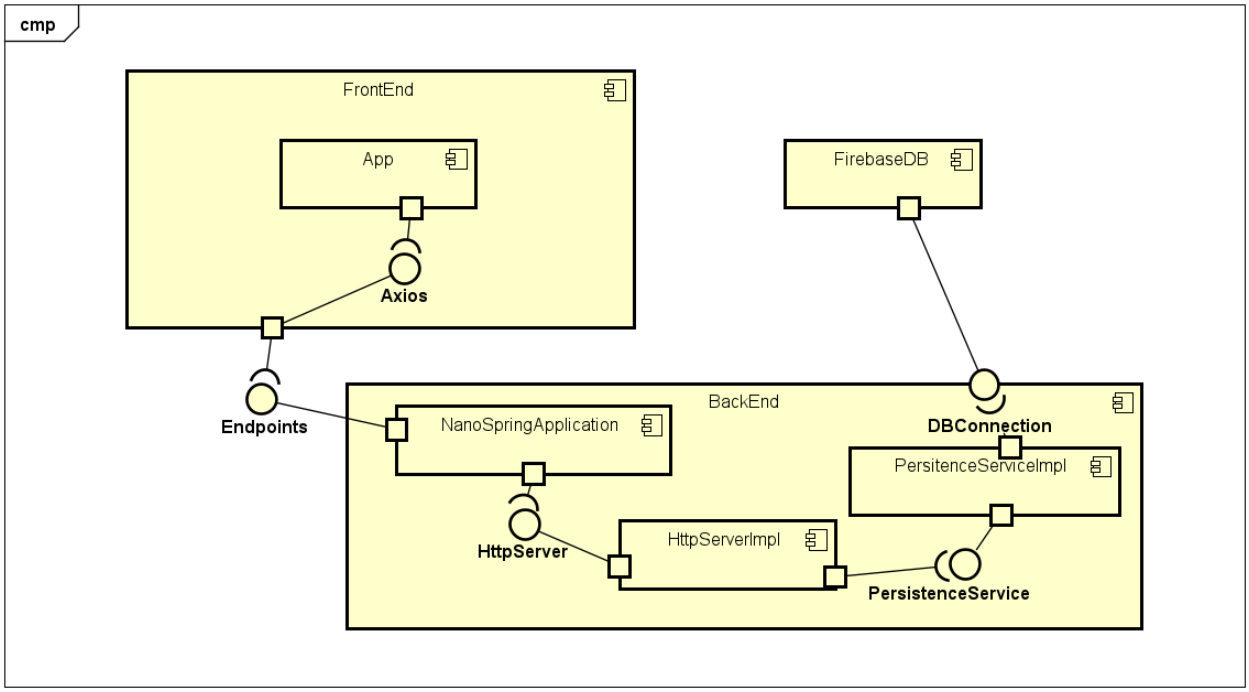
La clase NanoSpringApplication utiliza el método run para cargar los componentes recibidos en los argumentos del main usando reflexión, esto hace que no haya necesidad de modificar el codigo del framework para incorporar nuevos endpoints siempre y cuando utilicen la anotación RequestMapping[4] en el método del nuevo endpoint y la anotación PathVariable[1] en caso de que el path necesite variables.

3.2 Diagrama de Clases de la Aplicación



Para probar el framework se creo un servicio simple llamado HelloWebService que utiliza las anotaciones previas y además por medio de la interfaz PersistenceService accede a una base de datos, su implemenración se conecta a una base de datos Firebase desarrollada para probar en tiempo real la conexión.

3.3 Diagrama de Componentes

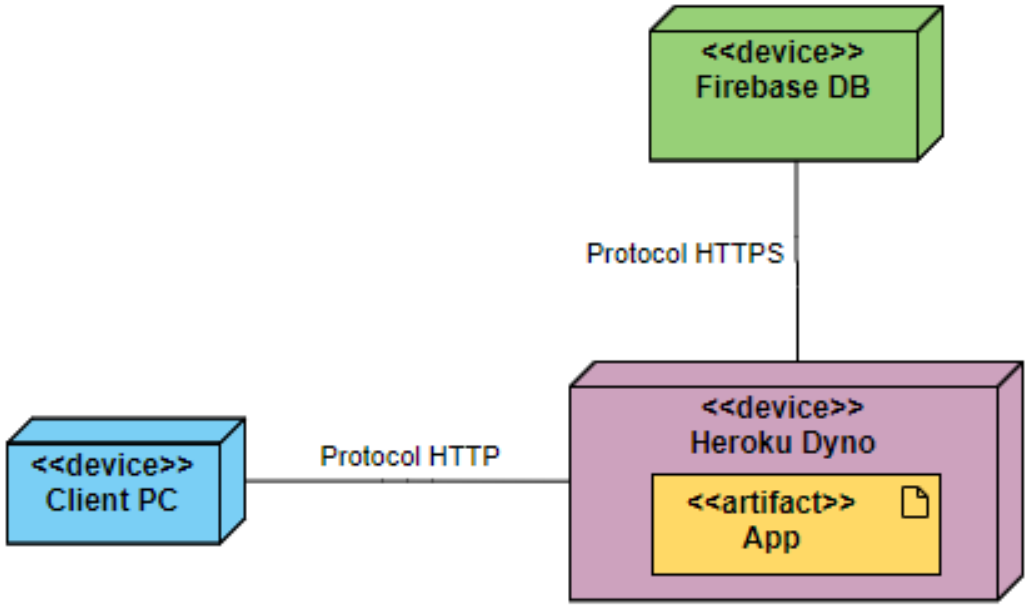


La aplicación se divide en tres componentes principales, FrontEnd, BackEnd y FirebaseDB.

El componente más funcional de FrontEnd es App, este es el que lee la información que registra el usuario y por medio de Axios accede a NanoSpringApplication para usar el endpoint definido en NanoSpring.

Este endpoint se configuró una conexión con el componente PersistenceServiceImpl, este se conecta la base de datos Firebase para obtener el saludo que retorna junto con el nombre del usuario.

3.4 Diagrama de Despliegue

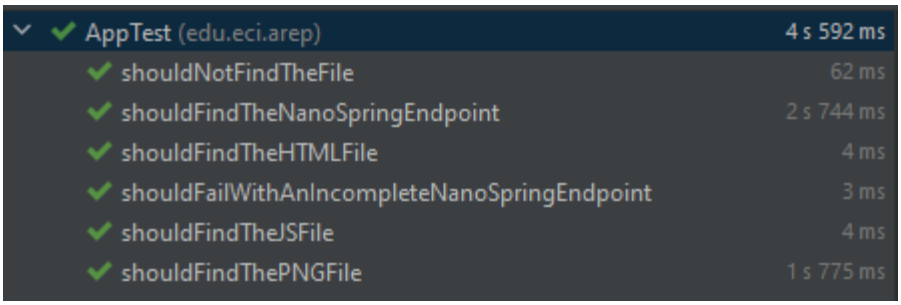


Debido a que utilizando que la aplicación desarrollada intenta emular el comportamiento de una aplicación web realizada con Spring, cualquier persona con conexión a internet puede acceder a la aplicación desplegada, la interacción del cliente con el servidor se realiza únicamente por el protocolo HTTP; por otra parte, la conexión del servidor con la base de datos firebase se realiza por el protocolo HTTPS.

4 Pruebas

El programa fue probado con seis pruebas unitarias de JUnit donde se contemplaron los siguientes casos:

- Búsqueda de un archivo HTML.
- Búsqueda de un archivo PNG.
- Búsqueda de un archivo JS.
- Búsqueda de un archivo inexistente.
- Uso de Endpoint Generado Con NanoSpring.
- Fallo Por Uso Erróneo de Endpoint Generado Con NanoSpring.



5 Conclusiones

- El programa carga efectivamente los archivos HTML,JS y PNG almacenados en disco.
- El uso de interfaces para generalizar comportamientos dentro de la aplicación permite la extensión o cambio de código sin la necesidad de alterar múltiples archivos.
- El despliegue de una aplicación web por medio de Heroku permite el uso comercial de la misma en todos los sitios que tengan conexión a Internet sin incurrir en altos costos por administración y soporte de servidores, esto se debe a que esta plataforma utiliza tecnologías en la nube.
- El desarrollo de NanoSpring por medio del mecanismo de reflexión de Java[3] permite crear programas completamente independientes de sus componentes, al conectarse por medio de anotaciones permiten un desarrollo de software extensible y muy eficaz.

6 Referencias

[1] baeldung. *Spring @Pathvariable*. URL: https://www.baeldung.com/spring-pathvariable. (entered: 01-01-2021).

[2] Ing. Emiliano Marini. *El Modelo Cliente/Servidor*. URL: https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf. (entered: 01-10-2012).

[3] Ira R. Forman y Nate Forman. *Java Reflection In Action*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.116.5796&rep=rep1&type=pdf. (entered: 10-02-2021).

[4] Eugen Paraschiv. *Spring @RequestMapping*. URL: https://www.baeldung.com/spring-requestmapping. (entered: 29-12-2020).