Taller Clientes Y Servicios

Daniel Felipe Walteros Trujillo

5 de Febrero del 2021

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

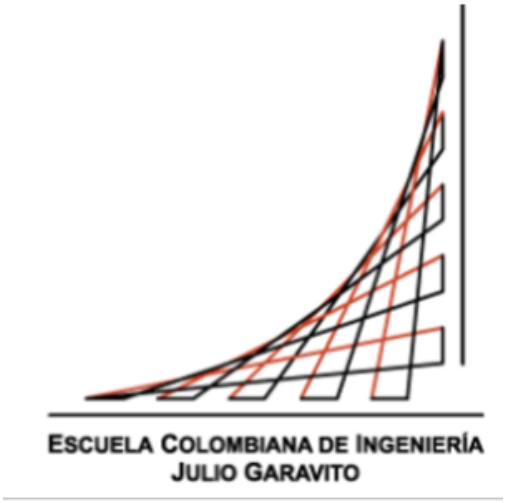


Tabla de Contenido

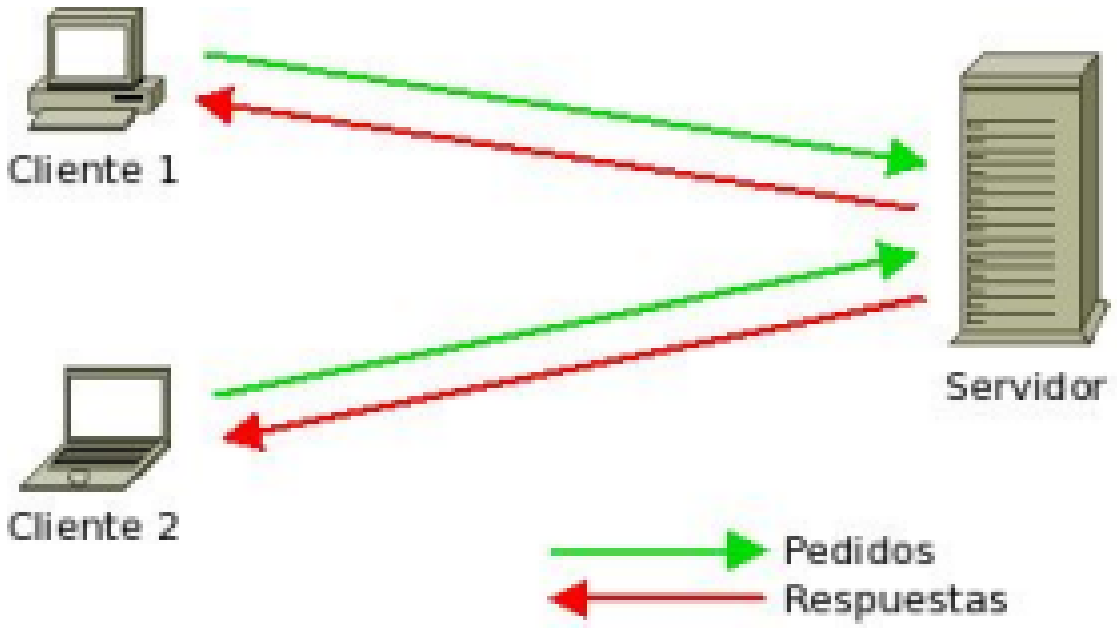
1 Prerrequisitos

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.[4]

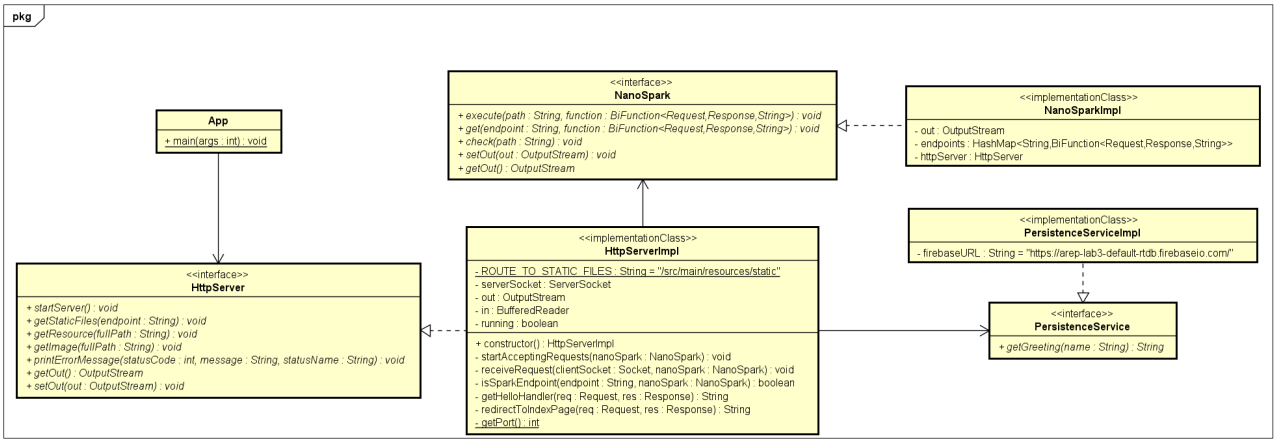


En este trabajo, un servidor web que soporta múltiples solicitudes seguidas (no concurrentes). El servidor retorna todos los archivos solicitados, incluyendo páginas html e imágenes. Su página principal es un sitio web con javascript.

Se encuentra desplegado en Heroku y no usa frameworks web como Spark o Spring, en vez usa un framework propio similar a Spark que permite publicar servicios web "get" con funciones lambda y permite acceder a recursos estáticos, la aplicación realizda sobre ella es simple, pero se conecta con una base de datos desde el servidor.

3 Diseño

3.1 Diagrama de Clases

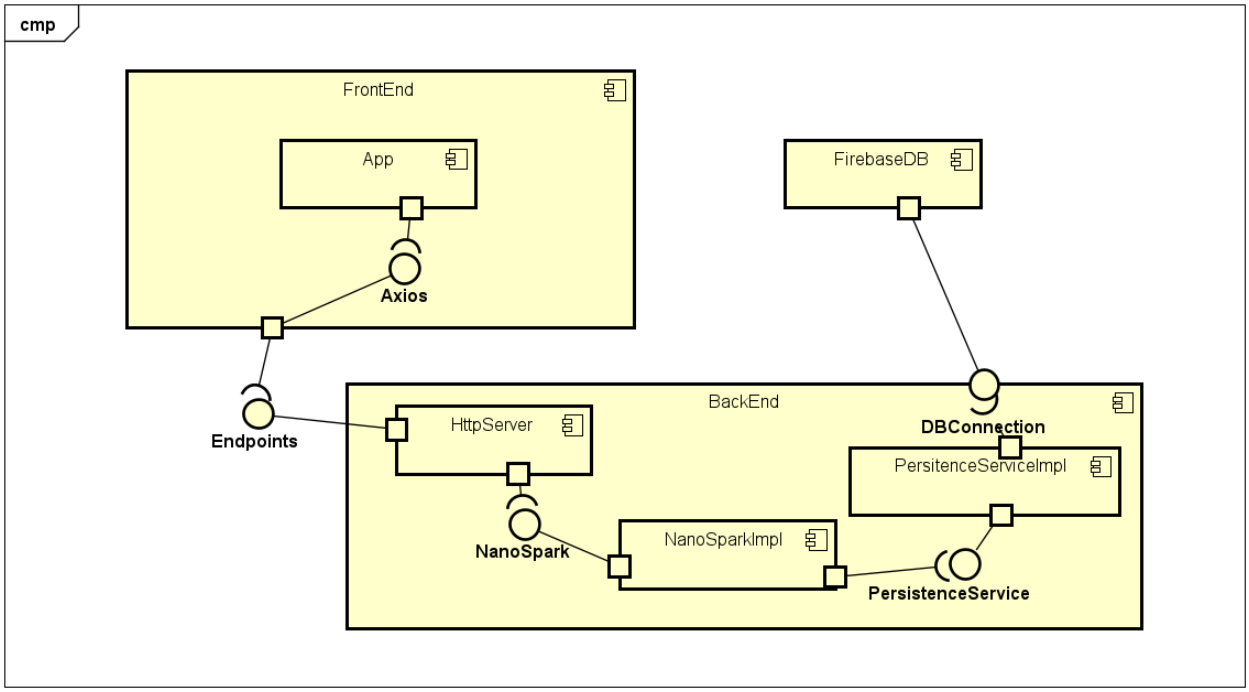


El programa principal utiliza la interfaz HttpServer para crear por medio de sockets un servidor sobre el cual corre una aplicación web, la implementación de esta interfaz utiliza la interfaz PersitenceService para acceder a la base de datos firebase y para simular le comportamiento del framework Spark tiene un atributo de la interfaz NanoSpark.

La interfaz NanoSpark usa el método get para definir endpoints como Spark y tanto la implementación de las funciones lambda[3] como su ejecución se realizaron por medio de la interfaz Funcional BiFunction[2]. El almacenamiento de los diversos endpoints se realizó por medio de un hashmap y cada vez que llega una solicitud al servidor se valida contra los endpoints establecidos.

Una gran ventaja de esta arquitectura es que al desacoplar las funcionalidades con interfaces se pueden realizar cambios o extensiones sin afectar otras capas del modelo, solo basta con crear una clase que implemente la interfaz respectiva y asignarla en la clase que la utiliza.

3.2 Diagrama de Componentes

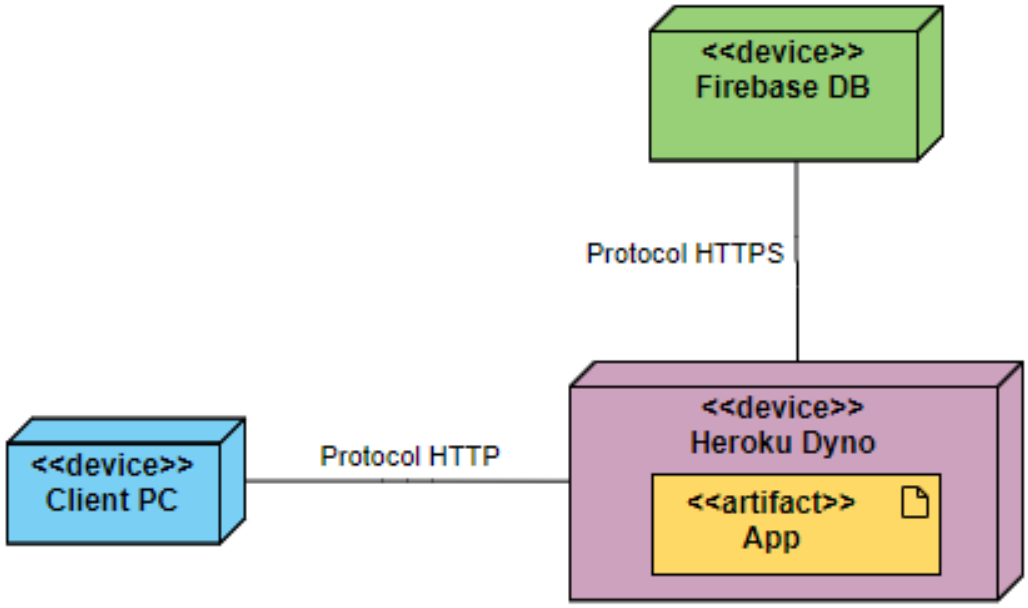


La aplicación se divide en tres componentes principales, FrontEnd, BackEnd y FirebaseDB[1].

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

La función lambda con la que se configuró este endpoint permite conectarse al componente PersistenceServiceImpl, este se conecta la base de datos Firebase para obtener el saludo que retorna junto con el nombre del usuario.

3.3 Diagrama de Despliegue

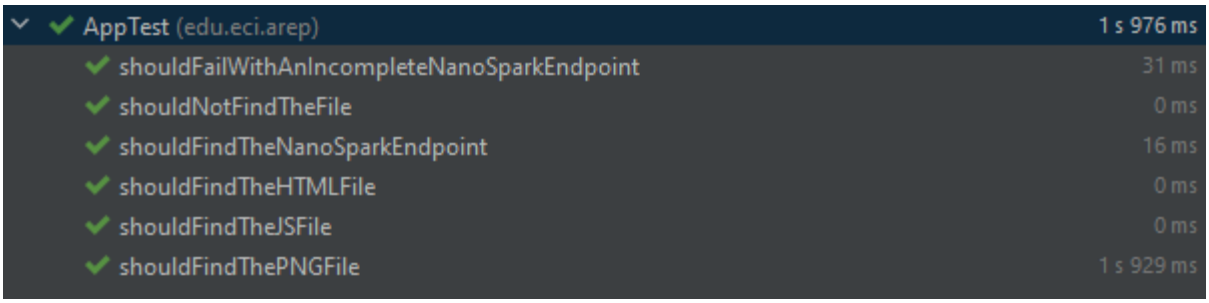


Debido a que utilizando que la aplicación desarrollada se intenta emular el comportamiento del framework Spark, su naturaleza es la de una aplicación web, cualquier persona con conexión a internet puede acceder a la aplicación desplegada, la interacción del cliente con el servidor se realiza únicamente 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 NanoSpark.
- Fallo Por Uso Erróneo de Endpoint Generado Con NanoSpark.



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 NanoSpark por medio de una función que requiere como parámetro una función lambda permite desarrollar de forma más rápida las funcionalidades esperadas que dividiendo cada servicio como componente.

6 Referencias

[1] Equipo Ascenso. *¿Qué es Firebase Realtime Database?* URL: https://ascenso.org/categoria/actualidad-digital/que-es-firebase-realtime-database/. (entered: 21-05-2017).

[2] Baledung. *Guide to Java BiFunction Interface*. URL: https://www.baeldung.com/java-bifunction-interface#bifunction. (entered: 13-07-2020).

[3] Enrique Albaladejo Barambio. *Un lenguaje de modelado para el diseño de lenguajes embebidos en Java 8*. URL: https://repositorio.uam.es/handle/10486/676879. (entered: 01-06-2016).

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