Taller Clientes y Servicios

Juan Sebastián Muñoz Dorado

6 de Septiembre del 2021

Profesor:
Luis Daniel Benavides Navarro

Arquitecturas Empresariales

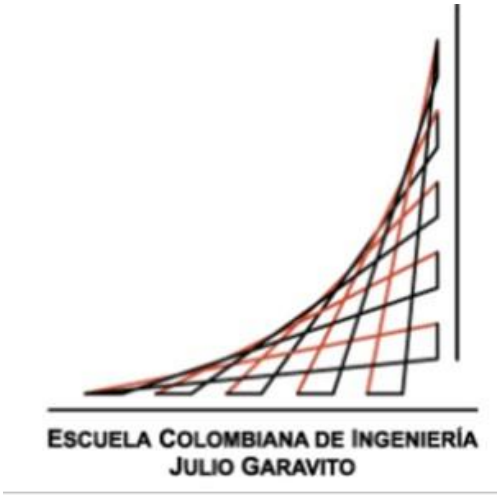


Tabla de Contenido

1 Prerrequisitos

Se utilizará Maven como una herramienta para la construcción y gestión del proyecto, el código fue desarrollado con lenguaje de programación Java; por lo tanto es necesario para la ejecución tener las dos herramientas en las versiones correctas.

- Java versión 8 o superior

- Maven versión 3.5 o superior

2 Introducción

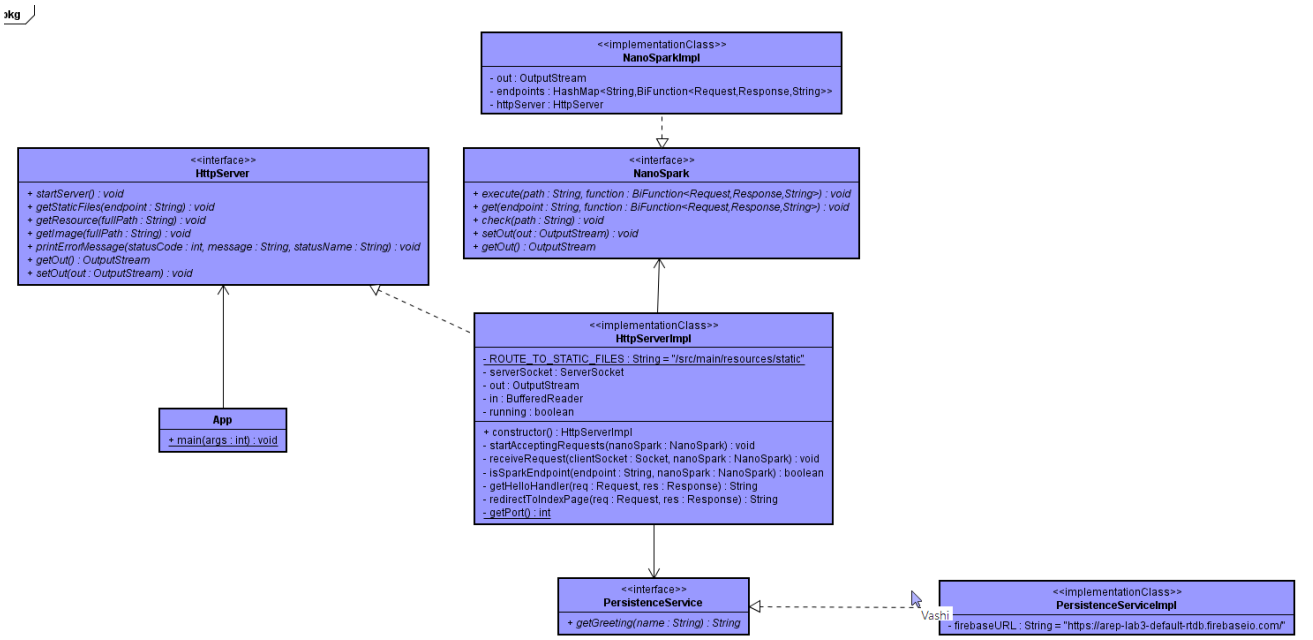
Se usará un servidor y java (en este caso se utilizó Spark). Donde se escribió un framework similar a Spark que le permita publicar servicios web "get" con uso de las funciones lambda y permitio acceder a los recursos estáticos definidos como páginas, (Javascripts, imágenes, y CSSs) .

Se creo una aplicación que conecto con una base de datos desde el servidor para probar la solución propuesta. El componente principal de la solución es un html con javascrpti

La aplicación se desplego en Heroku para su uso posterior.

3 Diseño

3.1 Diagrama de Clases

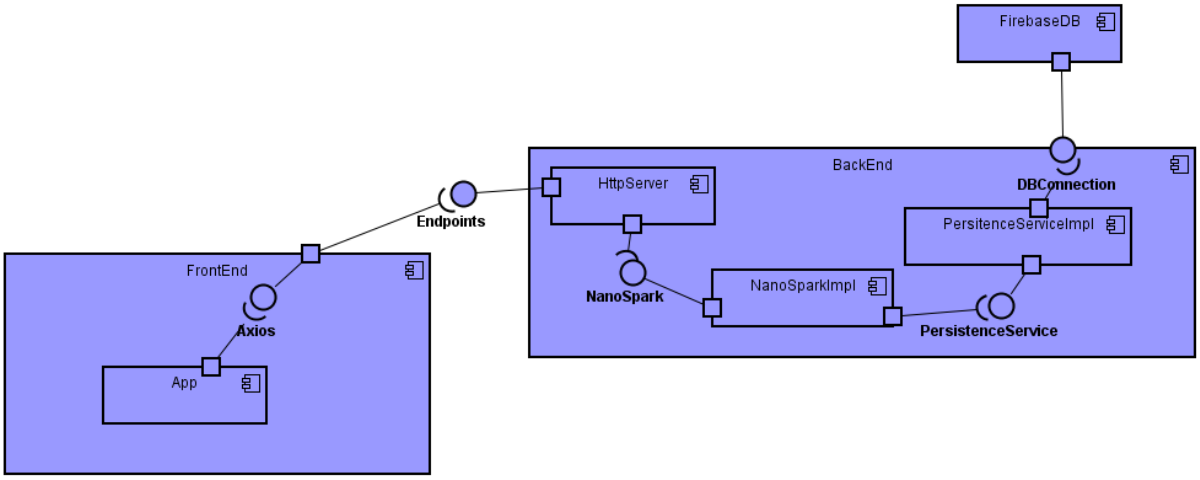


El programa principal utiliza una interfaz definida HttpServer que mediante el uso de sockets el servidor corre una aplicacion web, la solución hecha en esta interfaz utiliza los servicios PersitenceService para acceder a la base de datos Firebase y para simular el comportamiento del framework Spark.

La interfaz NanoSpark utiliza el método get para definir endpoints (tal como esta definido en Spark). La implementacion de las funciones lambda como la ejecución se realizaron por medio de la interfaz Funcional BiFunction.

Se utilizo esta arquitectura porque al desacoplar las funcionalidades con interfaces se pueden realizar cambios o extensiones sin necesidad de afectar las capas de la solución.

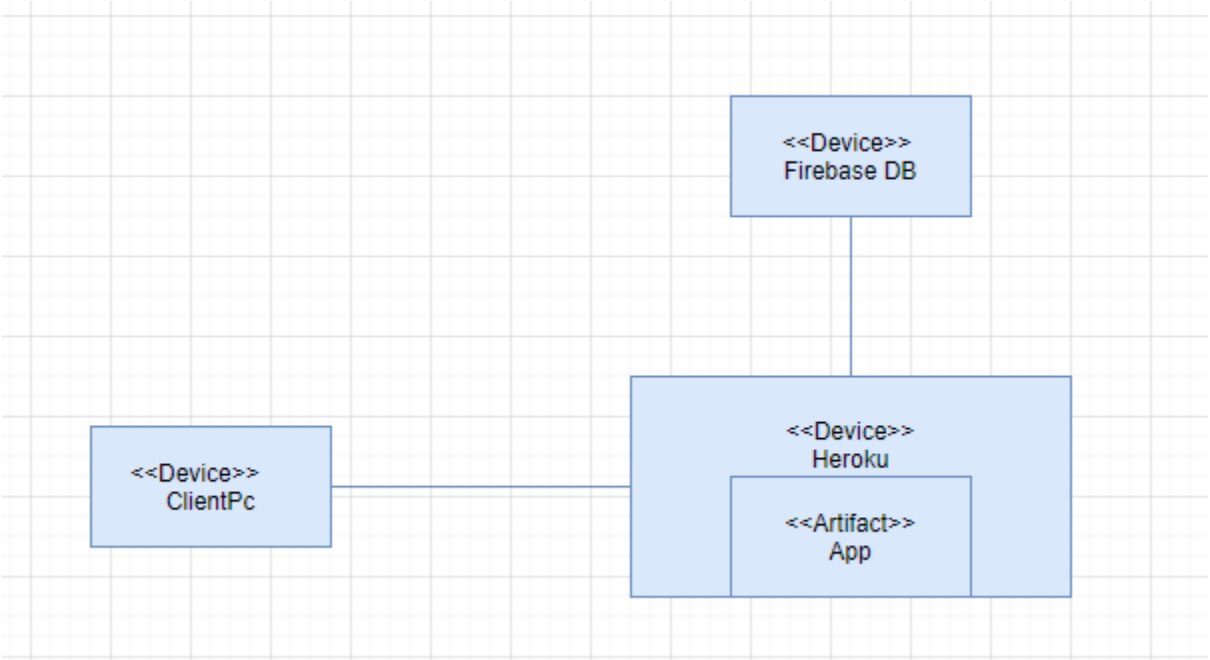
3.2 Diagrama de Componentes



La aplicación tiene tres componentes principales, FrontEnd, BackEnd y FirebaseDB.

El componente que tiene más funcionalidades del FrontEnd es App, el componente lee la información que registra el usuario y por medio de la función Axios accede a HttpServer para usar el endpoint.

3.3 Diagrama de Despliegue



4 Pruebas

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

- Encontrar un archivo HTML.
- Encontrar un archivo JS.
- Encontrar un archivo que no exista.
- Endpoint con NanoSpark.
- Fallo Endpoint con NanoSpark.

5 Conclusiones

- El laboratorio utilizó un servidor y el lenguaje de programación java sin utilizar ningún tipo de framework, asegurando que se cumplieran con los requisitos de implementación del código, para así implementar un framework de cero similar a Spark, esta solución permitió publicar servicios web get con funciones lambda y acceder a recursos estáticos como la página principal, en la cual se podían acceder a imágenes como el logo de la aplicación, interfaz de usuario y JavaScript.
- El despliegue en Heroku permitió poder ejecutar la aplicación solicitada, probando el funcionamiento de la aplicación web en nube y compilando el código en cualquier ordenador, utilizando el protocolo HTTP para la comunicación cliente-servidor para asegurar una total ejecución del código de manera remota.
- Por otro lado, se implementó la integración continua con CircleCI en todo el código fuente para llevar un control de calidad del código, para asegurar que el código esté funcionando totalmente sin ningún problema tanto de compilación como de los resultados retornados después de realizar las respectivas operaciones, y así poder desplegar la aplicación sin presentar ningún tipo de errores.

6 Referencias

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

[2] Baledung. *Guide to Java BiFunction Interface*. URL: https://www.baeldung.com/java-bifunction-interface#bifunction.

[3] Ing. Emiliano Marini. *El Modelo Cliente/Servidor*. URL: https://www.linuxito.com/docs/el-modelo-cliente-servidor.pdf.