WIS Report

Grupo E4.01

https://github.com/DP2-E4-01/D01-Acme-Toolkit

02/03/2022

Integrantes:

Daniel Díaz Nogales	(dandianog@alum.us.es)
Luis Miguel Bellido Zancarrón	(luibelzan@alum.us.es)
Diego González Quintanilla	(diegonqui@alum.us.es)
Eloy Moreno Dominguez	(elomordom@alum.us.es)
José Mª García Quijada	(josgarqui@alum.us.es)
Juan Antonio Mena Vargas	(juanmenvar@alum.us.es)

Versión	Descripción	Fecha
v1.0	Creación inicial	28/02/2022
v2.0	Revisión final del informe	02/03/2022

Resumen ejecutivo

Debido a la necesidad de crear un documento donde dar a conocer los conocimientos de la arquitectura de un WIS, se desarrolla el presente documento.

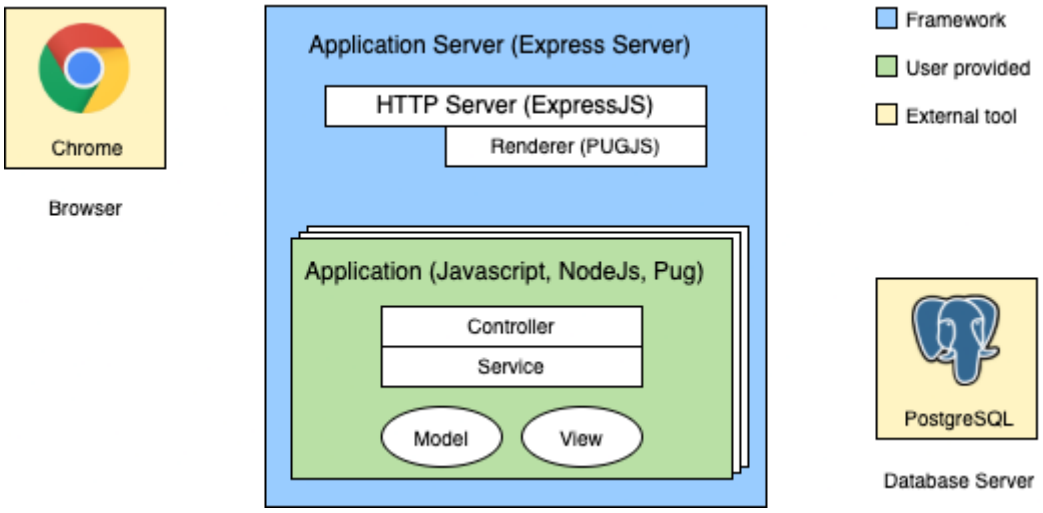
Con el objetivo de resolver dicho problema, dividimos este informe entre una sección donde se habla de NodeJS y otra sobre Spring. Contienen ambos subapartados de definición de componentes y las relaciones entre ellos.

Para finalizar podemos afirmar que, tras crear este informe, hemos adquirido nuevos conocimientos sobre la arquitectura de un WIS, sus componentes y las relaciones que se establecen entre ellos.

1. WIS Report Aplicación Web con NodeJS

Análisis de los componentes y sus relaciones de diversas aplicaciones web, una empleando NodeJS y usando el framework para servidores HTTP ExpressJS y base de datos en PostgreSQL.

1.1 Definición de componentes

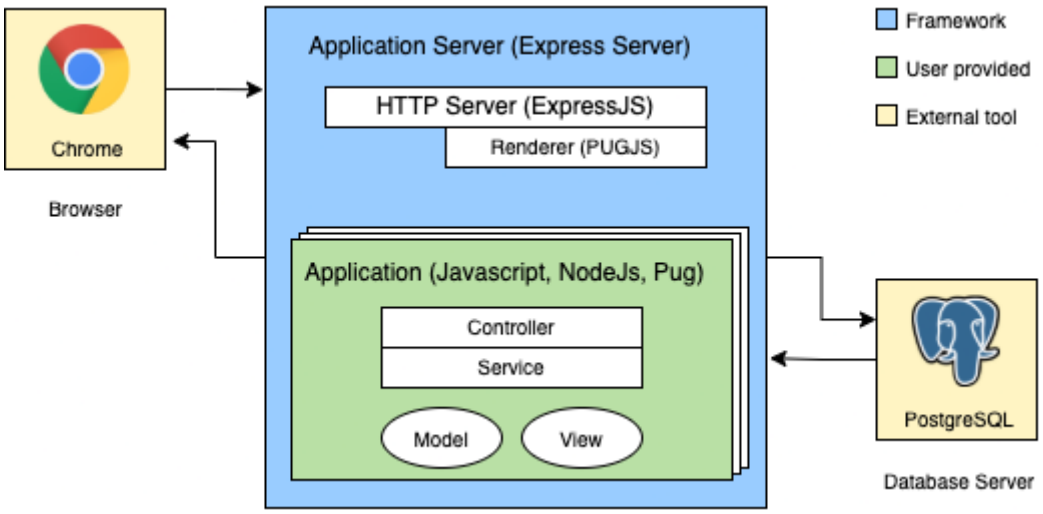


Navegador web como aplicación HTTP cliente solicita peticiones que son respondidas por una **aplicación HTTP servidor con ExpressJS,** y renderiza sus respuestas en HTML, CSS y JS (a no ser que el content type de la respuesta identifique otro formato como JSON, imagen, video, xml…), donde el navegador web tratará de renderizar, guardar o abrir la respuesta con otro programa externo.

La **aplicación web servidor**, recibe peticiones HTTP y genera respuestas que son interpretadas por el navegador. Las respuestas son generadas a través de métodos ligados a rutas.

El servidor de la base de datos recibe las peticiones de la aplicación web servidor con queries, y genera las respuestas con las filas filtradas.

1.2 Relaciones de los componentes



El navegador web realiza las peticiones al servidor web con métodos HTTP y resuelve la respuesta renderizando el contenido y mostrándolo en la aplicación.

Se utiliza el framework HTTP ExpressJS, que simplifica las peticiones y respuestas a través de la aplicación, al mismo tiempo que permite implementar la renderización de documentos en diferentes formatos de respuesta (html, json, img…).

Para el renderizado de html y con el objetivo de simplificar, se puede utilizar librerías externas como PUG, aunque también se puede renderizar archivos html con variables definidas en la parte de servidor.

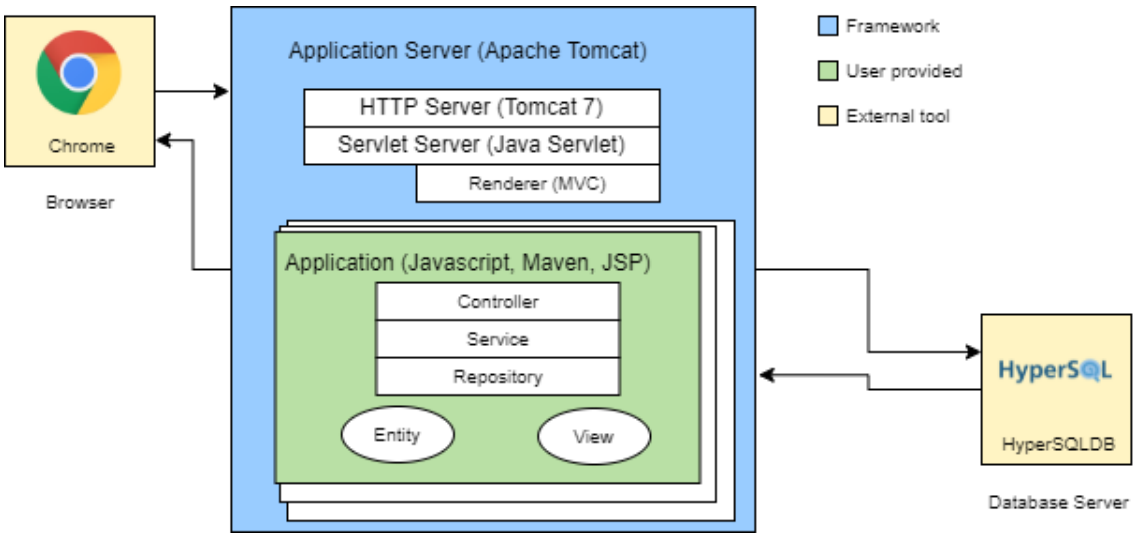
Express permite usar tu propio estilo arquitectónico, pero generalmente se define un archivo de virtualización de rutas que se vinculan a un controlador. Los controladores tienen middleware como callbacks que controlan el procesado de las peticiones HTTP, lo que significa que antes de devolver una respuesta pasan una función como filtro, o alterar la respuesta según diferentes condiciones, por ejemplo mandarte a la página 404 si no has iniciado sesión.

La aplicación web se puede conectar a bases de datos externas haciendo uso de los módulos nativos en NodeJs, o a través de módulos clientes de bases de datos con constructores SQL como Knex.

2. WIS Report Aplicación Web con Spring

Aplicación como Spring que aplica Maven y JSP , además del framework para servidores Tomcat 7 y base de datos HyperSQLDB.

2.1 Definición de componentes

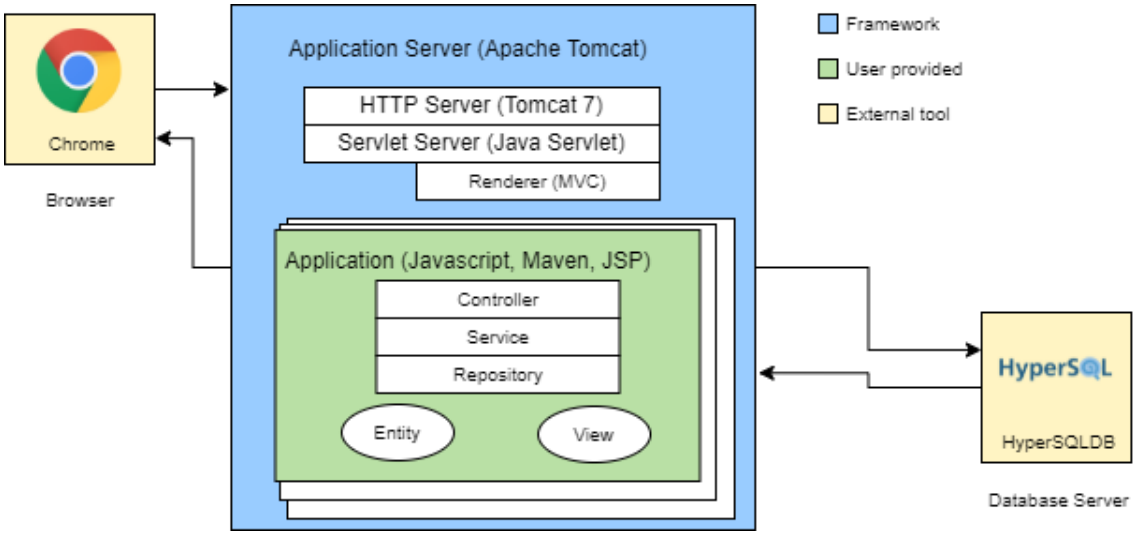


Al igual que en la aplicación anterior, el **navegador web** como aplicación HTTP cliente solicita peticiones que son respondidas por una **aplicación HTTP servidor**, no obstante en este caso es **Tomcat 7,** el cual renderiza sus respuestas en HTML, CSS y JS (a no ser que el content type de la respuesta identifique otro formato como JSON, imagen, video, xml…), donde el navegador web tratará de renderizar, guardar o abrir la respuesta con otro programa externo.

La **aplicación web servidor**, recibe peticiones HTTP y genera respuestas que son interpretadas por el navegador correspondiente. Dichas respuestas se generan gracias a diversos métodos ligados a rutas.

El servidor de la base de datos, en este caso **HSQLDB o HyperSQLDB**, recibe las peticiones de la aplicación web servidor y genera las respuestas apropiadas, las cuáles son devueltas con el propósito deseado.

2.2 Relaciones de los componentes



Teniendo en cuenta que **Spring** nace de la complejidad que se tenía a la hora de crear proyectos Java ya que los EJBeans eran muy complejos y la forma en que se realizaba el despliegue del proyecto era muy engorrosa. Su objetivo principal es proporcionar un conjunto de herramientas para construir rápidamente aplicaciones de Spring que sean fáciles de configurar.

Gracias a su tecnología, nos podemos olvidar de la arquitectura y enfocarnos únicamente en desarrollo, delegando a Spring Boot labores como configuración de dependencias, desplegar nuestro servicio o aplicación a un servidor de aplicaciones y enfocarnos únicamente en crear nuestro código.

Para esto Spring Boot utiliza internamente un servidor de aplicaciones embebido, por defecto utiliza **Tomcat** y no solo esto, Spring Boot también nos provee un completo gestor de dependencias como **Maven**.

Si bien es cierto que Spring es muy potente, la configuración inicial y la preparación de las aplicaciones son tareas bastante tediosas. Spring Boot simplifica el proceso al máximo gracias a sus **dos principales mecanismos**.

Spring cuenta con un **contenedor de aplicaciones integrado**, pudiendo así compilar nuestras aplicaciones Web como un archivo ".jar" que podemos ejecutar como una aplicación Java normal, obteniendo además una distribución de una forma sencilla al poder configurar el servidor junto con la aplicación.

Spring Boot nos proporciona diferentes dependencias llamadas **starters**. Estos nos proporcionan todas las dependencias que necesitamos, tanto de Spring como de terceros. con el objetivo de minimizar la necesidad de configuración a la hora de desarrollar.

Bibliografía

Intencionadamente en blanco.