Conference Paper

Comparative Analysis for Web Applications Based on REST Services: MEAN Stack and Java EE Stack

Análisis comparativo para aplicaciones web basados en servicios REST: stack MEAN y stack Java EE

Jaime Sayago Heredia and Gustavo Chango Sailema

Pontificia Universidad Católica del Ecuador. Sede Esmeraldas, Sede Esmeraldas, Espejo y subida a Santa Cruz Casilla 08-01-0065, 0984787662

Abstract

The use of REST (Representational State Transfer Protocol) web services has increased and has become the most widely used implementation today. The most crucial part of a web development project based on REST services is the choice of the right tools for the front-end, back-end, and database environment. The main objective of this work is to test the MEAN stack as an effective solution for a web application based on REST services. JavaScript has been the client-side programming language running in the browser. The MEAN stack consists of MongoDB (database), Node.js. (web server), Express (back-end), and Angular (front-end) to build web applications. We compare the MEAN stack with a very popular stack such as Java EE that groups MongoDB (database), Apache/Tomcat (web server), Spring boot (back-end) and JSP/HTML/CSS (front-end), with respect to its components, features, and environment configuration. Two similar applications were built with the MEAN and Java EE stacks. We compared and analyzed several features and test criteria such as response time, performance, and data loading.

Resumen

El uso de los servicios web REST (Representational State Transfer Protocol) ha aumentado y se ha convertido en la implementación más utilizada en la actualidad. La parte más crucial en un proyecto de desarrollo web basado en servicios REST es la elección de las herramientas correctas para el front-end, back-end y entorno de base de datos. El objetivo principal de este trabajo es realizar la comprobación del stack MEAN como solución efectiva para una aplicación web basada en servicios REST. JavaScript ha sido el lenguaje de programación del lado del cliente que se ejecuta en el navegador. La stack MEAN está conformada por MongoDB (base de datos), Node.js. (servidor web), Express (back-end) y Angular (front-end) para construir aplicaciones web. Comparamos la stack MEAN con una stack muy popular como lo es Java EE que agrupa a MongoDB (base de datos), Apache/Tomcat (servidor web),

Corresponding Author:
Jaime Sayago Heredia
jaime.sayago@pucese.edu.ec

Received: 4 December 2018
Accepted: 5 December 2018
Published: 27 December 2018

Publishing services provided by Knowledge E

Selection and Peer-review under the responsibility of the SIIPRIN-CITEGC Conference Committee.

Spring boot (back-end) y JSP/HTML/CSS (front-end), con respecto a sus componentes, características y configuración del entorno. Desarrollamos dos aplicaciones similares construidas con las stacks MEAN y Java EE. Comparamos y analizamos varias características y criterios para pruebas como el tiempo de respuesta, rendimiento y carga de datos.

Keywords: software architecture, Javascript, REST, web service, Java EE

Palabras clave: arquitectura software, Javascript, REST, servicio web, Java EE

1. Introducción

La sociedad ha cambiado hacia una mayor dependencia de las tecnologías web que han llegado a convertirse en parte de nuestra vida diaria. Motivando a la academia y a la industria a buscar soluciones para mejorar su disponibilidad, rendimiento y escalabilidad. De esta evolución surgieron los llamados servicios web que se podrían definir como un módulo de software ofrecido por un proveedor de servicios, disponible a través de la web y que es un elemento clave en la integración de sistemas de diferentes plataformas, lenguajes de programación y tecnologías [1]. Una de las implementaciones más utilizadas en la actualidad para servicios web es REST (Protocolo de transferencia de estado representacional) o RESTful. REST fue creado por Roy Fielding y propone un estilo de arquitectura orientada al diseño de software basado en la red [2]. Se puede definir como una arquitectura cliente/servidor. El cliente envía la solicitud al servidor, el servidor procesa la solicitud y devuelve la respuesta. Estas solicitudes y respuestas se basan en la transferencia de representaciones de recursos que se identifica a través de URIs [3]. El uso de servicios web REST han ido en aumento, dando surgimiento a varias herramientas, frameworks y stacks para desarrollo, integración y ejecución de aplicaciones web, las más utilizadas en la actualidad son la stack MEAN con Javascript y la stack de Java EE. El problema radica que al momento de elegir la arquitectura apropiada de tecnologías para desarrollo, integración y ejecución de aplicaciones web basados en tecnologías conocidas y probadas como son los servicios web REST. La elección del stack correcto no es fácil, ya que muchos parámetros tienen que considerarse en la base de datos, en el lado del servidor o el lado del cliente, etc. Diversos autores presentan a la stack MEAN como la mejor solución [4–6]. En este trabajo se plantea una comparación entre la stack MEAN y la stack Java EE y proponer

cuál stack es la solución más efectiva para el desarrollo de aplicaciones web basados en servicios web REST. Se realizó la implementación de una aplicación para el manejo de la hoja de vida y gestión docente para el departamento de talento humano de la Pontificia Universidad Católica del Ecuador PUCE, Sede Esmeraldas – Ecuador, que permite almacenar, gestionar y analizar la información de los docentes y su gestión docente semestral. Esta aplicación web fue construida utilizando las dos stacks para probar y analizar los pros y contras de las dos stacks, junto con algunas pruebas de evaluación comparativa.

2. Arquitecturas de software

Cada sistema informático, grande o pequeño, está formado por piezas que están unidas entre sí. Puede haber un pequeño número de estas piezas, o tal vez solo una, o puede haber docenas o cientos; y este vínculo puede ser trivial, o muy complicado, o en algún punto intermedio, es decir cada sistema tiene una arquitectura [7]. Al tratar de definir el concepto de Arquitectura de Software existen varias definiciones alternativas o contrapuestas. Pero hay algunas que son reconocidas, a continuación, las citamos. Se puede definir la arquitectura de software como la descomposición de un sistema de nivel superior en cada uno de sus componentes principales, las interfaces y su comunicación [9]. La IEEE la define de la siguiente manera: Es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución [10]. Dentro de las ventajas de la arquitectura de software, podemos mencionar que: simplifica la capacidad de comprender sistemas complejos planteando limitaciones en el diseño, reutiliza componentes en múltiples niveles, separa elementos, lo que permite mayor facilidad en el cambio y mantenimiento del desarrollo, aplica utilización de patrones [11]. El uso de arquitectura de software puede realizarse de una manera eficiente en función del tiempo y ser rentable ya que permite la reutilización de los componentes y patrones de diseño en los proyectos [12]. Al implementar una arquitectura de software es fundamental el uso de los patrones arquitectónicos que brindan un principio general de estructura. Un patrón arquitectónico expresa un esquema de organización estructural fundamental para los sistemas de software. Proporciona un conjunto de subsistemas predefinidos, especifica sus responsabilidades e incluye reglas y pautas para organizar las relaciones entre ellos [13].

2.1. REST

El estilo de Arquitectura REST utiliza identificador de recursos uniforme URI para acceder y manipular los estados de los recursos en cual se proporciona un identificador único para un recurso web [3]. REST normalmente tiene muchos recursos y es una arquitectura orientada a los recursos [14]. En REST se acceden a los recursos mediante una interfaz genérica la cual reduce dramáticamente la complejidad de la semántica de la interfaz durante la interacción de servicios. Las necesidades de ancho de banda y recursos de REST son bajos, por lo que se considera ligero [15]. Las aplicaciones desarrolladas utilizando REST son llamadas aplicaciones RESTful que proporcionan más flexibilidad y menos gastos generales, lo que resulta en una mejor solución para la mayoría de las implementaciones [16].

2.2. Servicios web

El término servicios web según la W3C (World Wide Web Consistorium) proporcionan un medio estándar de interoperabilidad entre diferentes aplicaciones de software que se ejecutan en una variedad de plataformas y/o marcos de trabajo [17]. Un servicio web es un sistema de software diseñado para soportar la interacción entre máquinas a través de una red. Tiene una interfaz descrita en un formato procesable por máquina. Otros sistemas interactúan con el servicio web de una manera prescrita por su descripción utilizando mensajes, típicamente transmitidos usando HTTP con una serialización en conjunción con otros estándares relacionados con la web [17]. Se podría entender como un conjunto de aplicaciones o tecnologías que pueden interoperar en la web, su principal función es la de intercambiar datos entre sí; tanto los proveedores ofrecen sus servicios y los usuarios solicitan el servicio llamando a estos procedimientos utilizando la web [18]. Estos servicios proveen de mecanismos de comunicación estandarizados entre las distintas aplicaciones, que pueden interactuar entre ellas para entregar la información al usuario. La interoperabilidad y extensibilidad de las aplicaciones y su combinación para operaciones complejas requiere la utilización de un estándar [17].

2.3. Arquitectura REST

REpresentational State Transfer (REST) es un estilo arquitectónico propuesto por Fielding [19]. REST es para sistemas hipermedias distribuidos a gran escala y que logra que

la World Wide Web (WWW) sea escalable. Fielding argumenta que en REST es la existencia de recursos (elementos de información), que pueden ser accedidos utilizando un identificador global (un Identificador Uniforme de Recurso). Para manipular estos recursos, los componentes de la red (clientes y servidores) se comunican a través de una interfaz estándar (HTTP) e intercambian representaciones de estos recursos (los ficheros que se descargan y se envían). El cliente puede navegar esencialmente a través de una amplia gama de recursos existentes, siguiendo los enlaces de un recurso a recurso [20]. Un principio clave de REST es la interacción sin estado entre los participantes en una conversación. Un estado en este caso significa el estado de la aplicación/sesión. El estado se mantiene como parte del contenido transferido del cliente al servidor/servicio y viceversa [21]. Más concretamente, en el caso de los servicios, los clientes que desean utilizar un servicio acceden a una representación particular de los recursos que representan el servicio mediante la transferencia de contenido de la aplicación utilizando un conjunto pequeño y definido globalmente de métodos remotos [21]. Estos métodos describen la acción a realizar sobre los recursos. Los métodos HTTP para crear, leer, actualizar y borrar recursos, cada uno identificado por un URI, son (PUT, POST, GET, y DELETE en HTTP 1.0, mientras que HTTP 1.1 permite extensiones) [22]. En REST, cada solicitud enviada a un objeto resulta en la transferencia de una representación de este objeto por ejemplo, texto, XML, JSON, etc. [20]. REST se ha convertido en la implementación más utilizada en la actualidad [23].

3. Stack MEAN

La stack MEAN es una solución potente y completa. Utiliza el lenguaje de programación JavaScript [24]. Comprende cuatro bloques principales: MongoDB como base de datos, Express como marco de trabajo del servidor web, AngularJS como marco de trabajo del cliente web y Node.js como plataforma del servidor. Estos componentes están desarrollados por diferentes equipos e involucran a una comunidad fuerte de desarrolladores y defensores impulsando el desarrollo y documentación de cada componente. Sin embargo, un problema que podría afectar dramáticamente su proceso de desarrollo y presentar problemas de escalamiento y arquitectura es la conexión de estas herramientas [25]. La principal fortaleza de la pila MEAN radica en su centralización de JavaScript como el principal lenguaje de programación, ya que cada componente está escrito en JavaScript, incluso la base de datos almacena los datos en formato JavaScript Object Notation (JSON) que es el único script que JavaScript entiende completamente [26]. Por lo tanto, JavaScript no sólo se utiliza como lenguaje de scripting del lado del

cliente, sino que también se utiliza a lo largo de la aplicación, es decir, en el lado del cliente, del servidor y de la base de datos [4]. El uso de JavaScript como lenguaje principal de programación tanto en el lado del cliente como en el lado del servidor hace que la pila MEAN sea más potente y reduce el tiempo en la construcción de la aplicación [5]. El uso de todo el JavaScript permite dividir la funcionalidad y las tecnologías utilizadas de la siguiente manera [26]:

- Base de datos objetos JSON utiliza MongoDB.
- Servidor web utiliza Node.js.
- Framework web (back-end) utiliza Express.
- Cliente (front-end) utiliza Angular.

3.1. MongoDB

MongoDB es un modelo de almacenamiento de documentos NoSQL potente, adaptable y escalable [27]. En MongoDB, los datos se almacenan en la base de datos en un formato JSON llamado BJSON, que significa JSON binario, en lugar de filas y columnas como en la base de datos relacional. MongoDB tiene una capacidad para escalar con varias características que la base de datos relacional proporciona como índices secundarios, clasificación y consultas [28], que proporciona alto rendimiento, alta disponibilidad y escalabilidad [25].

3.2. Express

Express es un framework web maduro y flexible para construir aplicaciones web sobre el ecosistema Node. Por defecto, el framework Express utiliza el motor Pug para soportar plantillas [29]. Express es un framework relativamente pequeño que se encuentra en la parte superior de la funcionalidad del servidor web de Node para simplificar sus APIs y añadir nuevas funciones útiles. Facilita la organización de la funcionalidad de su aplicación con middleware y enrutamiento; agrega utilidades útiles a los objetos HTTP de Node y el renderizado de vistas HTML dinámicas; define un estándar de extensibilidad fácilmente implementado [30]. Express es fácil de configurar, implementar, controlar y proporcionar varios componentes clave para manejar las solicitudes web. Express ayuda en la creación de aplicaciones web y servidores HTTP simples ya que es un framework mínimo y flexible [30]. En MEAN, Express funciona como un medio

para transferir las solicitudes de un cliente a una base de datos y envía las respuestas de la base de datos al cliente [26].

3.3. AngularJS

AngularJS es una librería escrita en JavaScript para el desarrollo de aplicaciones web, mantenida por Google, es un framework JavaScript de código abierto y aborda los retos de las single-page aplications (SPAs) [31]. Una aplicación web AngularJS sigue el patrón de diseño MVC, que resulta en el desarrollo de aplicaciones web ampliables, mantenibles, comprobables y estandarizadas [31]. La unión de datos AngularJS y la inyección de dependencias lo convierten en un socio ideal para cualquier tecnología de servidor, ya que elimina gran parte del código que de otro modo tendrías que escribir, y todo sucede dentro del navegador [26].

3.4. Node

Node es un framework de desarrollo desarrollado originalmente en 2009 por Ryan Dahl, basado en el motor JavaScript V8 de Google [32]. Node es una plataforma basada en el tiempo de ejecución JavaScript de Chrome para crear aplicaciones en red rápidas y escalables [25]. Node utiliza un modelo controlado por eventos y de no bloqueo de E/S que lo hace ligero y eficiente, perfecto para aplicaciones que requieren grandes cantidades de datos y que se ejecutan en dispositivos distribuidos [26]. Node es un lenguaje de scripting del lado del servidor que puede ser usado en el lado del servidor, lado del cliente, e incluso puede ser un servidor web. Antes de que existiera el Node, JavaScript se usaba simplemente para la interacción del usuario como script del lado del cliente [33].

4. Stack Java EE

Java que es uno de los lenguajes de programación más utilizados y que tiene una variedad de herramientas para diferentes propósitos [34]. En este caso se combina varias herramientas y componentes la conformar la stack Java EE. En una aplicación web los componentes están en el front-end o en el back-end [31]. La stack Java EE que es muy utilizada en para el desarrollo de aplicaciones modernas [35]. Estas tecnologías deben trabajar de una manera fácil y armoniosa. En el Front-End se usa JSP/HTML/CSS, utilizando el patrón MVC. En el back-end Spring Boot y la base de datos Postgresql,

MongoDB o cualquier otro motor de base de datos. El uso de toda la stack Java EE permite dividir la funcionalidad y las tecnologías utilizadas de la siguiente manera, que conforman la stack seleccionada:

- Base de datos MongoDB.
- Spring boot para el (back-end).
- Cliente (front-end) utiliza JSP/HTML/CSS.
- La plataforma Apache/Tomcat como servidor web.
- A continuación, se procede a describir cada una de las tecnologías utilizadas.

4.1. Spring boot

Spring Boot es un framework que posee las bibliotecas necesarias para la creación de aplicaciones ejecutables basadas en Spring. Los componentes que posee Spring Boot permiten simplificar los pasos de la selección de las dependencias necesarias para el proyecto a desarrollar y su despliegue en el servidor. No requiere de configuración usando archivos en XML/JSON y permite la creación de servicios REST en Java de manera fácil, centrando al desarrollador en los elementos críticos de desarrollo especifico de su aplicación [36].

4.2. Spring data rest

Spring MVC ofrece una base sólida para construir los tipos de servicios web REST que son el medio más utilizado para la integración de aplicaciones en la web, pero la implementación de la funcionalidad básica del servicio web REST puede ser tediosa y dar como resultado una gran cantidad de código [37]. Spring Data REST se basa en los repositorios de Spring Data y los exporta automáticamente como recursos de REST. Aprovecha los hipermedios para permitir a los clientes encontrar la funcionalidad expuesta por los repositorios y permite integrar los recursos en la funcionalidad relacionada basada en hipermedios tan fácilmente como sea posible. Spring Data REST es en sí misma una aplicación Spring MVC y está diseñada de tal manera que debe integrarse con sus aplicaciones Spring MVC existentes con muy poco esfuerzo. Una capa de servicios existente (o futura) puede funcionar junto con Spring Data REST con cambios menores.

4.3. Gradle

Es una herramienta de compilación basada en JVM, usa un lenguaje potente y expresivo Domain Specific Language (DSL) implementado en Groovy en lugar de XML. Debido a que Gradle es nativo de JVM, permite escribir una lógica personalizada sea en el lenguaje Java o Groovy. Sin embargo, escribir los archivos de construcción de Gradle no requieren un conocimiento profundo de Groovy [38]. Gradle tiene mayores ventajas a diferencia de Maven o Ant, una de ellas es que Gradle no usa XML, en cambio Maven y ANT si lo usan, lo cual queda algo corto en expresar el flujo del programa y la lógica condicional. Gradle brinda una sofisticada gestión de dependencias robusta y potente y son muy adecuados para construir proyectos basados en Java, debido a que permite sin mayor esfuerzo el consumo de componentes o módulos desarrollados por terceros o definir su propia biblioteca de dependencias por proyecto permitiendo construir aplicaciones escalables. Brinda muchos beneficios, el más importante es su flexibilidad. [39].

4.4. JSP/HTML/CSS

HTML (HyperText Markup Language) que en español significa Lenguaje de Marcas de Hipertexto es considerado un lenguaje de marcado principal de la World Wide Web que originalmente se diseñó para detallar documentos científicos semánticamente, sin embargo, al paso de los años se ha podido adaptar para detallar y describir otros tipos de documentos como por ejemplo aplicaciones [40]. HTML se encuentra en su versión 5.2 que provee diferentes características como estructura, funcionalidad, adaptabilidad y estilo; al hablar de este último, CSS (Cascading Stylesheets) se destaca que estas tecnologías son altamente dependiente entre sí [41]. Java Server Page o mejor conocida como JSP es una página HTML que brinda la ventaja de poder incrustar código Java facilitando así la creación de páginas web dinámicas [42]. Una de las principales ventajas que ofrece JSP ante otros lenguajes es que con él se pueden crear aplicaciones web que manejen una lógica de negocios y acceso a datos de manera fácil y sencilla, de tal manera que permita la separación de los niveles dentro de la aplicación. Otra ventaja es la posibilidad de heredar la portabilidad del lenguaje Java [42].

5. Stack MEAN y Java EE: Características, ventajas y desventajas

En este apéndice se muestran las características, ventajas y desventajas de cada una de las stacks. MEAN gana popularidad debido a las características de sus componentes: Node.js, MongoDB, AngularJS, junto con Express y todo utilizando JavaScript que es una de sus mayores ventajas. Aunque podemos afirmar que MEAN no se recomienda en sistemas pesados de procesamiento computacional, debido a que el componente Node.js es un entorno de un solo hilo y esto sería un inconveniente fuerte de esta tecnología. La madurez de la tecnología Java EE es la opción más correcta para este tipo aplicaciones pesadas de procesamiento computacional. Pero para una aplicación de una página (SPA), que requiere una alta interacción con el usuario, visualización y alta escalabilidad, la pila MEAN es la mejor opción. Cabe recalcar que en el ámbito empresarial Java EE es la tecnología más popular para el desarrollo de aplicaciones web por la seguridad que brinda esta tecnología. La elección de la stack depende principalmente de algunas características, que las vamos a analizar para conformar un conjunto de parámetros que permiten realizar la comparativa entre estas dos stacks, en definitiva, se trata de buscar un marco de evaluación de la arquitectura con los criterios elegidos. A continuación, se detalla los parámetros elegido y al final encontramos un resumen de las stacks a partir de la elección de parámetros efectuada. La siguiente tabla muestra un resumen de las características de cada stack. Podemos decir que ambas stacks tanto MEAN y Java EE, cumplen con las características evaluadas manejándolas de distinta forma a través de distintas tecnologías. La stack MEAN resuelve de mejor manera la mayoría de las características evaluadas antes que la stack Java EE.

6. Caso de Estudio: Comparaciones, pruebas y análisis

En esta sección se procederá se presenta la aplicación web, las pruebas junto con la interpretación de resultados de comparativas y pruebas de la aplicación web de hoja de vida y gestión docente a través de servicios REST. Para probar y analizar los pros y contras se las dos stacks, se desarrolló la aplicación de la hoja de vida y gestión de docente para el departamento de talento humano de la Pontificia Universidad Católica del Ecuador, Sede Esmeraldas, que fue construido utilizando ambas pilas, seguido de algunas pruebas de evaluación comparativa. Algunos de los criterios como el desarrollo de aplicaciones en tiempo de respuesta, el rendimiento, la carga de datos son demostrados en las siguientes secciones. Las pruebas de la aplicación fue con la ayuda

Tabla 1: Resumen de características de las stacks.

Característica	MEAN	Java EE
Capa de servidor simplificada	Construcción de servidor localmente	Empaquetamiento de artefacto en un servidor embebido
Código isomórfico	Código puede ser ejecutado en el front-end como el back-end	Código puede ser ejecutado en el front-end como el back-end
Escalabilidad	La aplicación puede expandirse los eventos se ejecutan libremente de manera asíncrona	La aplicación se expande a través del servidor embebido con una distribución multihilos.
Arquitectura sin bloqueo	Maneja el uso de bucles de eventos sin bloqueo en un solo hilo.	El bloqueo se resuelve a través de múltiples hilos de ejecución
Tiempo de desarrollo	Al utilizar el mismo lenguaje JavaScript se reduce el esfuerzo y tiempo de desarrollo	Al utilizar el mismo lenguaje Java se reduce el esfuerzo y tiempo de desarrollo. Existen varios marcos de trabajo para mejorar el tiempo de desarrollo
Transformación de datos y extensibilidad	Los datos manejados para el intercambio de información es JSON.	Los datos manejados para el intercambio de información por estándar son XML, pero se puede trabajar con JSON.

de tres herramientas de evaluación comparativa: Siege [43], Apache Bench [44] y de tiempo de respuesta fue JMeter [45]. La aplicación que se desarrolló es un CRUD, con varios datos del docente para llenar su hoja de vida por ejemplo datos personales, títulos, experiencia docente, experiencia no docente, capacitaciones, investigaciones, artículos, congresos, libros, etc. Junto con un informe de actividades de fin de cada semestre con la información de alumnos tutorados, cursos, seminarios, congresos y proyectos de vinculación realizados. El desarrollo de esta aplicación fue de apreciar que tan fácil o complicado puede ser realizar una aplicación utilizando ambas stacks. El objetivo principal era para ilustrar la el tiempo de respuesta tanto en la stack MEAN como en la stack Java EE, como también medir su rendimiento y carga de datos. Para el propósito de la prueba, alrededor de 300 registros de docentes y de gestión fueron insertados en las dos aplicaciones.

6.1. Prueba de tiempo respuesta de las aplicaciones

Las mediciones de tiempo de respuesta de las aplicaciones construidas con los dos stacks MEAN y Java EE, se han realizado mediante Apache JMeter. El tiempo de respuesta significa el intervalo entre el momento en que se envía una solicitud y el momento en que JMeter recibe la respuesta [46]. Para esta prueba se desarrolló un script en JMeter para la creación de la petición de datos. Básicamente, el script intenta acceder a la aplicación y realizar una función de búsqueda de información. El experimento se realizó en el entorno donde glassfish esto en la stack Java EE y para

la stack MEAN el entorno Node.js. El script de JMeter se lo ejecuto por separado. La siguiente tabla muestran los resultados de las dos pruebas.

Tabla 2: Tiempo de respuesta de las dos stacks MEAN y Java EE.

Etiqueta	Muestras	Promedio	Error %	Rendimiento	Kb/seg Recibidos
Java EE	1000	953	0,00%	350,30%	2351,28
MEAN	1000	2	0,00%	352,50%	412,12

Como se aprecia en la tabla se hicieron 1000 solicitudes de muestras. El tiempo promedio transcurrido (columna Promedio) fue de 953 milisegundos y el porcentaje de error (columna Error %) fue de 0 para la stack Java EE. Para la stack MEAN el tiempo promedio transcurrido (columna Promedio) fue de 2 milisegundos y el porcentaje de error (columna Error %) fue de 0. El porcentaje de error es la columna que muestra el porcentaje de solicitudes rechazadas de esta tabla. En ambas stacks esta columna es la que no varía entre ejecuciones. El resto de las columnas muestran la rapidez con la que se manejaron las solicitudes (columna de rendimiento). Respecto al rendimiento hay una pequeña diferencia favorable para la stack Java EE 350.30% frente a 352.50% para la stack MEAN. Por último, la cantidad de kilobytes recibidos (columna de kb/segundo recibidos). En esta podemos apreciar que la stack Java EE su velocidad es de 2351.28 kb/segundo frente a 412.12 kb/segundo de la stack MEAN. A continuación, la gráfica de la prueba realizada con el script de JMeter.

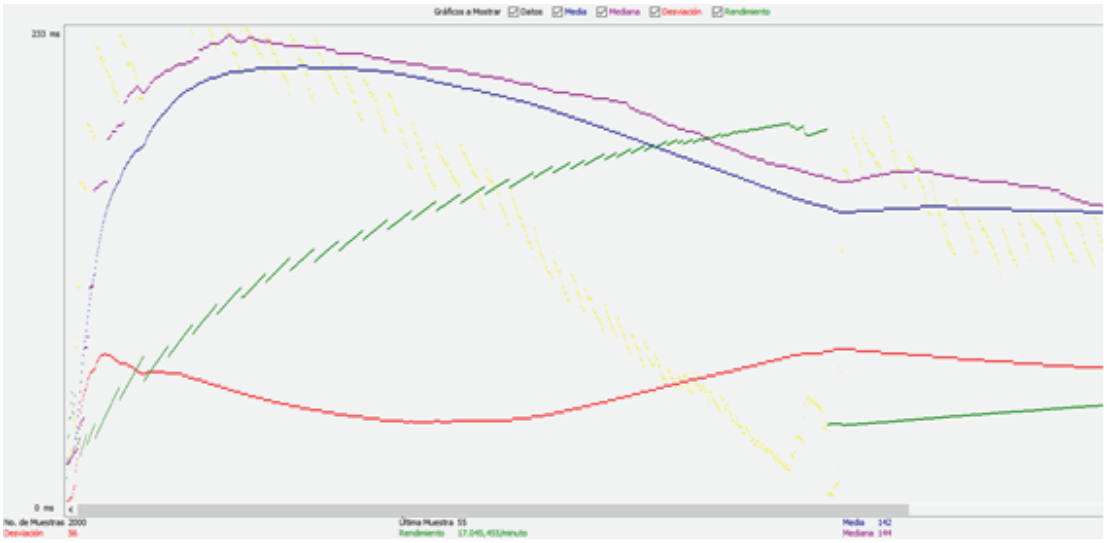

Figura 1: Gráfica JMETER prueba tiempo de respuesta.

6.2. Pruebas de rendimiento

Para esta prueba, usamos la herramienta Apache Bench para probar el tiempo de respuesta del sistema. La prueba se ejecutó con varios niveles de concurrencia y número de peticiones. La concurrencia es la medida de cuántas sesiones de usuario simultáneas están activas en una aplicación web en un momento dado. Con un aumento en la concurrencia, podemos probar cuántas sesiones de usuarios concurrentes puede soportar una aplicación web en términos de tiempo de respuesta para realizar cualquier tarea. Este es un comando de ejemplo para comparar un servidor que se ejecuta localmente en el puerto 8080 con 10.000 solicitudes en total y 500 solicitudes concurrentes: "ab -n 10000 -c 500 http://localhost:8083/GestionDocenteProyecto/". Se realizaron varias pruebas con aumentos en la concurrencia y el número de solicitudes como se muestra en la Tabla 1. Las pruebas se realizaron para MEAN y Java EE con la concurrencia establecida en 10, 50, 100 y 500, y el número de solicitud como 100, 500, 1000, 5000, 10000, 50000 y 100000.

Tabla 3: Tiempo de respuesta en milisegundos de 100000 solicitudes.

Concurrencia	**10**	**50**	**100**	**500**
JAVA (respuesta en ms)	64.216	180.072	224.170	1.451.031
MEAN (respuesta en ms)	8.395	34.006	68.301	697.426

Observamos en la tabla que es la prueba de 100000 respuestas y un nivel de concurrencia de 10, 50, 100 y 500. En la que para la stack Java EE el tiempo medio que el servidor ha tardado en atender un grupo de peticiones fue de 64.216, 180.072, 224.170 y 1.451.031 milisegundos respectivamente. Y para MEAN este tiempo de fue de 8.395, 34.006, 68.301 y 6097.426 milisegundos. Se puede apreciar que mientras aumenta el número de solicitud de respuesta y concurrencia mejora el rendimiento de la stack MEAN sobre el stack Java EE.

6.3. Pruebas de transferencia de datos y concurrencia

Prueba de transferencia de datos y concurrencia utilizando Siege. Para probar cargas en ambas aplicaciones construidas en las stacks MEAN y Java EE, la función Siege se utilizó un instrumento de evaluación comparativa. A continuación, se muestra un comando de ejemplo para comparar un servidor que se ejecuta localmente en el puerto 4200 con 150 peticiones simultáneas, durante un lapso de 5 minutos [47] para la stack MEAN:

```
siege --concurrent=150 --time=5M http://localhost:4200.
```

Para la prueba la realización de la prueba para la stack Java EE, se utiliza la ejecución localmente hacia el puerto 8083 donde está alojada la aplicación web. La prueba realizada por Siege proporciona el tiempo transcurrido, la cantidad de datos transferidos, tiempo de respuesta, rendimiento y tasa de transacciones. Estos datos fueron recolectados de Siege pasando el valor de concurrencia y el URL de la aplicación como entrada. El resultado de la prueba de asedio para los datos transferidos para las stacks Java EE y MEAN se muestra en las siguientes Figuras y sus respectivos análisis. En la siguiente tabla se resume la tasa de transacciones promedio de transacciones que el servidor fue capaz de manejar por segundo. Con los distintos niveles de concurrencia de 10, 20, 50, 100 y 500. Esta tasa fue de es menor de transferencias por segundo para MEAN y es superior los datos de transferencia por segundo para Java EE.

Tabla 4: Datos transferidos durante la concurrencia.

Concurrencia (MB)	5	10	20	50	100	150
MEAN	2,66	5,3	10,61	26,57	53,26	79,36
Java EE	18,48	34,29	74,25	185,09	365,46	547,32

Tabla 5: Datos de concurrencia de las stack MEAN y Java EE.

Concurrencia	5	10	20	50	100	150
MEAN	0,03	0,06	0,13	0,41	0,72	1,04
Java EE	0,08	0,16	0,34	0,93	2,17	3,97

Como lo podemos observar en la tabla 5, se observa de mejor manera la prueba de concurrencia que es mayor para la stack Java EE. Mientras que la stack MEAN es menor. Esta medida es un parámetro importante en el estudio. Por lo que podemos afirmar que el servidor tiene un mejor rendimiento en la stack MEAN que el servidor de la stack Java EE.

7. Resultados, conclusiones, recomendaciones y limitaciones

En el estado del arte se hizo una descripción de las tecnologías para arquitecturas y servicios web, describió y analizó dos tipos de tecnologías de stacks para el desarrollo de web aplicaciones utilizando servicios REST: la stack MEAN y la stack Java EE. La stack MEAN es un paquete de software que combina MongoDB como la base de datos NoSQL, Express como un framework de Node.js para el scripting para el

desarrollo del back-end, Angular como plataforma MVC para la construcción del front-end, construido con código JavaScript. La stack Java EE es la combinación de MongoDB como el sistema de base de datos, Spring Boot para el back-end y para el front-end se utiliza JSP/HTML/CSS, y Apache/Tomcat como servidor web. En la stack MEAN, JavaScript es el único lenguaje de programación tanto para el front-end y para el back-end. En la stack Java EE, utiliza el lenguaje de programación java del lado del back-end y del lado del front-end junto con HTML/CSS. La creciente popularidad del uso de JavaScript como secuencias de comandos del front-end y del back-end ha hecho que MEAN sea una las combinaciones de tecnologías más utilizadas para desarrollo aplicaciones web y servicios REST. La stack MEAN está construida exclusivamente en JavaScript, por lo que es un lenguaje para gestionar el lado del cliente, servidor y base de datos. Todos los componentes utilizados en la stack MEAN son completamente de código abierto y actualmente son soportados por desarrolladores corporativos como MongoDB y Google. Todos los componentes de MEAN son relativamente ligeros. La stack MEAN es flexible y escalable. Angular es una muy buena opción para el desarrollo de una Simple Page Aplication y es responsiva. Comparamos la stack MEAN con otra stack muy popular, la stack Java EE, que tiene ya trabajando mucho tiempo con éxito durante décadas. Java EE también es muy fácil de aprender ya que tanto el front-end como el back-end están construidos usando un solo lenguaje de programación Java. Todos los componentes de Java EE son tecnologías ya maduras. En Java EE, front-end son capaces de entender el código de back-end y los desarrolladores de back-end son capaces de entender el código del front-end, utilizan Java como lenguaje principal y adicionalmente HTML/CSS que son lenguaje de marcado muy populares. JSP es utilizada para el desarrollo de aplicaciones web modernas y es responsiva. Apache/Tomcat es escalable y realiza en el manejo de llamadas asíncronas y en el manejo de muchas tareas concurrentes en un a través de multi-threading. En este estudio, nos centramos en la probar la stack MEAN frente a la stack Java EE con el sistema de la hoja de vida y gestión docente, construida para comparar la stack MEAN contra la stack Java EE. El sistema de la hoja de vida y gestión docente demostró el rendimiento, incluyendo el rendimiento con solicitud de respuesta con concurrencia, transferencia de datos y concurrencia. El rendimiento de la aplicación se midió con la ayuda de herramientas de evaluación comparativa. El tiempo de respuesta de la stack Java EE es mayor que en comparación con la stack MEAN. La transferencia de datos fue más baja en la stack MEAN en comparación con la stack Java EE. Y en la prueba de concurrencia se observa que la concurrencia es mejor en el stack MEAN ya que es menor en comparación con la stack Java EE. En este estudio, nos centramos

en la puesta en marcha en una aplicación real con las stacks MEAN y Java EE, dando como resultado que la stack MEAN sobresale en relación con la stack Java EE y en las que, la stack MEAN es preferible a la stack Java EE. Falta hacer mención a situaciones en la que MEAN no debe ser utilizada, puesto que se requiere de mayor cantidad de criterios de análisis y variables de medición. Un análisis de las deficiencias de la stack MEAN nos proporcionaría con una mejor comprensión MEAN y su aplicabilidad. Sin embargo, este documento demostró cómo la stack MEAN es buena para el desarrollo de aplicaciones web basados en servicios REST, hay limitaciones en nuestro estudio de caso. Por ejemplo, nuestro estudio de caso no muestra cómo MEAN reacciona con las aplicaciones intensivas y de gran procesamiento en CPU y memoria RAM. Por lo tanto, este documento podría beneficiarse de la introducción de un estudio de caso que se ocupe del uso intensivo y gran procesamiento de la CPU y memoria RAM.

Referencias

[1] J. Tihomirovs and J. Grabis, "Comparison of SOAP and REST Based Web Services Using Software Evaluation Metrics," Inf. Technol. Manag. Sci., vol. 19, no. 1, pp. 92–97, 2016.

[2] R. T. Fielding and R. N. Taylor, "Principled design of the modern Web architecture," ACM Trans. Internet Technol., vol. 2, no. 2, pp. 115–150, 2002.

[3] S. Mumbaikar and P. Padiya, "Web Services Based On SOAP and REST Principles," Int. J. Sci. Res. Publ., vol. 3, no. 5, pp. 3–6, 2013.

[4] M. Stajcer and D. Orescanin, "Using MEAN stack for development of GUI in real-time big data architecture," 2016 39th Int. Conv. Inf. Commun. Technol. Electron. Microelectron. MIPRO 2016 - Proc., pp. 524–529, 2016.

[5] A. J. Poulter, S. J. Johnston, and S. J. Cox, "Using the MEAN stack to implement a RESTful service for an Internet of Things application," IEEE World Forum Internet Things, WF-IoT 2015 - Proc., pp. 280–285, 2015.

[6] R. Salunkhe, S. Telang, P. Shrigondekar, and A. Tanpure, Review of REST Ful Service Using MEAN Stack for Real Time Big Data Architecture, vol. 3297, no. 11. Birmingham: Packt Publ, 2007.

[7] N. Rozanski and E. Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley, 2005.

[8] P. Clements, R. Kazman, and M. Klein, Evaluating Software Architectures: Methods and Case Studies. Addison-Wesley Professional, 2001.

[9] D. Garlan and M. Shaw, "An Introduction to Software Architecture," Knowl. Creat. Diffus. Util., vol. 1, no. January, pp. 1–40, 1994.

[10] S. Engineering and S. Committee, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems," 2000.

[11] D. Garlan and D. Garlan, "Software Architecture: a Roadmap Software Architecture: a Roadmap," 2000.

[12] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice, Second Edi. Addison Wesley, 2003.

[13] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, Pattern-Oriented Software Architecture - Volume 1: A System of Patterns. Wiley Publishing, 1996.

[14] F. Bocchio, "Estudio Comparativo De Plataformas Cloud Computing Para Arquitecturas Soa," vol. 1, no. 5, p. 114, 2013.

[15] S. Malik and D. H. Kim, "A comparison of RESTful vs. SOAP web services in actuator networks," Int. Conf. Ubiquitous Futur. Networks, ICUFN, pp. 753–755, 2017.

[16] A. Arcuri, "RESTful API Automated Test Case Generation," 2017 IEEE Int. Conf. Softw. Qual. Reliab. Secur., pp. 9–20, 2017.

[17] D. Booth et al., "Web Services Architecture," W3C Note, vol. 22, no. February, pp. 1–98, 2004.

[18] K. Gottschalk, S. Graham, H. Kreger, and J. Snell, "Introduction to Web services architecture," IBM Syst. J., vol. 41, no. 2, pp. 170–177, 2002.

[19] R. T. Fielding, "Architectural Styles and the Design of Network-based Software Architectures," Building, vol. 54, p. 162, 2000.

[20] M. Zur Muehlen, J. Nickerson, and K. Swenson, "Developing web services choreography standards - The case of REST vs. SOAP," Decis. Support Syst., vol. 40, no. 1 SPEC. ISS., pp. 9–29, 2005.

[21] D. Fensel, F. M. Facca, E. Simperl, and I. Toma, Semantic Web Services, 1st ed. Springer Publishing Company, Incorporated, 2011.

[22] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. 'big'web services: making the right architectural decision," Proceeding 17th Int. Conf. World Wide Web, pp. 805–814, 2008.

[23] S. Patni, Pro RESTful APIs. 2017.

[24] D. Flanagan, JavaScript - The Definitive Guide. 2011.

[25] A. Q. Haviv, MEAN Web Development, vol. 1. 2014.

[26] S. Holmes, Getting MEAN with Mongo, Express, Angular, and Node, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2015.

[27] K. Chodorow, Mongo DB: The Definitive Guide. 2013.

[28] R. O. Obe and L. S. Hsu, MongoDB in Action. 2011.

[29] E. Brown, Web Development with Node and Express: Leveraging the JavaScript Stack. O'Reilly Media, 2014.

[30] E. Hahn, Express in Action: Node Applications with Express and Its Companion Tools, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2015.

[31] R. K. Soni, Full Stack AngularJS for Java Developers. 2017.

[32] S. Davis, "Mastering MEAN: Introducing the MEAN stack," IBM.com, pp. 1–20, 2014.

[33] M. Cantelon, M. Harter, T. J. Holowaychuk, and N. Rajlich, Node.Js in Action, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2013.

[34] J. Bloch, Effective Java (2Nd Edition) (The Java Series), 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008.

[35] S. Daschner, Architecting Modern Java EE Applications: Designing lightweight, business-oriented enterprise applications in the age of cloud, containers, and Java EE 8. Packt Publishing, 2017.

[36] F. Gutierrez, Pro Spring Boot. 2016.

[37] J. Brisbin, O. Gierke, and G. Turnquist, "Spring Data REST - Reference Documentation," 2015. [Online]. Available: https://docs.spring.io/spring-data/rest/docs/current/reference/html/. [Accessed: 06-Apr-2018].

[38] B. Muschko, Gradle in Action, 1st ed. Greenwich, CT, USA: Manning Publications Co., 2014.

[39] S. Daschner, Architecting Modern Java EE Applications: Designing lightweight, business-oriented enterprise applications in the age of cloud, containers, and Java EE 8. Packt Publishing, 2017.

[40] W3C, "HTML 5.2 Recomendation," 2017. [Online]. Available: https://www.w3.org/TR/html52/. [Accessed: 06-May-2018].

[41] J. D. Gauchat, El gran libro de HTML5. CSS3 y Javascript, vol. 354, no. 9. 2012.

[42] B. Noel and A. Chopra, "Introduction To Javaserver Pages," 2001.

[43] "Siege Home," 2012. [Online]. Available: https://www.joedog.org/siege-home/. [Accessed: 14-May-2018].

[44] "Apache HTTP server benchmarking tool." [Online]. Available: https://httpd.apache.org/docs/2.4/programs/ab.html. [Accessed: 14-May-2018].

[45] "Apache JMeter - Apache JMeterTM." [Online]. Available: https://jmeter.apache.org/. [Accessed: 13-May-2018].

[46] "Prueba de rendimiento usando Jmeter." [Online]. Available: https://www.guru99.com/jmeter-performance-testing.html. [Accessed: 14-May-2018].

[47] "Load Testing and Benchmarking With Siege." [Online]. Available: https://www.serverwatch.com/tutorials/article.php/3936526/Load-Testing-and-Benchmarking-With-Siege.htm. [Accessed: 14-May-2018].