Análisis comparativo de dos stacks tecnológicos para el desarrollo de aplicaciones web: LAMP contra MEAN

Carlos Giovanni Molina Ronceros[1]

[1] University of Alcalá de Henares, Madrid, SPAIN

[1] {giovanni.molina}@edu.uah.es

Resumen: El desarrollo de aplicaciones web es uno de los sectores laborales dentro de las Tecnologías de la Información (TI) que más crece actualmente. Empresas y trabajadores disponen de un gran abanico de tecnologías y enfoques que pueden utilizar, lo que en ocasiones supone un problema, por desconocer por cuál de ellas apostar dependiendo de las características de sus proyectos. El presente estudio analiza a diversos niveles los stacks tecnológicos LAMP y MEAN en lo que respecta al desarrollo de una API REST y una aplicación cliente que consume las operaciones expuestas por ésta. Se identificarán las ventajas y desventajas de cada uno de los stacks y de las herramientas que los conforman, y se obtendrán conclusiones acerca de en qué escenarios es más recomendable utilizar uno u otro stack.

Palabras clave: Desarrollo web, LAMP, MEAN, PHP, JavaScript, Laravel, Lumen, Express.js, Angular, Node.js, API REST, Nginx, MySQL, Docker.

1 Introducción

Actualmente los puestos de desarrollador e ingeniero del software de aplicaciones web son algunos de los más demandados en el mercado laboral [1]. Además, diversos estudios dentro del sector TI [2, 3] predicen que seguirán siendo muy solicitados en los próximos años. Esto significa que las pequeñas y grandes empresas ya consolidadas y las startups emergentes están apostando fuertemente por desarrollar sus actividades dentro de Internet.

En principio este nuevo enfoque empresarial resulta beneficioso para todos: los trabajadores del sector TI se encuentran con una gran oferta de empleos bien remunerados; las empresas pueden llegar a obtener grandes beneficios económicos ya que al ofrecer sus servicios a través de la Web disponen de un mercado más amplio de consumidores; y por último los usuarios tienen una amplia gama de proveedores donde elegir a solo un par de clics de distancia.

El problema llega cuando tanto empresa como desarrollador tienen que seleccionar con qué tecnologías trabajar. En el mundo de las aplicaciones web, el número de herramientas y enfoques disponibles es enorme: lenguajes de programación (Java, PHP, Python, Ruby, C#, JavaScript, etc.), entornos de ejecución (Node.js, JVM, CLR, etc.), frameworks backend (Spring, Laravel, Django, Ruby on Rails, .NET,

Express.js, etc.), frameworks frontend (Angular, React, Vue, etc.), servidores web (Nginx, Apache, IIS, etc.), bases de datos (MySQL, PostgreSQL, MongoDB, etc.). Elegir un conjunto de estas tecnologías para empezar un nuevo proyecto o formarse para un futuro empleo puede resultar una tarea muy complicada.

El presente estudio pretende ayudar a empresas y trabajadores a identificar las ventajas y desventajas de apostar por dos de los stacks tecnológicos más utilizados actualmente: el clásico LAMP [4], en su variación **LEMP** (Linux, Nginx, MySQL y PHP) y el prometedor **MEAN** [5] (MySQL en vez de MongoDB, Express.js, Angular y Node.js).

2 Objetivos

El principal objetivo de este estudio es realizar un análisis comparativo de los stacks tecnológicos LEMP y MEAN, orientados al desarrollo de dos tipos de aplicaciones web: una API REST y un cliente web que ofrezca un interfaz gráfica de usuario (GUI) para trabajar con las operaciones expuestas por la API. Los resultados obtenidos servirán a desarrolladores y empresas en el momento de decidir qué tecnologías usar en la implementación de sus proyectos TI. Es importante señalar que en ningún momento se pretende afirmar cual de los dos stacks tecnológicos es mejor. Como se verá en las conclusiones del presente trabajo, cada uno de ellos y las herramientas que los conforman tienen sus fortalezas y debilidades, y su elección dependerá más de las características del proyecto a desarrollar.

El objetivo secundario es verificar el estado del arte de las herramientas PHP y JavaScript disponibles para el desarrollo de aplicaciones web dentro de los ecosistemas LEMP y MEAN.

3 Tecnología utilizada

Para los elementos de los stacks LEMP y MEAN en los que se puede elegir entre diversas opciones se ha tenido en cuenta las recomendadas en el proyecto *Roadmap to becoming a web developer in 2018* [6]. Por otro lado también se ha valorado qué tecnologías se suelen utilizar en entornos reales de producción en lo que respecta a sistemas operativos y contenedores de ejecución.

En primer lugar, se decidió utilizar Docker como sistema para desplegar las aplicaciones desarrolladas. Con Docker se pueden crear imágenes que contengan todo el código de nuestra aplicación y el entorno necesario para ejecutarla. De esta forma nos aseguramos que los contenedores creados a partir de dichas imágenes se ejecutan siempre correctamente, independientemente de la plataforma donde realicemos el despliegue.

En segundo lugar se decidió utilizar Linux como sistema operativo base de todos los contenedores necesarios para crear el sistema completo. Las distribuciones Linux seleccionadas fueron Alpine y Debian Stretch Slim, por ser de las más ligeras a la vez que ofrecen lo necesario para ejecutar aplicaciones PHP y Node.js, y ser capaces de correr servidores web y de base de datos.

3.1 Tecnologías utilizadas en el stack LEMP

- **L:** Linux Alpine como sistema operativo.
- **E:** Nginx como servidor web HTTP.
- **M:** MySQL como sistema gestor de base de datos.
- **P:** lenguaje PHP v7.2, con el microframework **Lumen** para el desarrollo de la API REST y el framework **Laravel** para el desarrollo del cliente web.

3.2 Tecnologías utilizadas en el stack MEAN

- **M:** MySQL como SGBD, en detrimento de MongoDB.
- **E:** framework **Express.js**, que se utilizará para crear la API REST.
- **A:** framework **Angular**, que se utilizará para crear el cliente web.
- **N:** Node.js, que se utilizará como entorno de ejecución de la API REST.

4 Especificación de requisitos y diseño del prototipo de pruebas

4.1 Funcionalidades del prototipo de pruebas

El sistema desarrollado para realizar el análisis comparativo tendrá las siguientes funcionalidades:

- **Acceso multiusuario:** conexión simultánea de usuarios previa autenticación.
- **Gestión de permisos basada en perfiles:** cada perfil tendrá acceso de forma exclusiva a ciertas operaciones o secciones del sistema. Los perfiles serán administrador, instructor y usuario.
- **Gestión de exámenes:** los instructores podrán crear exámenes en los grupos de trabajo a los que tienen acceso. Los usuarios podrán realizar los exámenes disponibles en sus grupos de trabajo. Los administradores gestionarán el alta, baja y modificación de instructores y usuarios, y el acceso a los distintos grupos de trabajo existentes.

4.2 Análisis alto nivel

Para llevar a cabo un estudio lo más justo posible se decidió respetar los siguientes principios:

- Los sistemas a implementar con ambos stacks deberán compartir la misma base de datos, ya que se desea que la información expuesta por ambos sea idéntica. Como no es correcto comparar en rendimiento una base de datos relacional (MySQL del stack LEMP) con una no-relacional (MongoDB del stack MEAN), atendiendo a los requisitos funcionales del prototipo, se decidió utilizar en los dos stacks MySQL.

- En cada sistema deberán existir dos capas:
  - Capa 1 – API: aplicación que se encargará de realizar operaciones CRUD en la base de datos.
  - Capa 2 – APP: aplicación donde residirá la interfaz gráfica con la que trabajarán los usuarios finales. Se conectará con la API para leer o persistir datos.
- Las capas del mismo nivel deberán ser intercambiables de un sistema a otro. Por ejemplo, el cliente web del stack LEMP podrá trabajar con la API del stack MEAN y viceversa.

4.3 Diseño del sistema

Teniendo en cuenta que se desea crear un sistema que en un futuro se pueda desplegar en un entorno de producción, se decidió diseñar uno con los siguientes componentes:

- **Servidor Base de Datos:** máquina/contenedor con sistema operativo Debian Stretch Slim donde estará ejecutándose el SGBD MySQL versión 5.7.
- **Servidor Web API Lumen:** máquina/contenedor con sistema operativo Alpine 3.6 que tendrá los siguientes subcomponentes:
  - **PHP 7.2 FPM:** servicio que se encargará de interpretar el código PHP de la API desarrollada con el framework **Lumen**. Si es necesario, se conectará al servidor de base de datos.
  - **Nginx 1.12:** servidor web que se encargará de gestionar las peticiones HTTP. Interactuará con el servicio de PHP FPM mediante un socket TCP/IP.
- **Servidor Web APP Laravel:** máquina/contenedor con sistema operativo Alpine 3.6 que tendrá los siguientes subcomponentes:
  - **PHP 7.2 FPM:** servicio que se encargará de interpretar el código PHP de la GUI desarrollada con el framework **Laravel**. Si es necesario, se conectará mediante CURL al servidor web con la API Lumen.
  - **Nginx 1.12:** servidor web que se encargará de gestionar las peticiones HTTP. Interactuará con el servicio de PHP FPM mediante un socket TCP/IP.
- **Servidor Web API Express:** máquina/contenedor con sistema operativo Alpine 3.6 donde estará ejecutándose la API desarrollada con el framework **Express.js**. Node.js v8.11 se encargará de realizar las conexiones con el servidor de base de datos y de gestionar las peticiones HTTP.
- **Servidor Web APP Angular:** máquina/contenedor con sistema operativo Alpine 3.7 donde estará ejecutándose el servidor web Nginx 1.15. Éste se encargará de servir la GUI desarrollada con el framework **Angular**.
- **Navegador Web:** utilizado por el usuario para trabajar con los sistemas desarrollados.

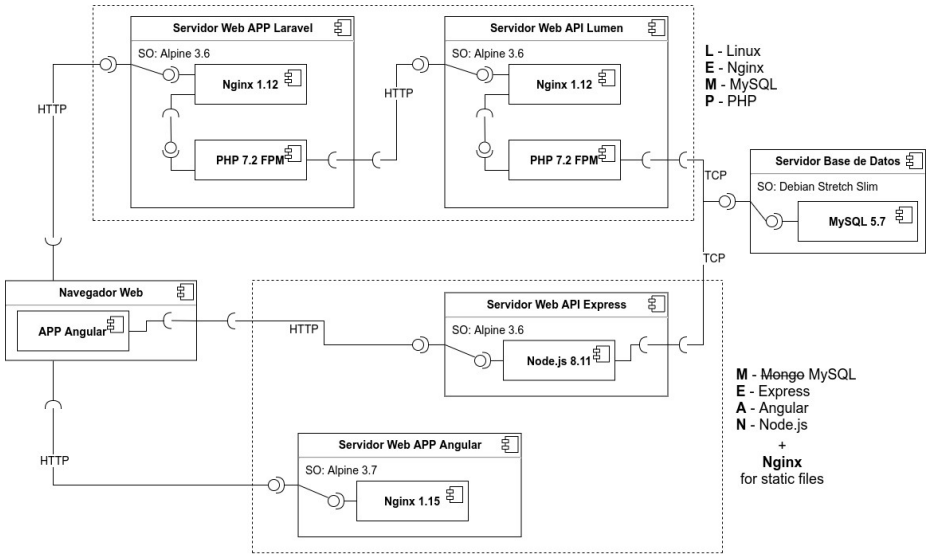

Fig. 1. Diagrama de componentes del sistema.

5 Criterios de comparación

Para realizar el enfrentamiento de los stacks tecnológicos se establecieron las siguientes categorías de comparación a nivel de frameworks utilizados:

- **Categoría A: Documentación disponible.** Se analiza la calidad de la documentación oficial y no oficial disponible.
- **Categoría B: Comunidad de usuarios y desarrolladores.** Se analiza la comunidad de usuarios y eventos relaciones.
- **Categoría C: Estado del proyecto.** Se analiza la actividad y popularidad del proyecto dentro de la comunidad de desarrollo web.
- **Categoría D: Recursos necesarios para preparar un entorno de ejecución.** Analiza las herramientas y recursos necesarios para construir un entorno de ejecución del framework utilizado.
- **Categoría E: Dificultad para crear proyecto "Hello world".** Analiza la documentación disponible y la dificultad para crear un proyecto base.
- **Categoría F: Recursos y funcionalidades que provee el framework.** Analiza la disponibilidad de los componentes más necesarios para implementar los sistemas diseñados.
- **Categoría G: Curva de aprendizaje.** Analiza el grado de dificultad en dominar el framework.
- **Categoría H: Rendimiento.** Analiza el rendimiento de la aplicación creada.
- **Categoría I: Coste despliegue en producción.** Analiza el coste de desplegar la aplicación creada en un entorno real de producción.

Cada uno de las categorías anteriormente citadas dispone de una serie de criterios de comparación para poder realizar un análisis más refinado. Los elementos de los stacks que se compararon fueron Lumen y Express a nivel de API REST y Laravel y Angular a nivel de cliente web.

6 Resultados obtenidos

Tabla 1. Implementación API REST: Lumen contra Express.

Categoría comparación	**Nº de criterios en los que gana Lumen 5.6**	**Nº de criterios en los que gana Express 4.16**	**Nº de criterios en los que empatan**
A: Documentación disponible	0	0	4
B: Comunidad de usuarios y desarrolladores	0	4	1
C: Estado del proyecto	0	2	0
D: Recursos necesarios para preparar un entorno de ejecución	0	2	1
E: Dificultad para crear proyecto “Hello world”	2	0	1
F: Recursos y funcionalidades que provee el framework	4	0	3
G: Curva de aprendizaje	1	0	1
H: Rendimiento	2	1	1
I: Coste despliegue en producción	0	1	1
Total	**9**	**10**	**13**

Tabla 2. Implementación aplicación web que interactúa con una API: Laravel contra Angular.

Categoría comparación	**Nº de criterios en los que gana Laravel 5.6**	**Nº de criterios en los que gana Angular 6.1**	**Nº de criterios en los que empatan**
A: Documentación disponible	0	0	4
B: Comunidad de usuarios y desarrolladores	1	2	2
C: Estado del proyecto	0	2	0
D: Recursos necesarios para preparar un entorno de ejecución	0	2	1
E: Dificultad para crear proyecto “Hello world”	1	0	2
F: Recursos y funcionalidades que provee el framework	1	0	5
G: Curva de aprendizaje	1	1	0

H: Rendimiento	0	1	2
I: Coste despliegue en producción	0	1	1
Total	**4**	**9**	**17**

7 Conclusiones

En la implementación de una API REST se establecieron 32 criterios de comparación, en los cuales Lumen ganó en 9, Express en 10 y terminaron empatando en 13 de dichos criterios. Express se lleva la victoria, pero por muy poco. Por lo tanto, para implementar una API REST, Express.js es más recomendable que Lumen.

En la implementación de una aplicación web de tipo GUI que interactúa con una API REST se establecieron 30 criterios de comparación, en los cuales Laravel ganó en 4, Angular en 9 y terminaron empatando en 17 de dichos criterios. Angular se lleva la victoria claramente en este tipo de aplicación. Por lo tanto podemos afirmar que para implementar una aplicación web de tipo GUI que interactúa con una API REST, Angular es mucho más recomendable que Laravel.

Finalmente vamos a hacer una valoración global de cada uno de los stacks tecnológicos. Entre las principales ventajas de utilizar el stack LEMP, tenemos las siguientes:

- Fácil de aprender. Se puede tener un producto funcional en poco tiempo.
- Está ampliamente probado en entornos reales de producción.0
- Es muy flexible, ya que podemos elegir que tecnologías utilizar en cada capa. Algunos ejemplos son LAMP, WAMP, LAPP, etc.
- Si somos una empresa y buscamos empleados que hayan trabajado con este stack no tendremos dificultades en encontrar perfiles con una amplia experiencia y sueldos razonables.

Y entre sus principales desventajas tenemos:

- Si queremos desarrollar una aplicación compleja y que vaya a prestar servicio a un gran número de usuarios, debemos contar con expertos en diversas áreas.
- PHP ha mejorado mucho en sus últimas versiones, pero sigue siendo un lenguaje que no proporciona un gran rendimiento.
- En servicios cloud, no escala tan bien como MEAN.

Respecto al stack MEAN, entre sus principales ventajas tenemos:

- Solo es necesario aprender JavaScript para dominar el backend y frontend de nuestras aplicaciones. Todo está implementado con dicho lenguaje.
- La transferencia de datos entre capas es muy rápida, ya que se utiliza el mismo formato (JSON) en todas ellas.
- Es posible implementar interfaces de usuario muy potentes, ya que el uso de frameworks que proveen esta funcionalidad está muy integrado en el stack.
- Node.js está preparado para escalar horizontalmente.

Y entre sus principales desventajas tenemos:

- El stack es relativamente nuevo, así que no tenemos el mismo soporte técnico ofrecido por LEMP.
- Es más difícil encontrar expertos en este stack, por lo que los sueldos que habrá que pagar por ellos será más alto que la media.

Como conclusión final, no recomendaría la utilización de uno de los dos stacks en detrimento del otro, ya que cada uno tiene sus propias fortalezas, que pueden ser aprovechadas dependiendo del tipo de proyecto software que se quiere implementar.

Por ejemplo recomendaría LEMP para los siguientes tipos de proyectos:

- Soluciones empresariales en las que no se dé tanta importancia a la interfaz de usuario pero sí a que el sistema funcione como se espera y que se asegure la integridad de los datos almacenados.
- Servicios web que no trabajen con datos tipo JSON (por ejemplo SOAP).
- Proyectos en los que se disponga de un presupuesto económico moderado.

Por otro lado recomendaría MEAN para los siguientes tipos de proyectos:

- Aplicaciones que necesitan manejar una gran cantidad de operaciones I/O, streaming, etc.
- APIs REST que trabajan con JSON.
- SPA (Single Page Applications).
- Aplicaciones que necesiten una gran experiencia de usuario.
- Aplicaciones que se van a desplegar en sistemas cloud.
- Proyectos en los que el aspecto económico no sea un impedimento y demos más importancia a trabajar con las últimas tecnologías.

Y para terminar, como observación y consejo personal para un ingeniero del software, recomendaría que se aprendiese ambos stacks y, siempre que fuese posible, se combinasen elementos de los dos para crear sistemas más robustos y completos.

Referencias

1. IEBS, Escuela de Negocios de la Innovación y los Emprendedores. Profesiones digitales mejor pagadas y más demandadas 2018. https://www.iebschool.com/blog/profesiones-digitales-mas-demandados-digital-business/. (Consultado el 13 de septiembre de 2018).
2. Business Insider. Estos son los empleos emergentes con más futuro de España y su salario. https://www.businessinsider.es/estos-son-empleos-emergentes-mas-futuro-espana-su-salario-253528. (Consultado el 13 de septiembre de 2018).
3. Business Insider. Information technology is one of the fastest-growing industries. https://www.businessinsider.es/it-jobs-information-technology-2018-5. (Consultado el 13 de septiembre de 2018).
4. Wikipedia. LAMP. https://es.wikipedia.org/wiki/LAMP. (Consultado el 13 de septiembre de 2018).
5. Wikipedia. MEAN. https://es.wikipedia.org/wiki/MEAN. (Consultado el 13 de septiembre de 2018).
6. GitHub. Roadmap to becoming a web developer in 2018. https://github.com/kamranahmedse/developer-roadmap.(Consultado el 13 de septiembre de 2018).