ACTUALIZACIÓN DE NODE, RETOS Y DESAFÍOS

Anthony Eduardo Ortega Cruz, OC200384
José Manuel Rodríguez Lovo, RL190301
Carlos Manuel, Cuestas Aguilar CA201748
Jocelyn Susana, Aguilar Corvera, AC180847
Santos Ronaldo Lemus Torres, LT191211

ARTICULO TÉCNICO

UNIVERSIDAD DON BOSCO

FACULTAD DE ING. CC. COMPUTACIÓN

SAN SALVADOR

2023

ACTUALIZACIÓN DE NODE, RETOS Y DESAFÍOS

Anthony Eduardo Ortega Cruz, OC200384
José Manuel Rodríguez Lovo, RL190301
Carlos Manuel, Cuestas Aguilar CA201748
Jocelyn Susana, Aguilar Corvera, AC180847
Santos Ronaldo Lemus Torres, LT191211

RESUMEN

La actualización de tecnologías en las empresas es esencial para la competitividad. Este artículo explora el desafío de actualizar Node.js de la versión 14.20.1 a la 18.18.2 en un ERP internacional basado en React.js, Node.js y MongoDB. Se destacan desafíos técnicos como la compatibilidad de módulos, estabilidad, rendimiento y seguridad. Sin embargo, esta actualización promete mejoras en eficiencia, escalabilidad y aprovechamiento de nuevas características. La empresa busca mantenerse competitiva y mejorar la experiencia del usuario en un mercado en constante cambio. Este estudio de caso ilustra cómo las empresas enfrentan y aprovechan la incorporación de tecnologías emergentes en entornos empresariales dinámicos.

Palabras claves: Node.js; Actualización; Rendimiento; Seguridad; Módulos; Bibliotecas; ERP; Usuario; Versiones

ABSTRACT

Incorporating new technologies is imperative for business competitiveness. This article explores the challenge of upgrading Node.js from version 14.20.1 to 20.8.0 in an international ERP based on React.js, Node.js, and MongoDB. Technical challenges, including module compatibility, stability, performance, and security, are highlighted. However, this update promises improvements in efficiency, scalability, and the utilization of new features. The company aims to remain competitive and enhance the user experience in an ever-changing market. This case study illustrates how companies confront and leverage the integration of emerging technologies in dynamic business environments.

Keywords: Node.js; Upgrade; Performance; Security; Modules; Libraries; ERP; User; Versions

I. INTRODUCCIÓN

Node.js se ha convertido en una tecnología fundamental en el desarrollo de aplicaciones web y servidores en la última década. Su evolución constante y la disponibilidad de nuevas versiones han llevado a muchas empresas a plantearse la actualización de sus aplicaciones y sistemas existentes. Sin embargo, la actualización de Node.js no es una tarea trivial y conlleva riesgos y desafíos que deben ser cuidadosamente considerados.

Este artículo técnico abordará de manera integral el proceso de actualización de Node.js en entornos empresariales, examinando los riesgos, los beneficios y las consideraciones clave que deben tenerse en cuenta. Comprender las implicaciones de esta actualización es fundamental para tomar decisiones y garantizar que la empresa pueda aprovechar al máximo las nuevas características y mejoras de Node.js sin poner en peligro la estabilidad y la seguridad de sus aplicaciones existentes.

Exploraremos los riesgos potenciales de actualizar Node.js, desde la incompatibilidad de módulos hasta posibles problemas de seguridad. A su vez, destacaremos los beneficios, como mejoras de rendimiento, nuevas características y soporte a largo plazo. Además, consideraremos los aspectos más críticos a tener en cuenta durante el proceso de actualización, incluyendo estrategias de migración, pruebas exhaustivas y la gestión de dependencias.

II. METODOLOGÍA

a) Node.js

Node.js es un entorno de ejecución de JavaScript de código abierto principalmente utilizado para crear aplicaciones del lado del servidor. Las características y conceptos principales de Node.js incluyen:

- Modelo de E/S no bloqueante
- NPM (Node Package Manager
- Event Loop
- Módulos y CommonJS
- Amplia comunidad y ecosistema
- Uso versátil
- Compatibilidad con múltiples plataformas

Debido a su escalabilidad, eficiencia y facilidad de uso, ganó una gran popularidad en la comunidad de desarrollo de software. La mayoría de las empresas y startups lo utilizan para crear aplicaciones web y servidores de aplicaciones en tiempo real.

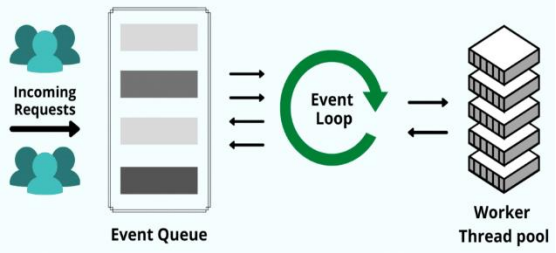

Figura I. Arquitectura básica de Node.js [5]

b) Node.js en un entorno de desarrollo

Node.js es una tecnología versátil que se adapta a una variedad de casos de uso en una empresa, ya que es eficiente, escalable y tiene una gran comunidad de desarrolladores. Su capacidad para manejar conexiones en tiempo real y E/S no bloqueante lo hace especialmente útil en aplicaciones que requieren interacción en tiempo real o un alto rendimiento. Entre las utilidades más frecuentes:

- *Desarrollo web*
- APIs y servicios web:
- Aplicaciones móviles
- Aplicaciones de escritorio
- Automatización de tareas
- IoT (Internet de las cosas)
- Procesamiento de datos en tiempo real
- Aplicaciones de mensajería y chat en vivo
- Aplicaciones de streaming de medios
- Administración de servidores

- Aplicaciones de juegos en tiempo real
- Análisis de datos en tiempo real

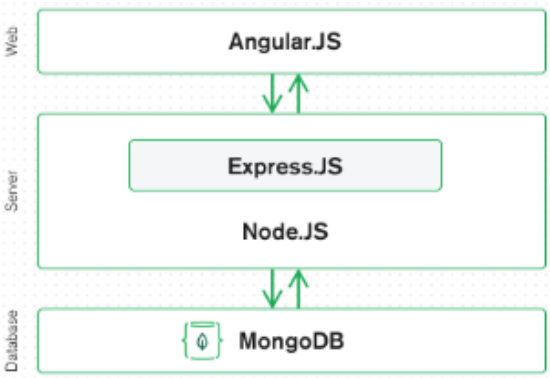

Figura 2. Flujo básico de una implementación de desarrollo con Node.js [6]

c) Riesgos de actualización

La actualización de la versión de Node.js puede causar varios problemas, ya que las nuevas versiones pueden incluir cambios significativos en el motor V8 de JavaScript y en las bibliotecas internas de Node.js. entre las que se incluyen:

Incompatibilidad de API: Las nuevas versiones de Node.js pueden introducir cambios en la API, lo que significa que algunas funciones o métodos que eran válidos en versiones anteriores pueden volverse obsoletos o cambiar su comportamiento.

Incompatibilidad de módulos o paquetes de terceros: Al actualizar Node.js, algunos módulos o paquetes de terceros que dependen de características específicas de una versión anterior de Node.js pueden dejar de funcionar correctamente.

Problemas de rendimiento: Si bien las nuevas versiones de Node.js generalmente incluyen mejoras de rendimiento, a veces pueden surgir problemas de rendimiento debido a cambios en el motor V8 de JavaScript o en otros componentes internos.

Errores de compilación: Las actualizaciones de Node.js pueden requerir que las dependencias binarias (aquellas que deben compilarse) se vuelvan a compilar. Si no se actualizan correctamente, esto puede llevar a errores de compilación o a que los módulos no se carguen adecuadamente.

Compatibilidad del sistema operativo: Algunas versiones de Node.js pueden requerir versiones específicas del sistema operativo o de las bibliotecas del sistema.
Problemas de configuración: Las configuraciones o ajustes específicos que funcionaban en una versión anterior pueden necesitar modificaciones en la nueva versión. Por ejemplo, la configuración del equilibrio de carga o las opciones de inicio.

d) Beneficios de la actualización

La actualización puede ofrecer varios beneficios importantes en un entorno de desarrollo. Algunos de los beneficios clave de actualizar Node.js incluyen:
Mejoras en el rendimiento: con frecuencia hay mejoras en el rendimiento en las nuevas versiones de Node.js, lo que significa que su aplicación puede volverse más rápida y eficiente. Esto es particularmente importante para las aplicaciones web y servidores que deben manejar grandes cantidades de trabajo.

Seguridad adicional: las actualizaciones más recientes de Node.js generalmente incluyen correcciones de seguridad para corregir vulnerabilidades conocidas. La actualización a la versión más reciente protege su aplicación de las amenazas de seguridad más recientes.

Soporte de características JavaScript modernas: Las nuevas versiones de Node.js suelen soportar las características JavaScript más recientes, lo que facilita el desarrollo de código más limpio y moderno. Esto puede hacer que el código sea más fácil de leer y mantener.

Nuevas características y API: Las actualizaciones a menudo introducen nuevas características y mejoras en la API

que pueden simplificar el desarrollo de aplicaciones y permitir la adopción de prácticas recomendadas más actualizadas.

Corrección de errores y problemas conocidos: Las versiones más nuevas de Node.js a menudo incluyen correcciones para errores y problemas conocidos que se han identificado en versiones anteriores. Esto puede mejorar la estabilidad de las aplicaciones.

Compatibilidad con los módulos y bibliotecas más recientes: actualizar Node.js garantiza que las bibliotecas y módulos de terceros sigan siendo compatibles con su entorno. Las últimas versiones de paquetes de terceros pueden no funcionar con las versiones más antiguas.

Mejorar la administración de paquetes: el administrador de paquetes de Node.js, NPM, se actualiza y mejora regularmente, lo que facilita la gestión de las dependencias de su proyecto.

Soporte a largo plazo (LTS): Las versiones LTS de Node.js reciben actualizaciones de seguridad y soporte a largo plazo durante un período de tiempo prolongado, lo que garantiza que las aplicaciones estén respaldadas y seguras durante más tiempo.

e) Consideraciones al actualizar Node.js

Al actualizar Node.js en una empresa, es fundamental realizar pruebas exhaustivas en un entorno de desarrollo antes de considerar la implementación en un entorno de producción. Durante dicho proceso, es esencial verificar la compatibilidad de todas las dependencias, como podrían ser los paquetes de terceros, para asegurarse de que funcionen sin problemas en la nueva versión. Además de ello, es importante revisar detenidamente las notas de la versión de Node.js para comprender los cambios clave que puedan afectar al desarrollo de aplicaciones. La utilización de herramientas como NVM (Node Version Manager) puede simplificar la gestión de versiones y facilitar la restauración en caso de presentar problemas.
Se recomienda hacer copias de seguridad del código y datos de todos los proyectos y dependencias antes de la actualización para minimizar riesgos. Finalmente, es esencial tener una estrategia de implementación planificada y realizar un seguimiento cuidadoso de los proyectos después de la actualización para garantizar un funcionamiento estable y sin fallos que puedan poner en riesgo la integridad de los proyectos.

III. RESULTADOS

a) Rendimiento

En la actualización a la última versión de Node.js, la 18.18.2 LTS, hemos experimentado mejoras significativas en el rendimiento y la optimización. Esta versión más reciente ha contribuido a hacer que nuestra aplicación sea más ágil y eficiente.

b) Nuevas características

Con cada actualización de Node.js, se incorporan innovadoras características. Una de las que hemos encontrado particularmente valiosa es el "Ejecutor de Pruebas Estable". Esta función nos ha permitido crear y ejecutar conjuntos de pruebas en JavaScript de manera rápida y eficiente, sin necesidad de instalar dependencias adicionales.

c) Soporte de Seguridad

Dado que versiones antiguas de Node.js a menudo presentan vulnerabilidades de seguridad conocidas, la transición a una versión más reciente ha significado un resultado beneficioso. Al adoptar una versión actualizada de Node.js, hemos reforzado la seguridad de la empresa.

d) Compatibilidad con nuevas librerías y módulos

Antes de la actualización, nos enfrentábamos con desafíos recurrentes al intentar incorporar bibliotecas novedosas en nuestra aplicación. Sin embargo, con la transición a la versión más reciente de Node.js, hemos abierto la puerta a la instalación de bibliotecas con funcionalidades avanzadas, enriqueciendo significativamente nuestro sistema.

IV. ANÁLISIS

El análisis de todo el proceso, desde la identificación del desafío hasta la implementación y los resultados obtenidos, arroja una panorámica de mejora significativa en varios aspectos clave. La actualización a la versión 18.18.2 LTS de Node.js ha demostrado ser un paso crucial para impulsar el rendimiento y la eficiencia de nuestra aplicación empresarial.

La respuesta más ágil y eficiente que esta nueva versión proporciona se traduce en una experiencia de usuario notablemente mejorada. Además, la incorporación de características innovadoras, como el "Ejecutor de Pruebas Estable", ha fortalecido la capacidad para desarrollar y ejecutar conjuntos de pruebas en JavaScript de manera rápida y eficiente, reduciendo la necesidad de depender de dependencias adicionales.

La seguridad es un aspecto de suma importancia, y la actualización a la versión más reciente de Node.js ha contribuido significativamente la tranquilidad de la empresa en este sentido. Al migrar a esta versión actualizada, se ha reforzado la protección de sus datos, permitiéndoles mitigar vulnerabilidades conocidas, manteniéndose un paso adelante ante las amenazas de seguridad en constante evolución.

Por último, la compatibilidad con nuevas bibliotecas y módulos ha sido una transformación notable. En el pasado, se presentaban obstáculos recurrentes al intentar incorporar bibliotecas con funcionalidades avanzadas. Sin embargo, la actualización ha eliminado estas barreras y ha abierto la puerta a la integración de bibliotecas de última generación. Esto ha enriquecido sustancialmente el entorno de desarrollo de la empresa, permitiendo mantenerse al día con las tendencias tecnológicas más recientes y brindando a sus usuarios una experiencia más completa y enriquecedora. En resumen, la actualización de Node.js ha resultado en una mejora global del sistema ERP, respaldando la competitividad de la empresa en un mercado en constante evolución.

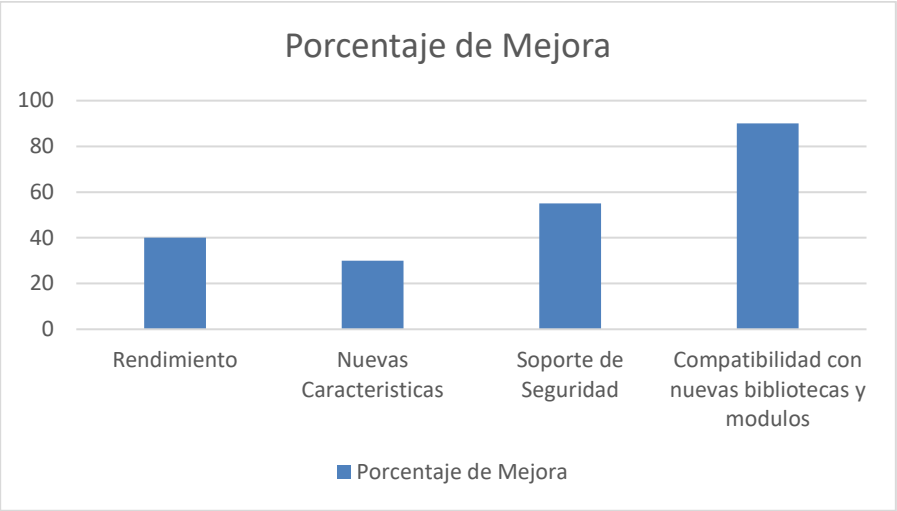

Figura III. Gráfico de barras referente a los porcentajes de mejora al realizar la actualización

ASPECTO	VERSIÓN 14.20.1	VERSIÓN 18.18.2
Rendimiento	Moderado	Mejorado
Seguridad	Riesgos conocidos	Reforzada
Nuevas Características	Limitadas	Incluye el Ejecutor de Pruebas Estable
Compatibilidad de módulos	Limitada	Mejorada
Bibliotecas Adicionales	Poca variedad	Mayor variedad
Eficiencia	Moderada	Optimizada
Escalabilidad	Limitada	Mejorada
Experiencia de usuario	Estable	Optimizada y mejorada

Tabla I. Tabla comparativa de las versiones de Node.js

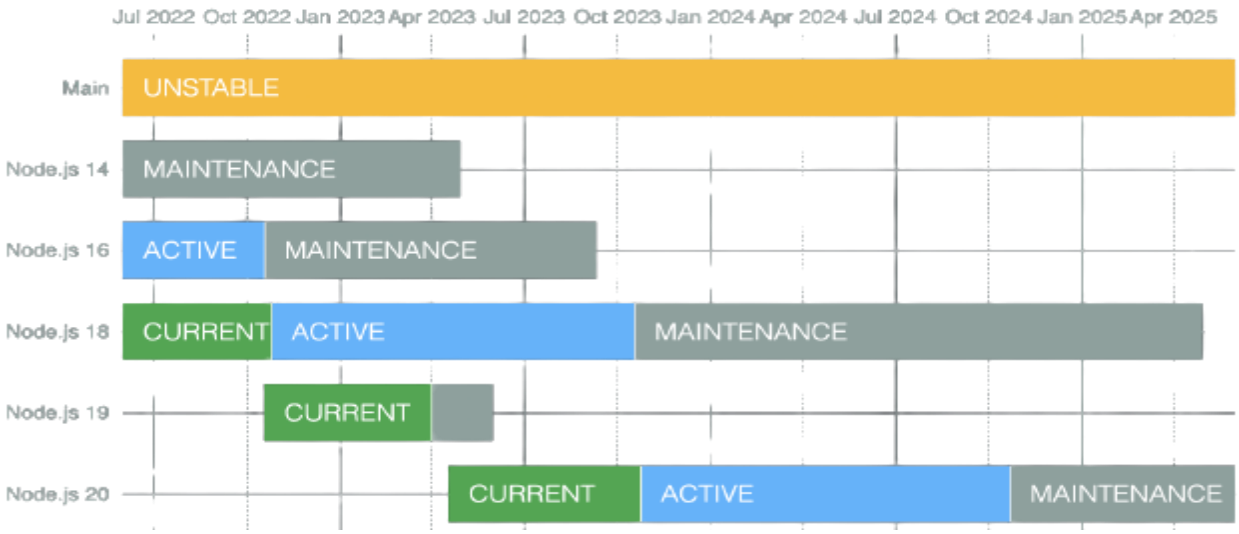

Figura IV. Proyección de las versiones de Node.js con el paso del tiempo [7]

V. CONCLUSIONES

La actualización de Node.js en un entorno empresarial, como se ha explorado en este artículo, es un proceso fundamental y estratégico. Los resultados obtenidos de la transición de la versión 14.20.1 a la 18.18.2 en un ERP internacional basado en React.js, Node.js y MongoDB demuestran que los desafíos técnicos inherentes a esta actualización se pueden superar con éxito, con notables beneficios y mejoras en la eficiencia y la seguridad del sistema.

En primer lugar, la mejora en el rendimiento se destaca como uno de los logros más significativos de esta actualización. La versión más reciente de Node.js ha contribuido a la agilidad y eficiencia del sistema, lo que se traduce en una experiencia de usuario más satisfactoria. La incorporación de nuevas características, como el "Ejecutor de Pruebas Estable," ha optimizado la velocidad y eficiencia en el desarrollo de aplicaciones, proporcionando a los desarrolladores una herramienta valiosa sin la necesidad de depender de dependencias adicionales.

La seguridad, un aspecto crítico en cualquier entorno empresarial, se ha fortalecido con la actualización. La adopción de la versión actualizada de Node.js ha contribuido a la protección contra vulnerabilidades conocidas, reduciendo los riesgos asociados a amenazas de seguridad. Además, la compatibilidad con nuevas bibliotecas y módulos ha simplificado la expansión y enriquecimiento del sistema, brindando a la empresa la capacidad de mantenerse al día con las tendencias tecnológicas emergentes.

Este articulo subraya la importancia de abordar la actualización de tecnologías con una estrategia cuidadosamente planificada y una comprensión completa de los beneficios y desafíos involucrados. Las empresas que enfrentan la necesidad de mantenerse competitivas y adaptarse a un mercado en constante cambio pueden aprender de este enfoque exitoso para incorporar tecnologías emergentes en entornos empresariales dinámicos. En última instancia, la actualización de Node.js se revela como un paso esencial en la búsqueda de la competitividad y la mejora de la experiencia del usuario en el contexto empresarial actual.

VI. REFERENCIAS

[1] Puciarelli, L. (2020). Node JS-Vol. 1: Instalación-Arquitectura-node y npm (Vol. 1). RedUsers.

[2] Puciarelli, L. (2020). Node JS-Vol. 2: Instalación-Arquitectura-node y npm (Vol. 1). RedUsers

[3] Cantelon, M., Harter, M., Holowaychuk, T. J., & Rajlich, N. (2014). Node. js in Action (pp. 17-20). Greenwich: Manning.

[4] Ruiz, F., & Polo, M. (2007). Mantenimiento del Software. Grupo Alarcos, Departamento de Informática de la Universidad de Castilla-La Mancha. URL https://alarcos.esi.uclm.es/per/fruiz/cur/mso/trans/S2.pdf

[5] Kinsta. (2023, June 15). *Qué es Node.js y por qué deberías usarlo*. Kinsta®. *https://kinsta.com/es/base-de-conocimiento/que-es-node-js/*

[6] KnowledgeZone. Node.JS *What? Why? When? https://knowledgezone.co.in/posts/5a966c0c4cbd61256cea187a*

[7] *Villa.M (2022) 11 features in Node.js 18 you need to try. The NodeSource Blog. https://nodesource.com/blog/11-features-nodeJS-18-to-try*