Arquitectura multi-tenant para la implementación de un software como servicio y su influencia en la usabilidad de las MYPES del sector comercio del nororiente Peruano

Multi-tenant architecture for software as a service implementation and its influence on usability of micro and small enterprises in the commercial sector of the Peruvian northeast region

Edwin Alexander Bautista Villegas[1] https://orcid.org/0000-0001-7882-4518
Johann James Valles Paz[1] https://orcid.org/0000-0002-4246-9662
Hitler Collantes Chules[1] https://orcid.org/0000-0002-9815-714X
Nancy Esther Casildo-Bedón[1] https://orcid.org/0000-0001-5255-7757
Jeison Elí Sánchez Calle[1*] https://orcid.org/0000-0001-8039-7682

Recibido 25 de julio de 2023, aceptado 28 de marzo de 2024
Received: July 25, 2023 Accepted: March 28, 2024

RESUMEN

CEATECSOFT E.I.R.L enfrenta un desafío que requiere una evaluación. A través del análisis de actores involucrados en la organización, se ha detectado una disminución en la rentabilidad económica. Esta situación está vinculada a diversos factores, como los costos laborales y las horas extras del personal técnico, la facturación de licencias y los gastos asociados con las actualizaciones del sistema de escritorio. El objetivo de esta investigación fue determinar en qué medida la implementación de la arquitectura multi-tenant en un software como servicio (SaaS) incrementa su usabilidad en las MYPES del sector comercio del Nor Oriente del Peruano. Para ello, se realizó una investigación pre-experimental con una muestra de 32 MYPES a quienes se les aplicó un cuestionario para medir el grado de usabilidad del software puesto en producción. En promedio, el 92 % de los encuestados reconoció mejoras en la solución propuesta, lo que marca una gran diferencia entre el pre y pos-test. En conclusión, el software como servicio (SaaS) desarrollado bajo la arquitectura multi-tenant ha aumentado la usabilidad del sistema implementado, generando beneficios significativos en términos de rentabilidad para la empresa CEATECSOFT E.I.R.L. En primer lugar, se ha logrado una reducción en los costos relacionados con el personal técnico, al eliminar la necesidad de gestionar múltiples versiones y al minimizar los gastos asociados con las visitas técnicas a los clientes, así como los costos relacionados con el registro de nuevos clientes.

Palabras clave: Innovación, sistema de información, interoperabilidad, arquitectura de microservicios, software integrado.

ABSTRACT

CEATECSOFT E.I.R.L. faces a challenge that requires an evaluation. Through the analysis of stakeholders in the organisation, a decrease in economic profitability has been detected. This situation is linked to several factors, such as labour costs and overtime for technical staff, licence invoicing and expenses

[1] Universidad Peruana Unión. Facultad de Ingeniería y Arquitectura, Escuela Profesional de Ingeniería de Sistemas. Tarapoto, Perú. Email: edwinbautista@upeu.edu.pe; johann.valles@upeu.edu.pe; hitler.collantes@upeu.edu.pe; nancy.casildo@upeu.edu.pe; jeisonsanchez@upeu.edu.pe
* Autor de correspondencia: jeisonsanchez@upeu.edu.pe

associated with desktop system upgrades. The objective of this research was to determine to what extent the implementation of multi-tenant architecture in software as a service (SaaS) increases its usability in MSEs in the commerce sector of North Eastern Peru. For this purpose, pre-experimental research was carried out with a sample of 32 MSEs to whom a questionnaire was applied to measure the degree of usability of the software put into production. On average, 92 % of the respondents recognised improvements in the proposed solution, which significantly differs between the pre-and post-test. In conclusion, the software as a service (SaaS) developed under the multi-tenant architecture has increased the implemented system's usability, generating significant profitability benefits for the CEATECSOFT E.I.R.L. company. Firstly, technical staff costs have been reduced by eliminating the need to manage multiple versions and minimising the costs associated with technical visits to customers, and the costs associated with registering new customers.

Keywords: Innovation, information system, interoperability, microservices architecture, embedded software.

INTRODUCCIÓN

El Software como Servicio (SaaS) es un servicio en la nube que se ha convertido en una de las soluciones más productivas y utilizadas actualmente, gracias a su impacto en la economía y su capacidad para resolver problemas de negocios [1].

Según [2] empresas de TI ofrecen cada vez más servicios SaaS como parte de sus ofertas de productos y servicios. [3] afirma que la tendencia hacia el uso de servicios en la nube ha aumentado significativamente en los últimos años, y las empresas de TI han visto una oportunidad para ofrecer soluciones basadas en la nube, como el software como servicio, para satisfacer las necesidades de los clientes [4].

Ofrecer servicios SaaS puede ser beneficioso para las empresas de TI, ya que les permite generar ingresos recurrentes a través de la suscripción de los clientes, en lugar de solo ingresos únicos por la venta de licencias de software [5]. Además, el modelo de SaaS permite una mayor flexibilidad y escalabilidad para las empresas, lo que puede ayudar a satisfacer las demandas cambiantes de los clientes [6].

La arquitectura multi-tenant es importante para desarrollar servicios SaaS por varias razones: eficiencia en la utilización de recursos, múltiples clientes pueden compartir la misma infraestructura y recursos de hardware [7]-[9], permitiendo una mejor utilización de los recursos disponibles y una reducción en los costos de infraestructura [10], [11]. Escalabilidad, los servicios pueden escalar de manera más eficiente para acomodar un mayor número de clientes y un mayor tráfico, logrando que los servicios SaaS crezcan con el tiempo [12]. Mantenimiento simplificado, los desarrolladores pueden realizar actualizaciones y mantenimiento en una única instancia del software, en lugar de tener que hacerlo para cada cliente individualmente [13]. Esto reduce la complejidad y el tiempo necesario para mantener el software y garantiza una mayor consistencia en el rendimiento [14]. Personalización de la experiencia del usuario: a pesar de que los clientes comparten la misma instancia de software, una arquitectura multi-tenant permite la personalización de la experiencia del usuario, facilitando a los clientes personalizar sus propias configuraciones y datos sin afectar a otros clientes [15]. Seguridad, la arquitectura multi-tenant implementa una seguridad más sólida y centralizada, optimizando la gestión de políticas de seguridad y el monitoreo de las actividades de los usuarios [16].

La arquitectura multi-tenant es un enfoque de diseño de software que permite que una sola instancia de una aplicación de software sirva a múltiples clientes o "inquilinos" (tenants) a la vez [1]. En este enfoque, los inquilinos comparten recursos comunes, como la infraestructura, la base de datos y las aplicaciones, pero cada uno tiene su propio espacio de trabajo y sus datos están completamente separados y seguros [1]. La arquitectura multi-tenant ofrece una serie de ventajas, como la reducción de costos, la escalabilidad, la eficiencia, la personalización y la seguridad, siendo una opción popular para los proveedores de SaaS y otras aplicaciones empresariales [10].

La compañía CEATECSOFT E.I.R.L, que se dedica al desarrollo y venta de software personalizado en la región de San Martín - Perú, tiene como principal

objetivo expandirse y mejorar su estabilidad financiera. Para lograrlo, han decidido adoptar la arquitectura "multi-tenant" para crear aplicaciones basadas en la nube. El objetivo es reducir los costos de mantenimiento de su aplicación SaaS (Software como Servicio) y acelerar la adaptación a nuevos clientes (suscriptores). En la actualidad, CEATECSOFT E.I.R.L está experimentando un problema que requiere una evaluación cuidadosa. A través del análisis de implicados en la organización, se ha identificado una disminución en la rentabilidad económica, la cual está relacionada con varios factores, tales como el costo de la mano de obra y las horas extras del personal técnico, la falta de eficacia en la recaudación y facturación de licencias, la falta de un sistema multi-tenant y los gastos en actualizaciones del sistema de escritorio. La consecuencia de esto es que se produce una disminución en la calidad del servicio, se reactivan servicios que se encontraban fuera del contrato, los clientes reactivados deben hacer pagos adicionales por los servicios, se pierde la capacidad de hacer un seguimiento adecuado al soporte al cliente y se pueden recibir sanciones económicas por parte de la SUNAT (Superintendencia Nacional de Aduanas y de Administración Tributaria - Perú).

El propósito de este estudio es resolver un problema específico, y para lograrlo, se busca determinar en qué medida la implementación de la arquitectura multi-tenant en un software como servicio (SaaS) incrementa su usabilidad en las MYPES (Micro y Pequeña Empresa) del sector comercio del Nor Oriente del Peruano.

MÉTODOS O METODOLOGÍA COMPUTACIONAL

Análisis estadístico

Este estudio se realizó a través de un enfoque cuantitativo y experimental, con un diseño pre-experimental. Se usó la encuesta como técnica para recopilar datos, utilizando el cuestionario "System Usability Scale" (SUS) desarrollado por John Brooke en 1986; que incluye una escala de valoración tipo Likert de 1 a 5. La población y muestra estuvo conformada por de 32 clientes. Se realizó un análisis inferencial con un nivel de confianza del 95% y un margen de error del 5%. Dada la cantidad de la muestra, se aplicó la prueba de Shapiro-Wilk. Se aplicó la estadística no paramétrica para grupos relacionados, dado que la muestra fue la misma para el pre y pos-test.

Para calcular e interpretar los resultados del cuestionario, se siguió el siguiente procedimiento: se sumaron las respuestas de los enunciados impares y se les restó 5; luego se sumó las respuestas de los enunciados pares y se restó 25; finalmente, se sumó ambos resultados y se multiplicó por 2,5. El puntaje obtenido fue interpretado de acuerdo con la escala siguiente (Figura 1).

Fases de desarrollo

Se utilizó un método evolutivo incremental, similar a OpenUp [17], el cual se describe en la Tabla 1. Este método está planteado por cuatro 4 fases: inicio, elaboración, construcción y transición; permitiendo asumir que, en la fase de inicio, la especificación de requerimientos funcionales y no funcionales puede cambiar en cualquier momento del ciclo de vida del proyecto [18].

Con el fin de lograr nuestro primer objetivo, hemos llevado a cabo la fase de inicio, en la cual nos enfocamos en identificar los objetivos del proyecto, evaluar su viabilidad y definir sus requisitos. Durante esta fase, realizamos un análisis preliminar del proyecto con el propósito de establecer su visión, alcance y objetivos. Para el cumplimiento del segundo objetivo, hemos realizado la definición detallada de los requisitos, la

Fuente: Elaboración propia

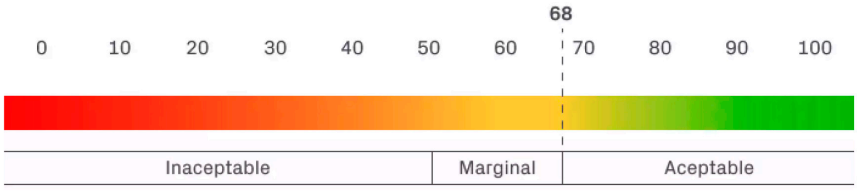

Figura 1. Representación de los resultados de un SUS.

Tabla 1. Fases del desarrollo de la solución.

Fase	Actividad
Inicio	– Establecer el alcance del proyecto. – Especificación de los requerimientos funcionales y no funcionales. – Planificación de la solución a nivel: presupuesto, cronograma de actividades, riesgos y calidad del proyecto.
Elaboración	– Diseñar, implementar y validar la arquitectónica multi-tenant. – Definir las herramientas tecnológicas a utilizar. – Desarrollar prototipos y modelos. – Establecer un ambiente de desarrollo.
Construcción	– Desarrollar el código. – Realizar pruebas. – Verificar la calidad del software. – Gestionar los cambios. – Preparar la transición del proyecto.
Transición	– Puesta en producción del software. – Capacitar al cliente. – Establecer el soporte y mantenimiento.

Fuente: Elaboración propia.

arquitectura del sistema y la planificación del proyecto. Durante esta fase, se definieron las características y funcionalidades del software.

En la fase de construcción nos centramos en desarrollar el software utilizando los requisitos y diseños definidos en la fase de elaboración. En la fase transición, el software se entrega al cliente para su validación y se prepara para su implementación en el entorno de producción.

RESULTADOS Y DISCUSIÓN

Caracterización del sistema actual

CEATECSOFT E.I.R.L cuenta con un sistema de escritorio que ha sido implementado en 32 MYPES del sector comercio del Nor Oriente Peruano. Este sistema, desarrollado en el lenguaje de programación Java, fue realizado bajo una arquitectura basada en componentes, lo que no solo facilitó su modularidad, sino también su escalabilidad. Para la gestión de la base de datos, se optó por utilizar SQL Server. Además de estas características técnicas, el sistema de escritorio se destacaba por su capacidad para operar sin conexión a Internet, una característica especialmente valiosa en entornos con conectividad intermitente o limitada.

El sistema se estructuraba en torno a cuatro módulos principales, cada uno diseñado para atender aspectos específicos de la gestión empresarial. El módulo restaurante proporcionaba herramientas para la gestión eficiente de establecimientos gastronómicos, mientras que el módulo comercial se centraba en optimizar las operaciones de negocio. El módulo de almacén, abarcando compras, inventario y kárdex, ofrecía una solución integral para la gestión logística. Por último, el módulo de finanzas, que comprendía contabilidad y costos, brindaba herramientas precisas para el análisis financiero y la toma de decisiones estratégicas.

Sin embargo, tras el análisis de implicados, se han identificado diversas deficiencias y limitaciones en este programa. Entre las más destacadas se encuentran la limitada accesibilidad, costos asociados a las actualizaciones y mantenimiento, problemas de seguridad y gastos en hardware.

Según [19]-[21], los sistemas de escritorio están quedando obsoletos, debido a una característica en particular: los usuarios tienen que descargar y actualizar manualmente el software en sus computadoras cada vez que hay una nueva versión. Esta limitación se debe a la arquitectura propia de los sistemas de escritorio, tal como se muestra en la Figura 2.

Con el propósito de evaluar lo mencionado, se aplicó el cuestionario SUS. Este instrumento tiene

Fuente. Elaboración propia.

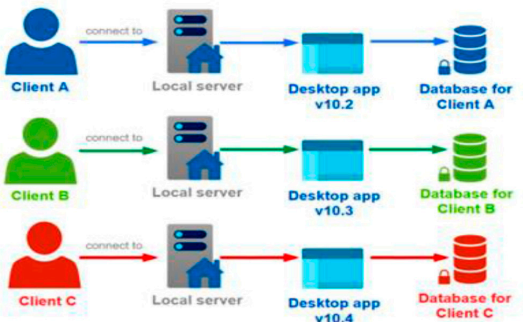

Figura 2. Arquitectura del sistema de escritorio.

como objetivo recopilar la opinión de los usuarios sobre la usabilidad del sistema de escritorio en sus respectivas MYPES. A continuación, se presenta el resultado respecto a la caracterización de la situación actual.

La Tabla 2 muestra que el software de escritorio no cumple con las expectativas de los usuarios en varios aspectos, destacando la falta de usabilidad. Un puntaje de 55 indica que el software no es ampliamente utilizado, lo que podría llevar a CEATECSOFT E.I.R.L a perder toda su base de clientes en el futuro.

El análisis de los datos presentados en la Tabla 2 pone de manifiesto que el software de escritorio no alcanza las expectativas de los usuarios en múltiples aspectos, con un énfasis particular en la falta de usabilidad. Como señala [20], la usabilidad es un factor crítico para la adopción y retención de los usuarios en el contexto de los sistemas de software. El puntaje registrado de 55 en este aspecto indica una insatisfacción significativa entre los usuarios, lo cual plantea una seria preocupación para CEATECSOFT E.I.R.L. De acuerdo con la investigación de [23], una baja satisfacción con la usabilidad puede resultar en una disminución del uso continuo del software, lo que a su vez podría conducir a la pérdida de clientes a largo plazo. En este sentido, es imperativo que la empresa aborde estas deficiencias de usabilidad de manera proactiva para garantizar su viabilidad futura en el mercado.

La Tabla 2 también resalta otras áreas de preocupación en relación con el software de escritorio. Por ejemplo, los puntajes relativamente bajos en términos de rendimiento y confiabilidad podrían indicar problemas técnicos subyacentes que afectan la experiencia del usuario. Como argumenta [19], el rendimiento y la confiabilidad son elementos fundamentales para la satisfacción del usuario y la percepción de calidad del software.

En base a lo anterior, nuestro propósito al implementar la arquitectura multi-tenant es potenciar la rentabilidad de CEATECSOFT E.I.R.L, enfocándonos en abordar los desafíos actuales que enfrenta. Estos incluyen los elevados gastos en personal especializado para llevar a cabo actualizaciones, la necesidad de gestionar las distintas versiones y mantener el código fuente separado para cada cliente, los costos relacionados con las visitas técnicas a cada cliente, y las deficiencias y los gastos al registrar a un nuevo cliente.

Arquitectura Multi-tenant implementada en un software como servicio (SaaS)

Para cumplir con nuestro segundo objetivo de estudio y ofrecer una solución a la problemática existente, hemos desarrollado un sistema de información utilizando la metodología multi-tenant. En la Figura 4 presentamos el flujograma que explica la integración de esta arquitectura en nuestro proyecto.

Cuando se utiliza una arquitectura multi-tenant, donde diferentes clientes comparten una misma instancia de una aplicación [11], es fundamental contar con un método eficiente y efectivo para identificar e inicializar los tenants. Esto implica configurar correctamente los recursos, como la conexión a la base de datos, para cada tenant. Aunque Laravel, un popular framework de desarrollo de aplicaciones web en PHP, no ofrece de forma predeterminada

Tabla 2. Resultado de la aplicación del SUS.

MYPES encuestadas	Puntaje obtenido	Estado	Puntaje ideal
32	55	Marginal	100

Fuente. Elaboración propia.

una solución específica para gestionar aplicaciones multi-tenant, se ha desarrollado una solución para abordar la identificación e inicialización de tenants en aplicaciones basadas en Laravel [22]. Para satisfacer esta necesidad, hemos elegido utilizar la librería "Tenancy for Laravel". Esta librería proporciona una solución integral y autónoma para gestionar tanto la identificación como la inicialización de los tenants, y también se enfoca en optimizar el rendimiento y fortalecer la seguridad en entornos multi-tenant [24].

En la Figura 3, se puede observar cómo el flujo de trabajo comienza con la intervención del middleware "InitializeTenancyByDomainOrSubdomain". Este componente de software es responsable de dirigir inicialmente el proceso de identificación del inquilino. Dependiendo del dominio de la solicitud, este middleware bifurca el flujo hacia "InitializeTenancyBy Domain" o "InitializeTenancyBySubdomain". Esta bifurcación tiene en cuenta la estructura de la solicitud y separa las solicitudes que provienen de dominios y subdominios.

A continuación, se invoca el método "initialize Tenancy", el cual se encuentra en la clase abstracta "IdentificationMiddleware" de la cual extienden "InitializeTenancyByDomain" y "InitializeTenancyBySubdomain". El método "initializeTenancy" recibe dos parámetros importantes: primero, la solicitud (request) que hace referencia a una instancia de la clase Request de Laravel, y segundo, un array de argumentos que contienen datos del subdominio. Dentro de este método, también se instancia la clase "Tenancy" a través del método "initialize", utilizando los argumentos del subdominio o dominio como parámetro.

Una vez dentro del alcance de la clase "Tenancy", se busca activamente el inquilino correspondiente dentro del modelo "Tenant". Esta acción representa la fase real de identificación, donde se busca el inquilino específico que realiza la solicitud. La identificación exitosa del inquilino desencadena dos eventos sucesivos: "InitializingTenancy" y "BootstrappingTenancy". El primero simboliza el inicio del proceso de establecimiento de la tenencia, mientras que el segundo representa la conclusión de la inicialización.

"BootstrappingTenancy" es particularmente relevante porque incorpora la clase "DatabaseManager".

Fuente. Elaboración propia.

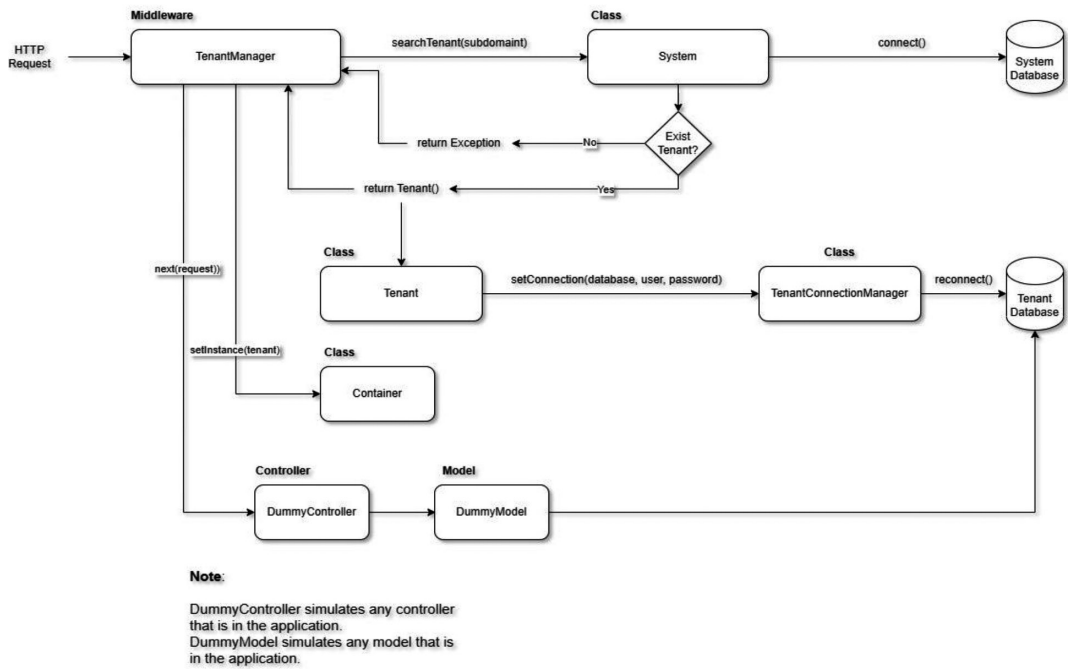

Figura 3. Flujograma de la arquitectura multi-tenant implementada en la solución.

Este componente es responsable de establecer la conexión con la base de datos correspondiente al inquilino identificado. Este proceso de conmutación de la base de datos es lo que finalmente permite a la aplicación interactuar con los datos específicos del inquilino.

En la Figura 4 se muestra el esquema del sistema de información desarrollado. El sistema consta de "N" clientes que realizan diversas peticiones al servidor, que se comunica a través de la vista (SPA - VUE) y consume la API (Api - Laravel). En esta etapa se realizan las validaciones correspondientes y se redirecciona a cada cliente a su respectiva base de datos.

El software desarrollado se compone de cinco módulos principales, cada uno de los cuales cuenta con sus respectivos submódulos y microservicios. Estos paquetes se ofrecen a las MYPES bajo un modelo de licencia mensual (Figura 5).

Influencia del sistema desarrollado en la usabilidad de las MYPES

En el marco de nuestro tercer objetivo, buscaremos analizar el impacto del sistema de información basado en la arquitectura multi-tenant en términos de usabilidad para las MYPES del sector comercial del Nor Oriente de Perú.

La Tabla 3 refleja los resultados obtenidos después de la implementación del sistema de información basado en la arquitectura multi-tenant. Con una puntuación de 92, casi perfecta, se demuestra que el nuevo sistema en producción cumple con las expectativas de los clientes, presentando altos niveles de aceptabilidad y usabilidad.

Inicialmente, se llevó a cabo un análisis para determinar qué tipo de estadística se emplearía en este estudio, es decir, si se utilizaría estadística paramétrica o no paramétrica. Para ello, se

Fuente. Elaboración propia.

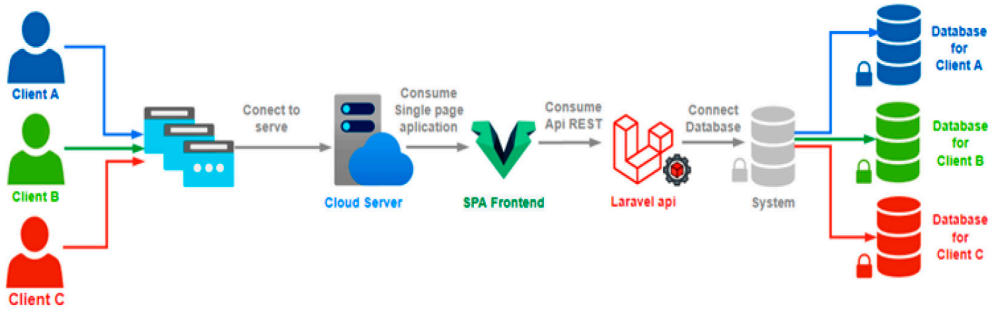

Figura 4. Esquema de Sistema de información puesto en producción.

Fuente. Elaboración propia.

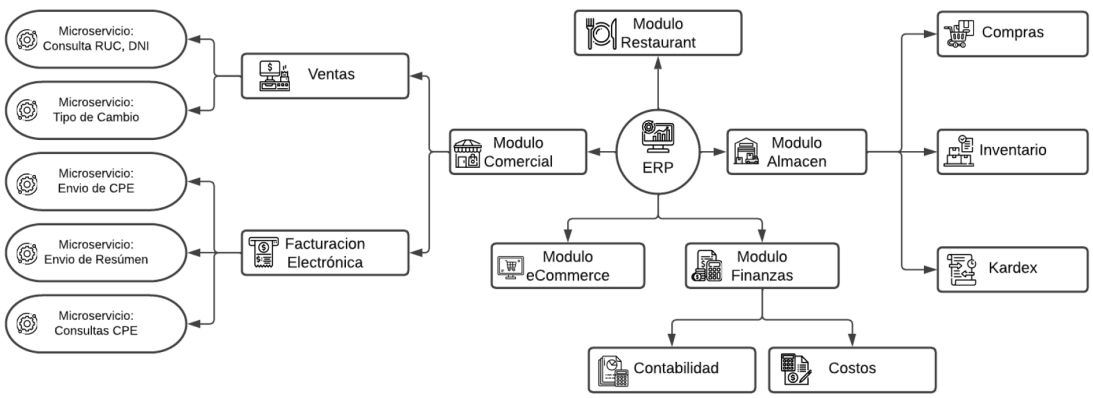

Figura 5. Sistema de información (ERP) desarrollado.

Tabla 3. Resultado de la aplicación del SUS.

MYPES encuestadas	Puntaje obtenido	Estado	Puntaje ideal
32	92	Aceptable	100

Fuente. Elaboración propia.

llevaron a cabo análisis de los tres supuestos de la estadística paramétrica: variable cuantitativa numérica, distribución normal y homogeneidad de varianzas [5]. Nuestra investigación cumple dos de estos supuestos: las variables son cuantitativas y presentan una distribución normal, como se puede observar en la Tabla 4, donde los valores de "Sig." son iguales o superiores a 0,05. Sin embargo, no se cumple con el supuesto de homogeneidad de varianzas, como se muestra en la Tabla 5, ya que los valores de "Sig." son inferiores a 0,05.

Dado que los supuestos de una técnica paramétrica no se cumplen, se aplicó una técnica no paramétrica. Se utilizó la prueba de Wilcoxon para muestras relacionadas dado a que se trabajó con la misma muestra para el pre y post test; cuyos resultados se presentan a continuación.

Tabla 4. Prueba de normalidad. *Con* P valor ≤ 0,05 → Se rechaza *H*o y acepta la H1.

	Shapiro-Wilk		
	Estadístico	**Gl.**	**Sig.**
Pre-test	0,957	32	0,229
Post-test	0,938	32	0,064

Fuente. Elaboración propia.

Los datos de la Tabla 6 respaldan el logro de nuestro objetivo principal, la implementación de una arquitectura multi-tenant implementado en un software como servicio ha demostrado influir significativamente en la usabilidad de las MYPES del sector comercial en la región nororiental del Perú. Este hallazgo se sustenta dado a que el valor de p es inferior a 0,05, lo que indica una relación estadísticamente significativa entre la variable independiente (la arquitectura multi-tenant) y la variable dependiente (usabilidad de las MYPES).

Hemos realizado un análisis de los costos de implementación antes y después del estudio, teniendo en cuenta tanto los costos únicos como los costos mensuales. Según nuestros cálculos, el sistema de escritorio tiene un costo total de S/.26,253.00, mientras que el sistema de información bajo la arquitectura multi-tenant tiene un costo total de S/.15,490.00. Esto demuestra que la implementación del sistema multi-tenant genera un ahorro del 41,03% para la empresa CEATECSOFT E.I.R.L

Tabla 5. Prueba de homogeneidad de varianza. Con *P valor ≤ 0,05 → Se rechaza Ho* y acepta la H1.

		Estadístico de Levene	**gl1**	**gl2**	**Sig.**
Pre y post Test	Se basa en la media	5,829	1	62	0,019
	Se basa en la mediana	4,988	1	62	0,029
	Se basa en la mediana y con gl ajustado	4,988	1	54,302	0,030
	Se basa en la media recortada	5,831	1	62	0,019

Fuente. Elaboración propia.

Tabla 6. Prueba de Wilcoxon.

Hipótesis nula	Prueba	Sig.	Decisión
La mediana de las diferencias entre la usabilidad del sistema de información antes del tratamiento y posterior al tratamiento es igual a 0.	Prueba de rangos con signo de Wilcoxon para muestras relacionadas	0,000	Rechaza la hipótesis nula

Fuente. Elaboración propia.

Tabla 7. Gastos de implementación de sistema de escritorio y sistema de información bajo la arquitectura multi-tenant.

Concepto	Sistema de escritorio			Sistema de información bajo la arquitectura multi-tenant		
	Monto	Cantidad	Sub total	Monto	Cantidad	Sub total
Recursos Humanos						
Desarrollador de Software	S/. 2200.00	3	S/. 6600.00	S/. 2500.00	2	S/. 5000.00
Soporte Técnico	S/. 1600.00	2	S/. 3200.00	S/. 1600.00	1	S/. 1600.00
Equipamiento						
Servidor local	S/. 5000.00	1	S/. 5000.00	S/. 0.00	0	S/. 0.00
VPS (Servidor Privado Virtual)	S/. 0.00	0	S/. 0.00	S/. 200.00	1	S/. 200.00
Estaciones de trabajo	S/. 2500.00	3	S/. 7500.00	S/. 2500.00	3	S/. 7500.00
POS	S/. 120.00	3	S/. 360.00	S/. 120.00	3	S/. 360.00
Servicios						
Mantenimiento de equipo	S/. 150.00	5	S/. 750.00	S/. 150.00	5	S/. 750.00
Electricidad	S/. 80.00	1	S/. 80.00	S/. 0.00	1	S/. 0.00
Internet	S/. 80.00	1	S/. 80.00	S/. 80.00	1	S/.80.00
Licencias						
SQL Server	S/. 1833.00	1	S/. 1833.00	–	–	S/. 0.00
Windows Server	S/. 850.00	1	S/. 850.00	–	–	S/. 0.00
MySQL	S/. 0.00	–	S/. 0.00	–	–	S/. 0.00
Total			S/.26,253.00			S/.15,490.00

Fuente. Elaboración propia.

en comparación con el sistema de escritorio. Se destaca la flexibilidad y escalabilidad inherentes a la arquitectura multi-tenant, lo que la convierte en una opción más adecuada para adaptarse a las fluctuaciones del entorno empresarial. Esta elección no solo implica un beneficio financiero, sino que también posiciona a la empresa en una mejor posición para innovar y mantener su competitividad a largo plazo.

CONCLUSIONES

Basándonos en los resultados de la prueba estadística Wilcoxon para muestras relacionadas, se concluye que el estudio ha logrado mejorar la usabilidad del software como servicio (SaaS) desarrollado bajo la arquitectura multi-tenant. Esta mejora ha resultado en un ahorro considerable para la empresa CEATECSOFT E.I.R.L en varios aspectos. Primero, se ha reducido los costos asociados con el personal técnico, al disminuir la necesidad de desarrolladores. Además, hemos eliminado la gestión de distintas versiones y el mantenimiento de código fuente separada para cada cliente, lo que ha generado ahorros adicionales. También se han reducido los gastos relacionados con las visitas técnicas a cada cliente y los costos asociados al registro de nuevos clientes.

Hemos identificado los indicadores que tienen un impacto directo en la usabilidad de un sistema de información. Basándonos en estos hallazgos, nos hemos asegurado de satisfacer las necesidades de los clientes al cumplir rigurosamente con los requerimientos funcionales y no funcionales.

En cuanto a la implementación del software, esta se llevó a cabo de manera eficiente, ya que los clientes lo adoptaron sin dificultades. Como resultado, podemos afirmar que el software como servicio

(SaaS) desarrollado bajo la arquitectura multi-tenant ha tenido un impacto positivo en la usabilidad de las micro y pequeñas empresas (MYPES) del sector comercial en el Nor Oriente del Perú.

La solución propuesta ha sido diseñada de manera que pueda ser implementada en cualquier empresa proveedora de software. Basándonos en los resultados favorables encontrados, recomendamos su aplicación, teniendo en cuenta las características y realidades específicas de las organizaciones donde se pretende implantar.

REFERENCIAS

[1] R. Santana, A. Malucelli, and S. Reinehr, "Práticas de customização de software aplicadas em SaaS multi-tenant," in *Simpósio brasileiro de qualidade de software (SBQS)*, 2020, pp. 55-60. [Online]. Available: https://sol.sbc.org.br/index.php/sbqsestendido/article/view/14193/14040

[2] J. Herrera-Cubides, N. Gelvez-García y D. López-Sarmiento, "LMS SaaS: Una alternativa para la formación virtual SaaS", *Revista Chilena de Ingeniería*, vol. 27, no. 1, pp. 164-179, 2019, doi: 10.4067/S0718-33052019000100164.

[3] T. Satpathy *et al.*, *Una Guía para el Conocimiento de SCRUM*, 3rd ed. Arizona, USA: SCRUMstudy, 2017.

[4] F. Matloob *et al.*, "Software defect prediction using ensemble learning: A systematic literatura review," *IEEE Access*, vol. 9, no. 1, pp. 98754-98771, 2021, doi: 10.1109/ACCESS.2021.3095559.

[5] O. Agudelo-Varela, J. Martínez-Baquero y S. Valbuena-Rodríguez, "Administración de TI en la facultad de ingeniería de la Universidad de los Llanos", *Revista Politécnica*, vol. 16, no. 31, pp. 68-76, 2020, doi: 10.33571/rpolitec.v16n31a5.

[6] S.F. Ruiz-Paz, R. Santaolaya-Salgado, O. G. Fragoso-Díaz, F.J. Álvarez-Rodríguez y J.C. RojasPérez, "Modelo de orquestación dinámica para flujos de trabajo del software como servicio", *Ingeniería, Investigación y Tecnología*, vol. 20, no. 3, pp. 1-12, 2019, doi: 10.22201/fi.25940732e.2019.20n3.034.

[7] D. Strode, T. Dingsøyr, and Y. Lindsjorn, "A teamwork effectiveness model for agile software development," *Empirical Software Engineering*, vol. 27, no. 2, pp. 1-50, 2022, doi: 10.1007/s10664-021-10115-0.

[8] E.M. Arvanitou, A. Ampatzoglou, A. Chatzigeorgiou, and J. Carver, "Software engineering practices for scientific software development: A systematic mapping study," *Journal of Systems and Software*, vol. 172, no.1, pp. 1-18, 2021, doi: 10.1016/j.jss.2020.110848.

[9] A.A. Khan *et al.*, "Software architecture for quantum computing systems -A systematic review," *Journal of Systems and Software*, vol. 201, no. 1, p. 111682, 2023, doi: 10.1016/j.jss.2023.111682.

[10] B. Fang, X. Zeng, and M. Zhang, "NestDNN: Resource-aware multi-tenanton-device deep learning for continuous mobile vision," *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, MobiCom'18*, 2018, pp. 115-127, doi: 10.1145/3241539.3241559.

[11] M.H. Hilman, M.A. Rodriguez, and R. Buyya, "Multiple workflows scheduling in multitenantdistributed systems: A taxonomy and future directions," *ACM Computing Surveys*, vol. 53, no. 1, 2020, doi: 10.1145/3368036.

[12] Y. Pérez *et al.*, "Design thinking en la planificación de pruebas de software", *Innovación y Software*, vol. 1, no. 2, pp. 40-51, 2020. [En línea]. Disponible: https://www.redalyc.org/journal/6738/673870835004/673870835004.pdf

[13] A.R. Ortega, M. Noguera, J.L. Garrido, K. Benghazi, and L. Chung, "Component-based design formulti-tenantmulti-target support in the cloud," in *Enterprise and Organizational Modeling and Simulation. Lecture Notes in Business Information Processing*, vol. 153, J. Barjis, A. Gupta, A. Meshkat Eds. 2013, pp. 146-160, doi: 10.1007/978-3-642-41638-5_10.

[14] A. Sunardi and Suharjito, "MVC architecture: A comparative study between Laravel framework and slim framework in freelancer project monitoring system web based," *Procedia Computer Science*, vol. 157, pp. 134-141, 2019, doi: 10.1016/j.procs.2019.08.150.

[15] V. Castro-Rivera, R. Herrera-Acuña y M. Villalobos-Abarca, "Desarrollo de un

software web para la generación de planes de gestión de riesgos de software", *Información Tecnológica*, vol. 31, no. 3, pp. 135-148, 2020, doi: 10.4067/S0718-07642020000300135.

[16] L. Mendoza-Munar, "El software como servicio y el habeas data: una aproximación desde el derecho privado y constitucional en Colombia", *Dixi*, vol. 20, no. 27, p. 13, 2018, doi: 10.16925/di.v20i27.2396.

[17] P.S.F. Yip *et al.*, "The opportunities and challenges of the first three years of open up, an online text-based counselling service for youth and young adults," *International Journal of Environmental Research and Public Health*, vol. 18, no. 24, pp. 1-10, 2021, doi: 10.3390/ijerph182413194.

[18] J. Medina, E. Pineda y F. Téllez, "Requerimientos de software: prototipado, software heredado y análisis de documentos", *Ingeniería y Desarrollo*, vol. 37, no. 2, pp. 327-345, 2022, doi: 10.14482/inde.37.2.1053.

[19] A. Uribe Arévalo y E. Norman Acevedo, "Internacionalización de la pequeña y mediana industria del software y de las tecnologías informáticas (SW & TI) a través del efecto trampolín del gremio", *Cuadernos Latinoamericanos de Administración*, vol. 16, no. 31, pp. 1-14, 2020, doi: 10.18270/cuaderlam.v16i31.3068.

[20] G.A. Vizalote-Rodriguez, "Contribución de las TI en la mejora de la productividad de las PYME", *Revista Amazonía Digital*, vol. 1, no. 1, p. e164, 2022, doi: 10.55873/rad.v1i1.164.

[21] R. Prada, M. Rueda y P. Ocampo, "Clima de trabajo y su relación con la productividad laboral en empresas de tecnología", *Revista Espacios*, vol. 41, no. 45, pp. 57-75, 2020, doi: 10.48082/espaciosa20v41n45p06.

[22] A. Quincho Poma, "Arquitectura multi-tenanten aplicaciones basadas en la nube para mejorar la rentabilidad en la empresa Group Solution Tecnology sociedad anónima cerrada", Tesis de Grado, Facultad de Ingeniería de Sistemas, Universidad Nacional del Centro del Perú, Huancayo, Perú, 2020. [En línea]. Disponible: https://repositorio.uncp.edu.pe/handle/20.500.12894/6754

[23] L. Oll Majin, J. Roa Valencia y R. Robles Molina, "Reestructuración de la Arquitectura de Software para la Implementación de Software como Servicio (SaaS) en la empresa Gearsis SAS", Tesis de Grado, Facultad Ingenierías, Universidad Cooperativa de Colombia, 2019. [En línea]. Disponible: https://repository.ucc.edu.co/server/api/core/bitstreams/d298f21b-fb10-42d7-978f-4ff8a45b7873/content

[24] M. Laaziri, K. Benmoussa, S. Khoulji, and M. Kerkeb, "A Comparative study of PHP frameworks performance," *Procedia Manufacturing*, vol. 32, pp. 864-871, 2019, doi: 10.1016/j.promfg.2019.02.295.