Revista de investigación de sistemas e informática
17(1): 93-102 (2024)
DOI: https://doi.org/10.15381/risi.v17i1.28097

ISSN: 1815-0268 (Impreso) / ISSN: 1816-3823 (Electrónico)
Facultad de Ingeniería de Sistemas e Informática - UNMSM

Diseño y desarrollo de un framework para la externalización de la configuración en entornos multitenant

Design and development of a framework for configuration externation on multitenant environments

José Miguel Mori Peña
https://orcid.org/0000-0002-0667-3869
jose.mori5@unmsm.edu.pe

Fany Yexenia Sobero Rodriguez
https://orcid.org/0000-0002-0323-6110
fsoberor@unmsm.edu.pe

Universidad Nacional Mayor de San Marcos. Lima, Perú

Recibido: 15/05/2024 - Aceptado: 15/06/2024 - Publicado: 31/07/2024

RESUMEN

La gestión de la configuración de las aplicaciones, es un aspecto de alto impacto en los procesos de desarrollo, ejecución y despliegue de las mismas, debido a que típicamente estas configuraciones, hacen referencia a servicios externos/terceros como base de datos, credenciales, correo electrónico, mensajería, personalización de la misma aplicación, entre otros, y suelen variar de acuerdo al entorno en el cual las aplicaciones son desplegadas y al tenant que en tiempo de ejecución hace uso de la aplicación. Bajo estas premisas, el trabajo de investigación propone el desarrollo de un framework, que permita externalizar la configuración con soporte multitenant, estableciendo un lugar central donde se puedan gestionar estas configuraciones, brindando a la aplicación, la capacidad de ejecutarse en múltiples entornos y dar soporte a múltiples tenants sin realizar modificación/recompilación del mismo. Con el objetivo de comprobar el aporte del framework propuesto, se trabajó con profesionales activamente involucrados en el desarrollo de software, a través de un taller dirigido y una encuesta de satisfacción Post Uso, adicionalmente se utilizó el método SQALE para evaluar la calidad del código fuente de la herramienta propuesta. Como resultado la encuesta reveló una alta satisfacción de los involucrados para con la herramienta, y una calificación alta de calidad de desarrollo.

Palabras clave: Framework; Externalización de la Configuración; Multitenant; Despliegue.

ABSTRACT

Application configuration management is a high-impact aspect in the development, execution, and deployment processes, since typically these configurations refer to external/third-party services such as databases, credentials, email, messaging, customization of the application itself, and more, and usually vary according to the environment in which the applications are deployed and the tenant that uses the application at runtime.Under these premises, the research work proposes the development of a framework, which allows externalize the configuration with multitenant support, establishing a central place where these configurations can be managed, enabling the application the ability to run in multiple environments and support multiple tenants without making modifications/recompile. In order to verify the contribution of the proposed framework, we worked with professionals actively involved in software development, through a directed workshop and a Post Use satisfaction survey. Additionally, the SQALE method was used to evaluate the quality of the proposed tool source code. As a result, the survey revealed high satisfaction of those involved with the tool, and a high development quality rating.

Keywords: Framework; Externalize the Configuration; Multitenant; Deployment.

© Los autores. Este artículo es publicado por la Revista de investigación de sistemas e informática de la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos. Este es un artículo de acceso abierto, distribuido bajo los términos de la licencia Creative Commons Atribución 4.0 Internacional (CC BY 4.0) [https://creativecommons.org/licenses/by/4.0/deed.es] que permite el uso, distribución y reproducción en cualquier medio, siempre que la obra original sea debidamente citada de su fuente original. Para mas información, por favor póngase en contacto con revistarisi.fisi@unmsm.edu.pe

I. INTRODUCCIÓN

Mover la información de la configuración fuera del paquete de despliegue, de una aplicación a una ubicación centralizada, para facilitar la administración y el control de los datos de configuración entre las aplicaciones y sus instancias, es considerada una buena práctica DevOps (Balalaie et al., 2016), y es el objetivo principal del patrón de diseño llamado Externalización de la Configuración (ckittel, 2023). Sin embargo, no siempre se cumple dicho objetivo durante el desarrollo de software, debido a los desafíos que un entorno multitenant genera, como el hecho de que las aplicaciones tienen que ser altamente configurables con configuraciones especificas por tenant (Hall, 2010) , y que la creación de un framework multitenant estable, robusto y extensible desde cero sea muy desafiante, dado que los desarrolladores tendrían que entender cómo se manejan las peticiones web para cada usuario de un tenant especifico (Bien & Dan Thu, 2015).

Este hecho genera que los equipos de desarrollo usen diversas herramientas para reducir el esfuerzo de desarrollo de sus aplicaciones. Sin embargo, estas herramientas ocultan la complejidad a los desarrolladores, permitiendo facilitar el desarrollo de aplicaciones simples, y dificultando el desarrollo de aplicaciones complejas, que requiere comprensión de la herramienta a bajo nivel (Bien & Dan Thu, 2015).

El uso de múltiples herramientas prima en los proyectos, debido a que estas no brindan todas las características necesarias para una aplicación multitenant o en su defecto sobrepasa la oferta de características que no están relacionadas (Kale & Borhade, 2013), dejando de lado el principio de Single Responsability.

Actualmente los equipos de desarrollo externalizan la configuración de sus aplicaciones, con el objetivo de hacerlas portables, para que puedan ser ejecutadas en diversos ambientes (desarrollo, pruebas, producción) y que se comporten de la misma forma sin importar el ambiente donde se encuentren, para lograrlo hacen uso de herramientas y/o servicios que facilitan esta tarea (ckittel, 2023), pero la mayoría de veces estas herramientas no consideran que estas configuraciones pueden también estar segregadas por tenants.

Por este motivo el presente trabajo de investigación propone el desarrollo de un framework, para la externalización de la configuración de aplicaciones basadas en arquitecturas multitenant, con el objetivo de reducir tiempos de desarrollo de software, abstrayendo la complejidad de la gestión de los tenants y facilitando la integración del framework con aplicaciones nuevas y/o existentes.

II. ANTECEDENTES

A lo largo de los años se han defnido diversas técnicas para externalizar la configuración, como las descritas por Hofmann et al. (2016) donde enfatiza las características que deben tener las aplicaciones para facilitar la transición entre entornos sin realizar cambios en los artefactos de despliegue, para esto describe 3 técnicas diferentes para externalizar la configuración de las aplicaciones a través de los entornos, las cuales son:

- Variables de Entorno
- Archivos de Configuración
- Sistemas de Configuración.

Las tres técnicas listadas son actualmente utilizadas, cada una con sus propias ventajas y desventajas.

Actualmente existen diversas herramientas que implementan las técnicas mencionadas para externalizar la configuración, bajo esta premisa el autor Sayagh et al, (2017) realizó un estudio donde se analizaron más de 2000 proyectos java alojados en GitHub, con el objetivo de analizar la popularidad de los frameworks de configuración y las características que poseen.

El estudio se enfoca en el propósito general de la configuración de los 11 frameworks de programación listados en la Tabla 1.

Con el objetivo de entender la diferencia entre los frameworks de configuración anteriormente listados, los autores construyeron la taxonomía de los mismos, basados en sus características y propiedades, para ello estudiaron la documentación publica, blogs, fórums asociados a cada framework de forma manual, obteniéndose como resultado 17 propiedades agrupadas en 3 dimensiones (General Properties, Feature Richness y Programming Support), dando como resultadoque los frameworks Common y Spring satisfacen la mayoría de las características de un framework de configuración.

Si bien las técnicas para la externalización brindan pautas para externalizar la configuración, estas no encajan perfectamente con ciertas arquitecturas de software como la multitenant, ya que en esta arquitectura se manejan varias estrategias de almacenamiento de la información, dado que los tenants tienen configuración propia.

Con el objetivo de determinar cuál era la estrategia de almacenamiento más usada y cuáles eran los criterios para su elección en arquitecturas multitenant, Karatas et al (2017) realizó un estudio donde se revisaron muchos trabajos de investigaciones de diversas fuentes indexadas.

Como primer paso, los autores describen las estrategias de almacenamiento de datos para delimitar el alcance:

- Aplicación y base de datos separadas.
- Aplicación compartida y base de datos separada.
- Aplicación y base de datos compartidas.

Como segundo paso, establecen su método de investigación identificando preguntas de investigación y algunas palabras reservadas con el objetivo de obtener los artículos más relevantes relacionados a la investigación.

Como resultado de las publicaciones examinadas, los autores generaron distribuciones usando diversas dimensiones, donde destacamos el Figura 1 que determina que la estrategia de almacenamiento preferida es la Aplicación y esquema compartido con un 35% de uso.

Finalmente, los autores concluyen que los sistemas multitenancy, solo son aplicados en un número limitado de áreas, lo que quiere decir que falta una mayor expansión y que a su vez la infraestructura de software tienes que prepararse para facilitar su implementación.

Siguiendo esta línea Odun Ayo et al. (2017) presentó un estudio exhaustivo de la literatura existente

Tabla 1

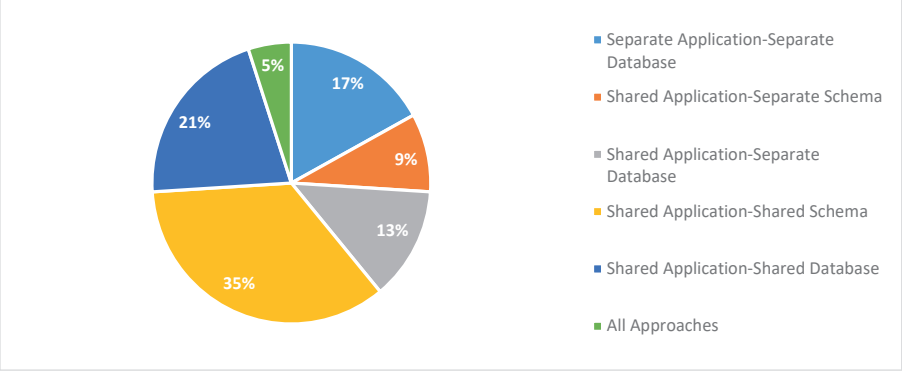
Framework de configuración estudiados.

#	FRAMEWORK	RELEASED	SIGNATURE
1	Properties	01.1996	import java.util.Properties
2	System	01.1996	System.getProperty
3	jConfig	10.2001	import org.jconfig.*
4	Preferences	07.2003	import java.util.prefs.*
5	Spring	06.2003	import org.springframework.*
6	Comomns	11.2005	import org.apache.commons.configuration.*
7	Constretto	05.2010	import org.constretto.*
8	Typesafe	12.2011	import com.typesafe.config
9	Deltaspike	10.2012	import org.apache.detaspike.*
10	Owner	12.2012	import org.aeonbits.owner.*
11	CFG4J	07.2015	import org.cgg4j.*

Fuente: adaptado de Sayagh (2017)

Figura 1

Estrategia de almacenamiento preferida.



Fuente: (Karatas et al., 2017)

relacionada al multitenancy. dicho trabajo permitió a los autores revisar diversos trabajos de investigación relacionadas a las aplicaciones multitenant y las tendencias de estas en la nube, con el propósito de describir las características esenciales que deben considerar y la arquitectura de las mismas.

De este estudio se desprenden las Características Multitenant que, de acuerdo con la literatura, las siguientes características son esenciales en una aplicación multitenant:

- Compartir recursos de hardware.
- Alto grado de configurabilidad.
- Aplicación y base de datos compartidas.

Como mencionan los autores, los entornos en la nube brindan una escalabilidad prácticamente ilimitada que permite resolver el problema de escalamiento de las aplicaciones, pero a su vez plantea nuevos problemas de seguridad y privacidad debido a la transferencia de tareas de procesamiento y almacenamiento a los proveedores cloud.

Algunos autores propusieron herramientas/frameworks para facilitar la gestión de los tenants en esta arquitectura, como Kale y Borhade (2013) que propusieron un framework SaaS, que a través del uso de licencias, permite gestionar múltiples tenants en una solo instancia, es decir, se crean diferentes portales webs con sus respectivas bases de datos y cada tenant es capaz de administrar su propio portal y a su vez gestionar tenants clientes, todo ello a través de una interfaz de usuario, lo cual permite reducir el tiempo y costo de desarrollo.

El framework esta implementado usando la tecnología .NET e implementa muchas características que un software de buena calidad debe considerar, como la portabilidad, personalización, seguridad y escalabilidad.

Basándose en los resultados de los análisis, los autores concluyeron que el framework es eficiente, debido a la reducción de la utilización de CPU y uso de memoria, por el solo hecho de usar una sola instancia para soportar múltiples tenants, también determinaron que es muy útil para reducir el costo y tiempo de desarrollo, ya que la gestión de tenants es de forma dinámica a través de las interfaces de usuario y son muy fáciles de usar por las características que provee el mismo framework.

Por otro lado Maenhaut et al (2015) propuso y evaluó el diseño de un framework que extiende aplicaciones multitenant existentes con el objetivo de lograr alta escalabilidad en términos de recursos de almacenamiento.

El framework propuesto intenta solucionar el problema de escalabilidad de las aplicaciones multitenant desplegadas en la nube cuando el número de tenants aumenta, ya que, aunque una única instancia es compartida por varios tenants, a su vez esta debe comportarse como una instancia privada para cada tenant con el propósito de garantizar la separación de datos, aislamiento de rendimiento y personalización por tenant.

El framework propuesto es agregado que contiene los 3 siguientes componentes con tareas específicas.

- El componente de Selección de Almacén de Datos es el responsable de la localización y ubicación de la base de datos correcta para el tenant que realiza las peticiones.
- El componente de Control de Acceso a Datos, que verifica si el tenant que realiza la petición tiene el permiso necesario para leer/modificar los datos seleccionados por el componente anterior.
- El componente de la Elasticidad de Datos es el más importante y está encargado de lograr la alta escalabilidad de los recursos de almacenamiento, es decir aumentando/disminuyendo el número de instancias base de datos y reubicando los datos de los tenants.

Bien y Dan Thu (2015) realizaron una investigación profunda identificando los requerimientos de un framework web multitenant del mundo real, para proponer un patrón que permita su construcción y utilización.

El patrón de desarrollo en mención genera altos niveles reutilización y modularidad que permitirá reducir tiempo y costos en el desarrollo de aplicaciones multitenant.

Como parte del trabajo de investigación, lo autores proveen un modelo, análisis y guías con los componentes e interacciones de estos, de forma detalladas para que sirva como base para la creación de frameworks web que puedan ser adaptados a diferentes sistemas.

La validación del producto fue realizada de forma práctica con la implementación de un framework multitenant usando el patrón propuesto y su posterior reutilización para la implementación de 3 sistemas de rubros completamente diferentes, donde satisfizo los requerimientos de los negocios.

Los autores concluyen que la creación de un framework web multitenant desde cero es muy complejo y tomaría mucho tiempo desarrollar uno, por este motivo identificaron las características que una aplicación web multitenant debe de cumplir en un entorno real, para proponer un patrón para implementar un framework reutilizable que satisfaga dichos requerimientos.

Los proveedores de nube están optando por arquitecturas multitenant con el objetivo de aumentar su eficiencia para el manejo de grandes volúmenes de datos, que últimamente se están volviendo más complejos, debido al aumento de las dimensiones de los datos, su falta de estructura y mayor necesidad de espacio de almacenamiento. Por este motivo los autores Sharma and Kaur (2019) proponen una arquitectura de base de datos multitenant usando almacenamiento de datos NoSQL.

Como resultado de un mapeo sistemático, los autores encontraron que la estrategia de almacenamiento de base de datos y aplicación compartida es la estrategia más popular en entornos multitenant y para comprobar la factibilidad de su propuesta, los autores la aplicaron a un caso de estudio convirtiendo una aplicación de un solo tenant a una multitenant.

Los resultados obtenidos mostraron que la arquitectura multitenant generó los valores deseados en menos tiempo para grandes almacenes de datos.

Mientras que Ziani (2014) exploró la configuración de une sistema ERP en un entorno multitenant en la nube, con el objetivo de proponer una arquitectura de servicio de configuración.

La arquitectura propuesta se basa en el soporte de múltiples bases de datos por tenant, una para sus transacciones y la otra para los datos de configuración, para esto el autor propone el Configuration Service, que está conformado por 3 módulos principales

- Autenticación, se definen 3 tipos diferentes de usuarios que solo pueden acceder a un número restringido de configuraciones con el objetivo de reducir la complejidad de la centralización de la configuración
- Asistente de configuración, es un conjunto de interfaces de usuarios que guían al usuario a través de diversos pasos para realizar las configuraciones
- Configuración mánager, es el responsable de almacenar y leer los datos de configuración, de y para la base de datos del tenant especifico.

El autor optó por el uso modelos preconfigurados (plantillas) por los desarrolladores para gestionar los cambios en las configuraciones basado en el aprendizaje que obtuvo de la revisión de la literatura y como trabajo futuro desea incluir las reglas de validación para las configuraciones.

III. MATERIAL Y METODOS

El trabajo de investigación tuvo un enfoque aplicado debido a la naturaleza del proyecto que consiste en implementar un framework para solucionar un problema real, Mientras que en términos de clasificación es catalogado No experimental, dado a que es realizada sobre situaciones ya existentes, donde las variables no pueden ser manipuladas.

La investigación inicio con una búsqueda intensiva de información relacionada a la externalización de la configuración y arquitecturas multitenant, así mismo, se investigaron las herramientas existentes y sus características que dan solución de alguna manera al problema de la externalización, como resultado se generaron las Tabla 2 y Tabla 3, donde se comparan las herramientas existentes y el framework propuesto, procediendo luego con la definición de la propuesta solución (Diseño de la arquitectura del framework, Implementación del framework y Evaluación de la propuesta).

Implementación del Framework.

El framework fue desarrollado bajo la metodología XP, debido a la mayor compatibilidad que se obtuvo como resultado del análisis comparativo basado en el trabajo de Matharu et al. (2015) para determinar la metodología ágil que mejor se adecua de acuerdo a ciertos parámetros.

Dado que la naturaleza del proyecto es un framework, se tenía que hacer mayor énfasis en el uso de técnicas para crear código con una calidad muy alta.

Definición de la propuesta solución

Se propuso un framework para la externalización de la configuración con las siguientes características que fueron extraídas como resultado del análisis comparativo Tabla 2 y Tabla 3.

- Exposición de APIs REST para la gestión de la configuración.
- Capacidad de configurar diversos motores de almacenamiento y poder ser extendido para adicionar más, si es requerido.

Tabla 2

Análisis comparativo de las herramientas I.

CARACTERISTICAS	HERRAMIENTAS				
	Consul	Conf	Zookeeper	Etcd	Archaius
Almacenamiento de la Configuración					
Motor de almacenamiento único	SI	NO	NO	SI	NO
Motor de almacenamiento intercambiable	NO	SI	NO	NO	SI
Gestión de la Configuración					
Actualización en tiempo real	NO	NO	NO	NO	SI
Control de versiones	SI	NO	NO	NO	NO
Soporte multitenant	SI	NO	NO	NO	NO
Clientes					
Interfaz web	SI	NO	NO	NO	NO
Interfaz de línea de comandos	SI	NO	NO	SI	NO
API REST	SI	NO	NO	SI	NO
JMX	NO	NO	SI	NO	SI
Licencia					
Open source	SI	SI	SI	SI	SI
De pago	SI	NO	NO	NO	NO
Community edition	SI	NO	NO	NO	NO

Fuente: Elaboración propia

Tabla 3

Análisis comparativo de las herramientas II.

CARACTERISTICAS	HERRAMIENTAS			
	Spring Cloud Config	AWS AppConfig	Azure App Configuration	Framework Propuesto
Almacenamiento de la Configuración				
Motor de almacenamiento intercambiable	SI	NO	NO	SI
Gestión de la Configuración				
Actualización en tiempo real	SI	NO	NO	SI
Control de versiones	SI	SI	SI	NO
Soporte multitenant	NO	SI	SI	SI
Cliente				
Interfaz web	NO	SI	Si	SI
Interfaz de línea de comandos	NO	SI	SI	NO
API REST	SI	SI	SI	SI
JMX	NO	NO	NO	NO
Licencia				
Open source	SI	NO	NO	SI
De pago	NO	SI	SI	NO
Community edition	NO	NO	NO	NO

Fuente: Elaboración propia

- Integración transparente con el contexto de las aplicaciones basadas en Spring Framework.

Con el objetivo de obtener un consumo confiable de los servicios de externalización, se priorizó la consistencia de datos y la disponibilidad, tomando como premisa el propósito del framework y el Teorema de CAP que indica, que ningún sistema distribuido es capaz de garantizar las tres características (Consistencia, Disponibilidad y Tolerancia de partición) de forma simultánea (Lomotey & Deters, 2013)

Instanciación de la Metodología

Inicia con la fase de exploración, donde se definieron las historias de usuario basados en los requerimientos del software y la arquitectura del sistema.

1. Consumir la configuración del proyecto almacenada en archivos planos.
2. Consumir la configuración del proyecto almacenada en base de datos relacional.
3. Consumir la configuración del proyecto almacenada en base de datos NoSql.

4. Gestionar la configuración del proyecto a través de perfiles de desarrollo y proyectos.
5. Registrar el historial de cambios de todos los eventos.
6. Facilitar la configuración y uso del framework para la aplicación cliente.
7. Permitir el almacenamiento local de la configuración en la aplicación.
8. Permitir la actualización de la configuración en tiempo real.
9. Gestionar la configuración del proyecto a través de tenants.
10. Permitir la creación de plantillas para dinamizar la creación de tenants.

Para el diseño de la arquitectura del software se utilizó el modelo de vistas de arquitectura 4+1 para describir la arquitectura del framework desde diferentes puntos de vista. En la Figura 2 se observan las conexiones físicas de los componentes del framework (Configuration Server y Configuration Client) a través de la vista física.

Luego en la **fase de Planificación** se establecieron las prioridades de las historias de usuario y la estimación de las mismas a través de la definición de sus tareas de programación.

En la **fase de iteraciones**, se realizaron los diagramas de clases y la implementación de las tareas de programación las cuales van evolucionan en cada iteración hasta obtener la solución definitiva. Y culminamos con la **puesta a producción** donde se alcanza la primera versión del framework.

V. RESULTADOS

Se involucró a profesionales de software con el objetivo de validar el framework propuesto a través de la medición de la satisfacción del usuario y se utilizó el método SQALE para evaluación del código fuente (Figura 5).

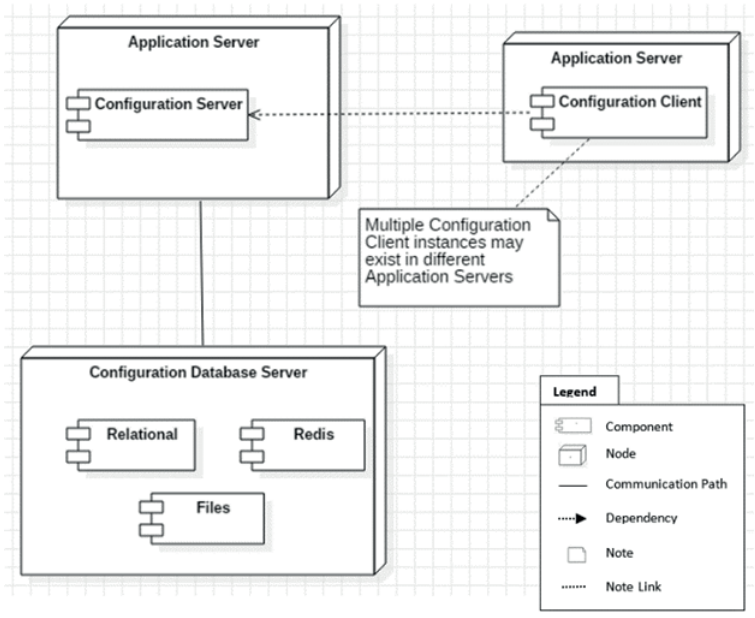
Medición de la Satisfacción del Usuario

En este proceso participaron 12 profesionales entre desarrolladores y arquitectos de software con una experiencia mínima de 5 años en el rubro.

Los participantes fueron capacitados en los procesos de configuración y utilización del framework. Luego se realizó la encuesta Post-Uso (PSSUQ) con el objetivo de medir la satisfacción de estos sobre la herramienta presentada y con los resultados obtenidos se procedió a realizar la validación de confiabilidad haciendo uso de la variable Alfa de

Figura 2

Vista física de la arquitectura propuesta.



Fuente: Elaboración propia

Cronbach. Tal como se muestra en la Tabla 4, se alcanzó 0.93 siendo el resultado de la confiabilidad excelente, garantizando los resultados obtenidos en la encuesta Post-Uso.

Tabla 4

Alfa de Cronbach.

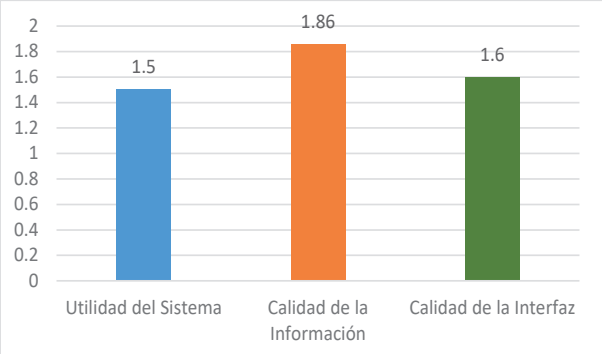
ALFA DE CRONBACH	CANTIDAD DE PARTICIPANTES
0.93	12

Fuente: Elaboración propia

Tomando como referencia PSSUQ (Lewis, 2002) se obtiene una puntuación general 1.66 y los valores por escala entre el rango de 1.50 y 1.86, entonces se infiere que los participantes quedaron satisfechos con el producto debido a que los valores resultantes son muy cercanos al valor 1(ver Figura 3).

Figura 3

Resultados de la encuesta de satisfacción de usuario.



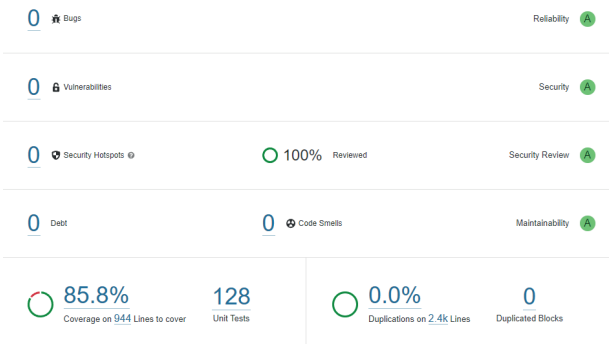
Fuente: Elaboración propia

Evaluación del Código Fuente

La herramienta SonarQube fue utilizada intensivamente con el objetivo de obtener las métricas relacionadas a la calidad del software, las cuales resultaron favorables para el framework con una clasificación de A, ausencia de deuda técnica y vulnerabilidades (Figura 4).

Figura 4

Métricas SonarQube.



Fuente: Elaboración propia

Y con el objetivo de integrar estos resultados como parte de la estrategia de negocio, se hizo uso del método SQALE para la evaluación del código fuente, ya que este implementa índices consolidados de los resultados de la deuda técnica por característica, permitiendo realizar toma de decisiones inteligente para las correcciones de los errores del software(figura 5).(Guaman et al., 2017)

Figura 5

Calificación SQALE.

Fuente: Elaboración propia

VI. DISCUSIÓN

El objetivo principal del trabajo de investigación fue desarrollar un framework para la externalización de la configuración en entornos multitenant, donde las decisiones de desarrollo fueron soportadas por los trabajos de investigación relacionados.

La investigación de las herramientas existentes (Sayagh et al., 2017), el análisis comparativo de las mismas (véase Tabla 2 y Tabla 3), la identificación de los requerimientos de un framework web multitenant (Bien & Dan Thu, 2015) y las tendencias de estas aplicaciones en la nube (Odun-Ayo et al., 2017) permitieron obtener las características esenciales con las que el framework propuesto debía contar para dar soporte a una arquitectura multitenant.

Adicionalmente, el estudio de Karatas (2017), permitió identificar la estrategia de almacenamiento preferida por los proyectos multitenant, respaldando así la decisión tomada para almacenar la configuración de los tenants en base de datos compartidas, cuando el motor de almacenamiento configurado en el framework propuesto, es base de datos.

Como resultado de la validación de la solución propuesta, se obtuvieron puntajes entre el rango 1.50 y 1.86 (véase el Gráfico 2), el cual indica que se obtuvo una alta satisfacción de uso por parte de los participantes, tomando como referencia a Lewis (2002).

Por otro lado, el trabajo de investigación beneficiará a diversos sectores, como el sector educativo, donde educandos y educadores podrán utilizar y modificar el framework a su conveniencia y de forma práctica en los proyectos a desarrollar durante su formación, sin importar el tamaño de estos y sin incurrir en gastos por ser de naturaleza open source.

En el Sector empresarial las organizaciones podrán adoptar la herramienta en sus sistemas existentes para dar solución a los problemas de externalización, y mitigar los desafíos que son generados por los sistemas legados, donde usualmente no se pueden realizar cambios drásticos.

Finalmente, el framework intenta fortalecer la capacidad de externalizar la configuración en entornos multitenant, donde no existe una solución que se integre de forma natural a las características de la arquitectura en mención.

VII. CONCLUSIONES

El presente trabajo de investigación propuso un framework para externalizar la configuración de aplicaciones, con la finalidad de facilitar y agilizar la implementación de aplicaciones basadas en arquitecturas multitenant, para esto, el framework contempla la interacción de la aplicación que provee y centraliza la configuración y la aplicación que consume la configuración, pudiendo ser esta segunda, una aplicación existente.

Se desarrolló el framework tomando como base un análisis comparativo de las herramientas más populares y existentes que brindan una solución parcial al problema de la externalización, de donde se pudo extraer las características que el framework debía de brindar para ser parte de la lista de opciones de los frameworks disponibles.

Se utilizaron diversos métodos para medir los atributos de calidad del software, como el uso del método SQALE para validar la calidad del código fuente, a través de la mantenibilidad, fiabilidad y deuda técnica, y por otro lado el uso de PSSUQ para medir la satisfacción del usuario con relación al uso del framework

REFERENCIAS BIBLIOGRÁFICAS

[8] Balalaie, A., Heydarnoori, A., & Jamshidi, P. (2016). Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture. IEEE Software, 33(3), 42-52. https://doi.org/10.1109/MS.2016.64

[9] Bien, N. H., & Dan Thu, T. (2015). Multi-tenant web application framework architecture pattern. 2015 2nd National Foundation for Science and Technology Development Conference on Information and Computer Science (NICS), 40-48. https://doi.org/10.1109/NICS.2015.7302219

[10] ckittel. (2023). External Configuration Store pattern—Azure Architecture Center. https://learn.microsoft.com/en-us/azure/architecture/patterns/external-configuration-store

[11] Guaman, D., Quezada-Sarmiento, P. A., Barba-Guaman, L., & Enciso, L. (2017). Use of SQALE and tools for analysis and identification of code technical debt through static analysis. 2017 12th Iberian Conference on Information Systems and Technologies (CISTI), 1-7. https://doi.org/10.23919/CISTI.2017.7975677

[12] Hall, G. M. (2010). Data Access Layer. En G. M. Hall, Pro WPF and Silverlight MVVM (pp. 163-183). Apress. https://doi.org/10.1007/978-1-4302-3163-9_8

[13] Hofmann, M., Schnabel, E., & Stanley, K. (2016). Microservices Best Practices for Java.

[14] Kale, S. S., & Borhade, R. H. (2013). Development of Multitenant SaaS framework at single instance and with zero effort multitenancy. 2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI), 834-839. https://doi.org/10.1109/ICACCI.2013.6637284

[15] Karatas, G., Can, F., Dogan, G., Konca, C., & Akbulut, A. (2017). Multi-tenant architectures in the cloud: A systematic mapping study. 2017 International Artificial Intelligence and Data Processing Symposium (IDAP), 1-4. https://doi.org/10.1109/IDAP.2017.8090268

[16] Lewis, J. R. (2002). Psychometric Evaluation of the PSSUQ Using Data from Five Years of Usability Studies. International Journal of Human-Computer Interaction, 14(3-4), 463-488. https://doi.org/10.1080/10447318.2002.9669130

[17] Lomotey, R. K., & Deters, R. (2013). Reliable Consumption of Web Services in a Mobile-Cloud Ecosystem Using REST. 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, 13-24. https://doi.org/10.1109/SOSE.2013.10

[18] Maenhaut, P.-J., Moens, H., Ongenae, V., & De Turck, F. (2015). Design and evaluation of a hierarchical multi-tenant data management framework for cloud applications. 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), 1208-1213. https://doi.org/10.1109/INM.2015.7140468

[19] Matharu, G. S., Mishra, A., Singh, H., & Upadhyay, P. (2015). Empirical Study of Agile Software Development Methodologies: A Comparative Analysis. ACM SIGSOFT Software Engineering Notes, 40(1), 1-6. https://doi.org/10.1145/2693208.2693233

[20] Odun-Ayo, I., Misra, S., Abayomi-Alli, O., & Ajayi, O. (2017). Cloud Multi-Tenancy: Issues and Developments. Companion Proceedings of The10th International Conference on Utility and Cloud Computing, 209-214. https://doi.org/10.1145/3147234.3148095

[21] Sayagh, M., Dong, Z., Andrzejak, A., & Adams, B. (2017). Does the Choice of Configuration Framework Matter for Developers? Empirical Study on 11 Java Configuration Frameworks. 2017 IEEE 17th International Working Conference on Source Code Analysis and Manipulation (SCAM), 41-50. https://doi.org/10.1109/SCAM.2017.25

[22] Sharma, A., & Kaur, P. (2019). A Multitenant Data Store Using a Column Based NoSQL Database. 2019 Twelfth International Conference on Contemporary Computing (IC3), 1-5. https://doi.org/10.1109/IC3.2019.8844906

[23] Ziani, D. (2014). Configuration Inerpsaas Multi-Tenancy. International Journal of Computer Science, Engineering and Information Technology, 4(2), 57-62. https://doi.org/10.5121/ijcseit.2014.4206

Financiamiento:

Propio.

Conflictos de interés:

Los autores declaran no tener conflictos de interés.

Contribuciones de autoría:

Primer autor: Jose Miguel Mori Peña: Investigación, análisis formal, redacción, software

Segundo autor: Fany Yexenia Sobero Rodriguez: Metodología, redacción (revisión).

Agradecimiento

A mi asesora de tesis la Profesora Fany Sobero Rodriguez, por la orientación que me brindó para la realización y culminación de esta tesis.

A la profesora Nora La Serna, por su apoyo y amistad que me permitió aprender mucho más de lo estudiado en el proyecto