METODOLOGÍA PARA LA EJECUCIÓN DE PRUEBAS NO FUNCIONALES (CARGA Y ESTRÉS) DE SOFTWARE PARA KACTUS-HCM EN DIGITAL WARE

AUTORA

NATALIA ANDREA CONTRERAS MEJÍA

UNIVERSIDAD DE PAMPLONA
INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA, NORTE DE SANTANDER
2019

METODOLOGÍA PARA LA EJECUCIÓN DE PRUEBAS NO FUNCIONALES (CARGA Y ESTRÉS) DE SOFTWARE PARA KACTUS-HCM EN DIGITAL WARE.

AUTORA

NATALIA ANDREA CONTRERAS MEJÍA

TRABAJO DE GRADO PRESENTADO COMO REQUISITO PARA OPTAR AL TITULO DE INGENIERO EN TELECOMUNICACIONES

DIRECTOR

ING. GUSTAVO ADOLFO QUIJADA MACUART

CODIRECTOR

MSc. EDWIN GIOVANNY BARRETO LIZCANO

UNIVERSIDAD DE PAMPLONA
INGENIERÍA EN TELECOMUNICACIONES
DEPARTAMENTO ELECTRÓNICA, ELÉCTRICA, SISTEMAS Y TELECOMUNICACIONES
FACULTAD DE INGENIERÍAS Y ARQUITECTURA
PAMPLONA, NORTE DE SANTANDER
2019

METHODOLOGY FOR THE EXECUTION OF NON-FUNCTIONAL TESTS (LOADING AND STRESS) OF SOFTWARE FOR KACTUS-HCM IN DIGITAL WARE

AUTHOR

NATALIA ANDREA CONTRERAS MEJÍA

UNIVERSIDAD DE PAMPLONA
TELECOMMUNICATIONS ENGINEERING
ELECTRONIC DEPARTMENT, ELECTRICAL, SYSTEMS AND TELECOMMUNICATIONS
FACULTY OF ENGINEERING AND ARCHITECTURE
PAMPLONA, NORTH OF SANTANDER
2019

METHODOLOGY FOR THE EXECUTION OF NON-FUNCTIONAL TESTS (LOADING AND STRESS) OF SOFTWARE FOR KACTUS-HCM IN DIGITAL WARE

AUTHOR

NATALIA ANDREA CONTRERAS MEJÍA

GRADE WORK PRESENTED AS A REQUIREMENT TO CHOOSE THE TITLE OF ENGINEER IN TELECOMMUNICATIONS

DIRECTOR

ING. GUSTAVO ADOLFO QUIJADA MACUART

CODIRECTOR

MSc. EDWIN GIOVANNY BARRETO LIZCANO

UNIVERSIDAD DE PAMPLONA
TELECOMMUNICATIONS ENGINEERING
ELECTRONIC DEPARTMENT, ELECTRICAL, SYSTEMS AND TELECOMMUNICATIONS
FACULTY OF ENGINEERING AND ARCHITECTURE
PAMPLONA, NORTH OF SANTANDER
2019

AGRADECIMIENTOS

A mis profesores de la Universidad de Pamplona, por los aprendizajes brindados a nivel personal y profesional.

A mis amigos y compañeros con los que pude compartir este camino.

A los Ingenieros Gustavo Quijada y Edwin Barreto, por su dirección y guía en este proyecto.

A la empresa ***DIGITAL WARE*** por darme la oportunidad de formar parte de su equipo de trabajo poniendo en práctica mis habilidades y adquiriendo nuevos conocimientos.

“El futuro nos reserva algo mejor, siempre que tengamos el valor de seguir intentándolo, seguir trabajando, seguir luchando”

-Barack Obama-

“A mis padres y Lu, por su apoyo incondicional y ser parte de este arduo camino para culminar mi sueño, por alentarme en mis momentos difíciles y siempre creer en mí”

Natalia Contreras Mejía.

ÍNDICE

TABLAS

FIGURAS

RESUMEN

La compañía ***DIGITAL WARE*** destacada en el sector de tecnología y localizada en la ciudad de Bogotá DC, fue el escenario elegido para la ejecución de la práctica empresarial; llevando a cabo funciones tales como: la definición, documentación y verificación de casos de uso para la automatización de diferentes procesos del software ***KACTUS HCM***, ofertado por la misma. Inicialmente en la automatización fueron elegidos procesos de línea base para el producto como lo son: el liquidador de nómina, el registro de incapacidades, *self service, KACTUS RL*, entre otros; con el fin de darle a la compañía pruebas funcionales que alivianan el proceso de *testing.*

De este modo surgió la necesidad de complementar esta labor con el desarrollo de una metodología para aplicar las pruebas no funcionales a este producto, ya que con anterioridad no se habían tenido en cuenta. Una vez aprobada esta iniciativa por parte de la compañía se da paso al desarrollo metodológico comprendido por estas etapas: el establecimiento de la necesidad de las pruebas no funcionales, el diseño de casos de uso para la integración con las mismas, los parámetros estructurales, el desarrollo del caso de prueba y la ejecución de las pruebas de carga y estrés. Haciendo uso de herramientas como: Visual Studio Enterprise 2017, ***KACTUS-HCM***, SQL Server Management.

Obteniendo como resultado que el proceso del liquidador de nómina (***KNmLinom***) del software ***KACTUS-HCM*** soporta concurrencia con las condiciones y recursos otorgados por la compañía para la ejecución de las pruebas de carga y estrés como lo son las características del servidor y la base de datos empleada. También con estas pruebas se notó que los tiempos de respuesta por proceso fueron aumentando a medida que se hicieran más solicitudes.

Palabras Clave: AUTOMATIZACIÓN, KACTUS-HCM, METODOLOGÍA, PRUEBAS NO FUNCIONALES, SOFTWARE.

ABSTRACT

The company ***DIGITAL WARE*** highlighted in the technology sector and located in the city of Bogota DC, was the scenario chosen for the execution of business practice, carrying out functions such as: definition, documentation and verification of use cases for the automation of different processes of the software ***KACTUS HCM***, offered by it. Initially in the automation were chosen baseline processes for the product such as: the payroll liquidator, the registry of disabilities, self-service, KACTUS RL, among others, in order to give the company functional tests that alleviate the testing process.

Thus, the need arose to complement this work with the development of a methodology to apply the non-functional tests to this product, since previously they had not been taken into account. Once this initiative has been approved by the company, a step is taken to the methodological development understood by these stages: the establishment of the need for non-functional tests, the design of use cases for integration with them, the structural parameters, the development of the test case and the execution of load and stress tests. Making use of tools such as: Visual Studio Enterprise 2017, ***KACTUS-HCM***, SQL Server Management.

Obtaining as a result that the process of the payroll liquidator (***KNmLinom***) of the ***KACTUS-HCM*** software supports concurrence with the conditions and resources granted by the company for the execution of load and stress tests such as the characteristics of the server and the database used. Also with these tests it was noted that the response times per process were increasing as more requests were made.

Keywords: AUTOMATION, KACTUS-HCM, METHODOLOGY, NON-FUNCTIONAL TESTS, SOFTWARE.

INTRODUCCIÓN

En el presente proyecto se elabora una metodología para conocer cómo trabaja el producto ***KACTUS-HCM*** ofrecido por la empresa ***DIGITAL WARE*** como un *software* de nómina y gestión humana para empresas de diferentes áreas en la industria. Esta metodología se lleva a cabo con el fin de ver reflejadas las necesidades de *hardware* al momento en que este *software* presente una alta demanda de usuarios y genere inconvenientes como lo son: los cuellos de botella, deficiencia en el tiempo de recuperación, problemas de seguridad, entre otros.

De este modo el presente documento inicia con la descripción del lugar donde se ejecutó el proyecto y se presentan algunas investigaciones que por la pertinencia de sus contenidos fueron tomadas como base para esta metodología; dando paso a la descripción detallada de cada una de las etapas y procesos que debían ser tenidos en cuenta para la aplicación de las pruebas, así como también las herramientas implementadas. Es importante resaltar que las pruebas no funcionales son punto clave para esta metodología, ya que a través de ellas se logra tener una idea del funcionamiento y dimensionamiento que necesita este software para trabajar adecuadamente, tal como se evidencia en los resultados descritos en el último apartado de este documento.

Finalmente, con la implementación de esta metodología ***DIGITAL WARE*** ofrecerá a sus clientes las condiciones específicas con las que deben contar de acuerdo a la demanda de usuarios que estos tengan; y de este modo poder gozar del *software* sin inconvenientes algunos de retraso o mal funcionamiento.

INTRODUCTION

This project elaborates a methodology to learn how the product ***KACTUS-HCM*** offered by the company ***DIGITAL WARE*** works as a payroll and human management software for companies in different areas of industry. This methodology is carried out in order to see reflected the hardware needs at the time that this software presents a high demand for users and generates inconveniences such as: bottlenecks, deficiency in recovery time, security problems, among others.

In this way, this document begins with a description of the place where the project was executed and presents some researches that, due to the pertinence of their contents, were taken as a base for this methodology; giving way to a detailed description of each one of the stages and processes that had to be taken into account for the application of the tests, as well as the implemented tools. It is important to emphasize that non-functional tests are a key point for this methodology, since through them it is possible to have an idea of the functioning and dimensioning that this software needs to work properly, as evidenced in the results described in the last section of this document.

Finally, with the implementation of this methodology, DIGITAL WARE will offer its clients the specific conditions that they must have according to the demand of the users that they have; and in this way be able to enjoy the software without any inconvenience of delay or malfunction.

1. JUSTIFICACIÓN

"***DIGITAL WARE*** es una compañía del sector de tecnología especializada en *Software* para el diseño e implantación de soluciones empresariales en diferentes áreas"[1] Uno de los productos más destacados es ***KACTUS-HCM***, el cual es un *software* de nómina y gestión humana para empresas de diferentes sectores en la industria. Se hace imprescindible que al momento de adquirir este producto las empresas cuenten con ciertas características a nivel estructural, para así poder garantizar que el mismo pueda desempeñarse de manera óptima.

Como se menciona anteriormente ***KACTUS-HCM*** es uno de los productos más solicitados; este producto se encarga de realizar toda la gestión desde que un nuevo empleado ingresa a la empresa (Hoja de vida, Contrato, Cuentas, entre otros) y asiste todo su proceso en cuanto a nómina (Liquidación, Vacaciones, Primas, entre otros). Sin embargo, al ser un *software* que por su variedad de programas es concurrido paralelamente, puede presentar inconvenientes en su ejecución debido al dimensionamiento que se tuvo en cuenta en la implantación del producto, conflictos de red y errores de programación. Estos inconvenientes hacen necesario la elaboración de pruebas no funcionales puesto que ***DIGITAL WARE*** no las ha realizado anteriormente.

Las pruebas no funcionales permiten identificar las limitaciones del *software*, estas pruebas hacen parte de la calidad del mismo. Por tanto, esta metodología permitirá a las personas que intervienen en el proceso (manufactura de la solución de *software*) identificar los alcances y condiciones con los que va a contar un programa al ser ejecutado, también entregará métricas (tiempos de respuesta, cantidad de usuarios que pueden ingresar a la aplicación al mismo tiempo y capacidad de la infraestructura) con las cuales se podrá evaluar los riesgos de liberar una corrección o una nueva versión del *software.*

De acuerdo con lo anterior surge la siguiente pregunta problematizadora:

¿Cómo incidir en el rendimiento del proceso ***KNmLinom*** del software ***KACTUS-HCM*** mediante la ejecución de pruebas funcionales y no funcionales, de manera que se puedan establecer recomendaciones mínimas de operación?

[1]Digital Ware Tecnology that changes People´s Lives. [En línea]. Bogotá DC. [Consultado: 19 de septiembre de 2018]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

1.1 DELIMITACIÓN

1.1.1 OBJETIVOS

1.1.1.1 OBJETIVO GENERAL

- Elaborar una metodología para la ejecución de pruebas no funcionales (carga y estrés) de software para ***KACTUS-HCM*** en ***DIGITAL WARE***.

1.1.1.2 OBJETIVOS ESPECÍFICOS

- Identificar el funcionamiento del *software* ***KACTUS-HCM*** entorno a las pruebas y los requisitos no funcionales.

- Analizar los estándares y metodologías que implementan las pruebas no funcionales de software para ajustar la metodología a las necesidades de la empresa ***DIGITAL WARE.***

- Diseñar la metodología para la ejecución de pruebas no funcionales (carga y estrés) de software para ***KACTUS-HCM.***

1.2 ALCANCES Y LIMITACIONES

En este proyecto se implementa una metodología con el fin de ver reflejadas las necesidades de hardware al momento en que el *software* ***KACTUS-HCM*** presentase una alta demanda de usuarios. A través de estas pruebas se logra tener una idea del funcionamiento y dimensionamiento que necesita este software para trabajar adecuadamente.

Para las debidas pruebas no funcionales se utilizó la *herramienta Visual Studio Enterprise 2017*. Este proyecto fue desarrollado en el *software* ***KACTUS-HCM*** de la empresa ***DIGITAL WARE*** ubicada en la ciudad Bogotá D.C., la duración del proyecto estuvo contemplada en un período no superior a los 6 meses.

2. MARCO INSTITUCIONAL

2.1 DIGITAL WARE

La práctica empresarial fue desarrollada en ***DIGITAL WARE***, la cual como lo menciona en su página oficial es una:

> Empresa del sector de tecnología especializada en *Software ERP, Software* de Nómina y Gestión Humana y *Software* para IPS y Clínicas, con más de 20 años en el mercado, líder en diseño e implantación de soluciones empresariales en las áreas de RRHH, Finanzas, Logística, Manufactura, Seguridad, Petróleos, Energía, Cajas de Compensación, Gobierno, Educación y Salud. Ganadora de varios premios, entre ellos la Orden de la Democracia SIMÓN BOLÍVAR en el grado de Cruz Comendador otorgada por el Congreso de la República, y la Medalla al Mérito de las Comunicaciones MURILLO TORO otorgada por el Ministerio de Tecnologías de la información y las comunicaciones, además del Premio Portafolio a la Innovación [2]

2.2 MISIÓN

"Mejorar la vida de las personas y ayudar a las empresas a cumplir sus metas, utilizando el conocimiento, el talento y la tecnología, soportados en visión, pasión, conciencia, disciplina e intensidad" [3]

2.3 VISIÓN

"Nos convertiremos en la más innovadora, motivadora y servicial empresa de la tecnología en américa latina en el desarrollo de plataformas informáticas con foco en:

-ERP: *Enterprise Resource Plannig*, planificación de recursos empresariales.

- HR: *Human Resources* - Recursos Humanos.

- HIS*: Hospital Information System* - Sistemas de Información Hospitalaria

[2] Ibid., p.02.

[3] Ibid., p.01.

- *BPM:* *Business Process Management*- Gestión por Procesos Inteligentes"[4]

2.4 CARGO EJECUTADO

El cargo ejecutado fue Pasante de Ingeniería en Telecomunicaciones durante un periodo de 6 meses, comprendido entre el 7 de junio y el 6 de diciembre del año 2018; bajo la supervisión del Ingeniero Edwin Barreto, líder del Área de Automatización. Desempeñando las siguientes funciones:

- Creación de casos de uso para la automatización de procesos de *KACTUS.*
- Validación y ejecución de casos de uso de la automatización de procesos de *KACTUS.*
- Pruebas funcionales para el proceso de automatización de procesos de *KACTUS.*

Los procesos anteriormente mencionados son los siguientes:

- Liquidador de Nómina (KNmLinom)
- Novedades de seguridad social (KNmNovss)
- Bases de retención en la fuente (KNmBasre)
- Prenómina (KNmPreno)
- Incapacidades (KNmIncap)
- Novedades de nómina (KNmNoved)
- Ausentismos (KNmAusen)
- Días no trabajados (KNmDiasn)
- Proceso de acumulados (KNmProac)
- Acumulados (KNmAcumu)
- Selfservice

[4] Digital Ware Tecnology that changes People´s Lives. [En línea]. Bogotá DC.Reglamento interno [Consultado: 09 de Noviembre de 2018]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

- Reclutamiento web (Kactus RL)
- Checklist Maestros gestión humana
- Checklist Maestros Nómina

Cabe resaltar que, cada una de ellas aportó al diseño y ejecución de la metodología planteada en este proyecto.

3. MARCO REFERENCIAL

3.1 ESTADO DEL ARTE

El presente estado contempla algunas de las investigaciones más relevantes de la bibliografía disponible acerca de pruebas no funcionales, que permite recopilar diversas perspectivas conceptuales y metodológicas a fin de garantizar un proceso analítico, comparativo y constructivo en este proyecto.

La investigación propuesta por Álvarez y Ponce[5] titulada: Metodología para el desarrollo de pruebas de *software* basada en métricas de funcionalidad y rendimiento del año 2013, mediante un enfoque de investigación de tipo cuantitativo desarrollado en 5 fases; establece una metodología de pruebas de *software*, orientada a pruebas de rendimiento y funcionalidad en aplicaciones *web*. En búsqueda que la misma permitiese evaluar su calidad en términos de funcionalidad y rendimiento a partir del análisis de una muestra representativa en tres sistemas: ***MRI-SEIPA-E1, RIB-SIGRA, Y SIG-SEIPA-E2,*** que en un primer momento se estableció en 80 casos de uso, pero debido a la complejidad de manejo se redujo a 74. Para el análisis de resultados se clasificó la información en dos aspectos: análisis de errores funcionales y no funcionales; el primero de ellos evidencia que la aparición de los mismos tiene una similitud en los tres sistemas y que en un porcentaje superior fueron encontrados con pruebas automáticas y su minoría con pruebas manuales; y finalmente, los errores no funcionales destacan que los programadores inexpertos son un factor determinante en el número de errores, así como también que los requisitos no funcionales incrementan la ocurrencia de errores funcionales.

Del mismo modo Echeverría y Abella[6] en el año 2014 proponen el *testing* como práctica para evaluar la eficiencia en aplicaciones *Web*; estableciendo para dicho

[5] ÁLVAREZ HERNÁNDEZ, Jorge Eduardo y PONCE CRUZ, Francisco Javier. Metodología para el desarrollo de pruebas de software basada e métricas de funcionalidad y rendimiento [en línea]. Trabajo Terminal 2013-B061.México, DF. Instituto Politécnico Nacional Escuela Superior de Computo. Departamento de Formación Integral e Institucional. Comisión Académica de Trabajo Terminal, 2015. p. 36. [Consultado 17 Agosto de 2018]. Disponible en Internet: https://goo.gl/hhLJgK

[6] ECHEVERRÍA PEREZ, Devis y AVELLA PAUMIER, Ariannis. Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web [en línea]. En: Centro Nacional de Calidad de Software: Revista Latinoamericana de Software.2014, vol.2 , no. 5. p. 307-308. [Consultado: 17 Agosto de 2018]. Disponible en Internet: file:///E:/255-Texto%20del%20art%C3%83_culo-551-1-10-20141027.pdf

propósito una serie de especificaciones en cuanto al mismo, tales como: los tipos de test basados en características de: Rendimiento, Carga y Estrés. La propuesta se basa en un procedimiento a seguir, compuesto por una serie de fases continuas que precisamente guían su ejecución. De ello puede destacarse la utilización de herramientas automatizadas particularmente la herramienta **JMETER**, que brinda mayor confiabilidad debido a que reduce en gran medida el error humano, así como también la disminución de inversión a realizar.

Para el caso particular de la creación de una base de casos de posibles soluciones de errores de **HTTP**, fue una acción que los investigadores consideraron de relevancia debido a los posibles problemas que pudiesen presentarse en los equipos de desarrollo de *software*. En cuanto a los resultados obtenidos la investigación logró establecer la evaluación de la característica de Eficiencia de la norma ***ISO/IEC 9126***, se obtuvieron indicadores, y se brindó mayor organización dentro de las pruebas de rendimiento, estrés y carga.

Finalmente, Acosta, Nieto y Barahona[7] de la Universidad de Cundinamarca en el año 2015 aportan a la iniciativa de una Metodología para la evaluación de calidad de los productos *software*; para ello fue necesario basarse en el ***Framework UML 2.0*** y los criterios a tener en cuenta fueron los siguientes: Actividades, casos de uso, secuencia, colaboración, clases y despliegue. Dentro de dicha metodología también se integraron al proceso evaluativo la gestión de pruebas, pruebas funcionales, y las pruebas de carga y rendimiento. En cuanto a los escenarios manejados se seleccionaron herramientas tales como: ***QABook, Selenium IDE y LoadUI***, A Partir de ellos los resultados evidencian que al establecerse y aplicar herramientas libres que permiten consolidar una metodología, es posible concertar la capacidad competitiva y la calidad de *software* desarrollado. Así como también que se debe fortalecer el ámbito de la seguridad informática para el manejo de la información y que no es tan ajena la posible vinculación del servicio para la evaluación de aplicaciones móviles; abarcando así aspectos generales de cualquier tipo de plataforma.

---

[7] ACOSTA MESA, Ingrid Ximena, NIETO GUEVARA, Erika Alejandra y BARAHONA RODRÍGUEZ, Cesar Yesid. Metodología para la evaluación de calidad de los productos software de la Universidad de Cundinamarca [en línea]. En: ENGI: Revista Electrónica de la Facultad de Ingeniería. Universidad de Cundinamarca. 2015, vol. 3, no 2. p. 15-16. [Consultado 18 de Agosto de 2018]. Disponible en Internet: https://goo.gl/dWyGHy

4. MARCO TEÓRICO

El siguiente contenido se encuentra orientado a fundamentar el diseño metodológico de este proyecto y brindar la conceptualización adecuada a los términos que dentro de esta metodología serán utilizados.

4.1 PRUEBAS DE SOFTWARE

Para establecer una definición precisa sobre pruebas de *software*, se toma como referencia la propuesta por Parada en su tesis doctoral: contribución a la gestión de los procesos de pruebas de *software* y servicios, donde las mismas se presentan como:

> Una actividad que se realiza con el objetivo de detectar fallos y evaluar las características del *software*. Las pruebas regulan la ejecución de los proyectos y garantizan la calidad del *software* desarrollado. Desde el modelo de desarrollo en cascada hasta la aparición de las metodologías ágiles, las prueban han pasado de ser una simple etapa en el proceso de desarrollo a constituirse en un conjunto de etapas que controlan la duración del ciclo de vida, la calidad y la confiabilidad del *software* desarrollado[8]

Del mismo modo, desde el instituto técnico de Florida, Kanner[9] manifiesta que mediante la realización de las pruebas se tendrán datos sobre la calidad del producto y no únicamente la detección de fallas.

> El hecho de someter los sistemas y la documentación a pruebas rigurosas puede ayudar a reducir el riesgo de complicaciones durante las operaciones y contribuir a la calidad del sistema de *software*, siempre que los defectos detectados se corrijan antes de que el sistema se ponga a disposición para su

[8] PARADA GELVÉZ, Hugo Alexer. Contribución a la gestión de los procesos de pruebas de software y servicios [En línea]. Tesis doctoral Ingeniería de Sistemas Telemáticos. Madrid, España. Universidad Politécnica de Madrid. Departamento de ingeniería de Sistemas Informáticos. Escuela Técnica Superior de Ingenieros de Telecomunicación ,2010. p. 82. [Consultado 18 Agosto de 2018] Disponible en Internet: http://oa.upm.es/4019/1/HUGO_ALEXER_PARADA_GELVEZ.pdf

[9] KANER, Cem. Actualización de algunos conceptos básicos en pruebas de software [En Línea].Cem Kaner, JD, Ph.D. Rockledge FL, Estados Unidos. (21 de Noviembre de 2006), p. 1. [Consultado: 24 de Noviembre de 2018]. Disponible en internet: http://kaner.com/?p=30

uso operativo. Las pruebas de *software* también pueden servir para cumplir requisitos contractuales o legales, o estándares específicos del sector[10]

Una vez establecida la definición, es de relevancia mencionar que la realización de esta prueba en el producto ***KACTUS-HCM*** favorece a la compañía, ya que se va a ofertar un producto de mejor calidad. A continuación, se presentan diversos tipos de pruebas, debido a que en la aplicación de las mismas se contemplan el desarrollo y la metodología utilizada en las mismas.

4.2 PRUEBAS DE CAJA NEGRA

Desde el Departamento de Teoría de la señal y Comunicaciones, de la Universidad Politécnica de Madrid, Sánchez en el año 2015, establece que la técnica de pruebas de caja negra "consiste en ver el programa que queremos probar como una caja negra despreocupándose del comportamiento interno y concentrando el esfuerzo en encontrar el comportamiento incorrecto, de acuerdo a las especificaciones de dicho programa, teniendo solo en cuenta las entradas y salidas del mismo"[11]

En efecto otra definición de relevancia para el caso es la de Muller quien afirma que esta técnica: "puede utilizarse para lograr objetivos de cobertura de entrada y salida, con entradas humanas, vía interfaces a un sistema, o parámetros de interfaz de las pruebas de integración"[12] En la siguiente tabla se presentan las condiciones, reglas y acciones que se deben tener para este tipo de técnica:

[10] International Software Testing Qualifications Board. Probador Certificado Programa de estudio de nivel básico. [Medio Electrónico]. Versión 2011. Copyright © ISTQB®. 31 de Marzo de 2011. p.11. [Consultado 24 de Noviembre de 2018]. Disponible en: http://www.sstqb.es/ficheros/sstqb_file95-637831.pdf

[11] SÁNCHEZ PEÑO, José Manuel. Pruebas de Software fundamentos y técnicas [En Línea]. Trabajo terminal Ingeniería y sistemas de telecomunicciones. Madrid, España. Universidad Politécnica de Madrid. Departamento de Teoría de la Señal y Comunicaciones, 2015. p. 22. [Consultado: 26 de Noviembre de 2018]. Disponible en: http://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf

[12] T. Müller, Libro Programa de Estudio de Nivel Básico para Probador Certificado, istqb, Version 2013, p. 14. Disponible en: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

CONDICIONES	La totalidad de las acciones que puedan presentarse.
REGLAS DE LAS CONDICIONES	El valor que debe asociarse a cada condición
ACCIONES	Listado de pasos a seguir cuando se presenta una condición

Tabla 1. Condiciones, reglas y acciones de la técnica de caja negra.

Fuente: Elaborada por el Autor

4.3 PRUEBAS DE CAJA BLANCA

De acuerdo a Zamora en su proyecto análisis de los procesos de verificación y validación en las organizaciones software del año 2011:

> Una prueba de caja blanca se define como una prueba de las propiedades internas de un objeto, mediante el conocimiento de funciones internas. Estas pruebas son ejecutadas por los desarrolladores. La prueba unitaria y la de integración son pruebas bastante conocidas de caja blanca. Al igual que la Revisión Estructurada, estas pruebas son efectuadas en las etapas iniciales del ciclo de vida antes de llegar a las pruebas de caja negra. Por otro lado, las pruebas de caja blanca son relativamente baratas porque se requiere menor comunicación y porque el análisis es más sencillo (la persona que detecta los defectos es regularmente la misma persona que hace las reparaciones. Además, se prueban menos objetos)[13]

Del mismo modo en una de las publicaciones del grupo *TestingBaires* en su blog se afirma que: “Aunque las pruebas de caja blanca son aplicables a varios niveles —unidad, integración y sistema—, habitualmente se aplican a las unidades de *software*. Su cometido es comprobar los flujos de ejecución dentro de cada unidad

[13] ZAMORA HERNÁNDEZ, Jorge. Análisis de los procesos de verificación y validación en las organizaciones software [En Línea]. Proyecto fin de carrera Ingeniería del Sotfware.Madrid, España. Universidad Carlos III de Madrid. Departamento de Ingeniería del Software, 2011. p.91. [Consultado: 20 de Diciembre de 2018]. Disponible en Internet : https://e-archivo.uc3m.es/bitstream/handle/10016/12880/PFC_Jorge_Zamora_Hernandez.pdf

(función, clase, módulo, etc.) pero también pueden testear los flujos entre unidades durante la integración, e incluso entre subsistemas, durante las pruebas de sistema".[14]

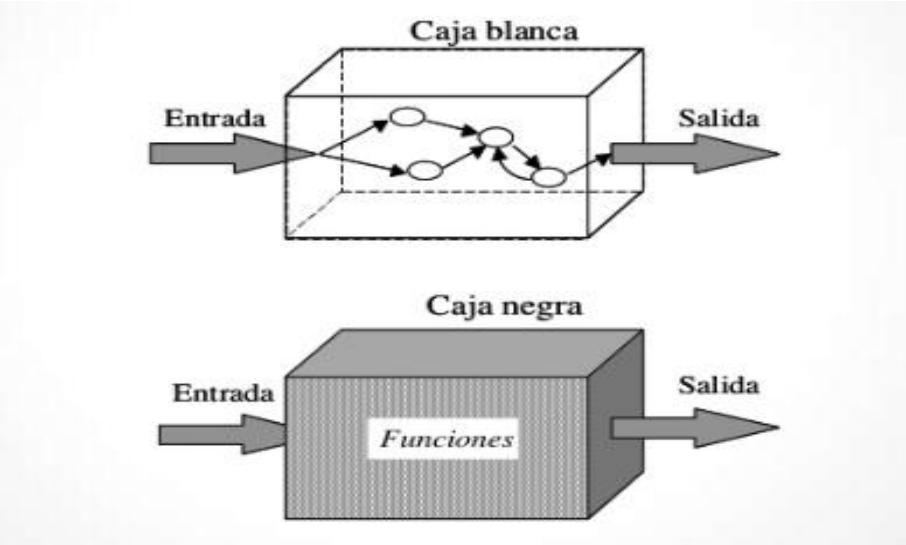


Figura 1. Representación de pruebas de Caja Blanca y Caja Negra

Fuente: BARBADO GUTIÈRREZ, Iván. Representación de pruebas de Caja Blanca y Caja Negra. [Imagen]**.** TPV Web: Plataforma web para la gestión de productos y facturación de una librería/papelería. Universidad de Valladolid.2016. p.365. [Consultado: 18 de marzo de 2019]. Disponible en Internet: https://uvadoc.uva.es/bitstream/10324/18632/1/TFG-B.944.pdf

4.4 PRUEBAS FUNCIONALES

Tomando la definición dada por Sánchez en el 2015, las pruebas funcionales se presentan como:

> Este tipo de pruebas se basan en las funcionalidades de un sistema que se describen en la especificación de requisitos, es decir, lo que hace el sistema. También pueden no estar documentadas, pero se requiere un nivel de experiencia elevado para interpretar estas pruebas. Estas prueban suelen estar asociadas a las técnicas de diseño de pruebas de caja negra, ya que tienen en cuenta el comportamiento externo del *software*.[15]

[14] TERRERA, Gustavo. Testing de caja blanca–parte I [En Línea]. TestingBaires.3 de Junio del 2014, p.1. [Consultado: 20 de Diciembre de 2018]. Disponible en Internet: https://testingbaires.com/2014/01/03/testing-de-caja-blanca-parte/

[15] SÁNCHEZ PEÑO, Pruebas de Software fundamentos y técnicas, Op. Cit., p. 23.

Figura 2. Esquema Pruebas Funcionales

Fuente: LOPÈZ CEBALLOS,Alfonso Gabriel.Pruebas Funcionales. [Imagen]**.** 2012. [Consultado: 18 de marzo de 2019]. Disponible en Internet: https://es.slideshare.net/Alfonsolopezceballos/pruebas-funcionales

4.5 PRUEBAS NO FUNCIONALES

La siguiente afirmación fue propuesta por Ocampo y Correa en el año 2011 acerca de las pruebas no funcionales resaltando que:

"Estas se realizan para verificar que el *software* desarrollado cumple con los requerimientos no funcionales establecidos por el cliente. Existen varios tipos de pruebas no funcionales, entre las más comunes están las pruebas de seguridad, pruebas de rendimiento, pruebas de usabilidad, pruebas de portabilidad, entre otras".[16]

[16] OCAMPO ACOSTA, Alejandro y CORREA TAPASCO, Luisa Marcela. Impacto de las pruebas no funcionales en la medición de la calidad del producto software desarrollado [En Línea]. Tesis de Ingeniería de sistemas y computación. Pereira, Colombia. Universidad tecnológica de Pereira. Facultad de ingenierías eléctrica, electrónica, física y Ciencias de la computación. Programa de ingeniería de sistemas y computación 2011. p.16. [Consultado: 21 de diciembre de 2018]. Disponible en Internet: https://n9.cl/cVGb

4.5.1 PRUEBAS DE CARGA

Tal como se presentan en el informe oficial en su página *web* sobre pruebas de carga de *MICROFOCUS* son definidas como aquellas que "permiten identificar y diagnosticar los problemas que afectan al rendimiento, la capacidad de ampliación y la fiabilidad de una aplicación".[17] Así mismo, dentro de este documento se menciona que: "hay muchos tipos de pruebas de carga, y cada uno de ellos aborda un área problemática específica"[18]

4.5.2 PRUEBAS DE ESTRÉS

Para una definición práctica sobre este tipo de pruebas, se toma como referencia la publicada en el blog de 4R Soluciones, donde las mismas se establecen como aquella que "pone a prueba la robustez y la confiabilidad del *software* sometiéndolo a condiciones de uso extremas. Entre estas condiciones se incluyen el envío excesivo de peticiones y la ejecución en condiciones de hardware limitadas. El objetivo es saturar el programa hasta un punto de quiebre donde aparezcan *bugs* (defectos) potencialmente peligrosos"[19].

Otra definición referencia para estas pruebas es la siguiente: "se enfrenta el sistema a situaciones anormales, es decir, enfrentarlo a peticiones de usuario con una frecuencia definida, para determinar su uso de recursos en dichas circunstancias y su reacción ante la misma. Busca sobrecargar el sistema, para hallar estados de inestabilidad o procesamiento inapropiado"[20]. Publicada en la tesis sobre el Impacto de las pruebas no funcionales en la medición de la calidad del producto *software* desarrollado, elaborada por Ocampo y Correa en el año 2011.

---

[17] MICROFOCUS. Informe oficial pruebas de carga la clave para garantizar la ejecución ininterrumpida de las aplicaciones empresariales [En Línea].Microfocus.Colombia. (02 de febrero de 2017), p.3. [Consultado: 13 de Enero de 2019]. Disponible en Internet: https://www.microfocus.com/es-es/media/white-paper/load_testing_wp_es.pdf

[18] Ibid., p. 3

[19] 4R SOLUCIONES. Pruebas de stress sobre aplicaciones web [En Línea].4r Productora Digital y Software Factory. Argentina 03 de Octubre de 2012. p.1. [Consultado: 13 de Enero de 2019]. Disponible en : http://www.4rsoluciones.com/blog/pruebas-de-stress-sobre-aplicaciones-web-2/

[20] OCAMPO ACOSTA, Alejandro y CORREA TAPASCO, Luisa Marcela. Impacto de las pruebas no funcionales en la medición de la calidad del producto software desarrollado, Op. Cit., p. 47.

4.6 CASOS DE USO

Tal como se define en el artículo: Introducción al modelado de sistemas de software usando el Lenguaje Unificado de Modelado (UML) “el modelo de casos de uso describe la funcionalidad propuesta del nuevo sistema. Un caso de uso representa una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema. Un caso de uso es una unidad de trabajo significativo; por ejemplo, crear una solicitud y modificar una solicitud son todos casos de uso”[21]

4.7 CONCURRENCIA

Por otra parte, en su presentación sobre: Introducción a la Programación Concurrente, Rodríguez establece que: “Se habla de concurrencia cuando ocurren varios sucesos de manera contemporánea. En base a esto, la concurrencia en computación está asociada a la “ejecución” de varios procesos que coexisten temporalmente”[22]

4.8 METODOLOGÍA EN CASCADA

“La metodología en cascada parte de los principios definidos para el modelo de desarrollo de software en cascada, también conocido como ciclo de vida clásico o modelo convencional, donde según Pressman (2005), dicho modelo sugiere un enfoque sistemático y secuencial hacia el desarrollo de software que indica ciertas etapas y actividades.

En este sentido, el modelo mismo se ha constituido en una metodología que ordena rigurosamente todas las etapas del ciclo de vida del software, implicando en sí misma un desarrollo de proyecto rígido y lineal.

---

[21] SPARKS, Geoffrey. El Modelo de Casos de Uso. Introducción al modelado de sistemas de software usando el Lenguaje Unificado de Modelado (UML) [En Línea]. Solus - Craftware Consultores Ltda. p.3[Consultado: 12 de Marzo de 2019]. Disponible en internet: http://www.sparxsystems.com.ar/downloads/whitepapers/El_Modelo_de_Casos_de_Uso.pdf

[22] RODRÌGUEZ FLORIDO, Miguel Ángel. Introducción a la Programación Concurrente. [En Línea]. Universidad de Las Palmas de Gran Canaria. p.4. [Consultado: 12 de Marzo de 2019]. Disponible en internet: https://www2.ulpgc.es/hege/almacen/download/20/20233/tema1.pdf

Esta es la metodología clásica dentro del contexto del desarrollo de software, que busca producir un software robusto y estrictamente documentado en cada una de sus etapas, donde a la finalización de cada una permite el inicio de la siguiente, tomando como parte del insumo los datos producidos en la anterior.

Así, según Sommerville (2005), las etapas definidas para el modelo se transforman en actividades fundamentales de desarrollo:

- "Análisis y definición de requerimientos. Los servicios, restricciones y metas del sistema se definen a partir de las consultas con los usuarios. Entonces se definen en detalle y sirven como una especificación del sistema.

- Diseño del sistema y del software. El proceso de diseño del sistema divide los requerimientos en sistemas hardware y software. Establece una arquitectura completa del sistema. El diseño del software identifica y describe las abstracciones fundamentales del sistema software y sus relaciones.

- Implementación y prueba de unidades. Durante esta etapa, el diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla con su especificación.

- Integración y prueba del sistema. Los programas o las unidades individuales de programas se integran y prueban como un sistema completo para asegurar que se cumplan los requerimientos del software. Después de las pruebas, el sistema software se entrega al cliente.

- Funcionamiento y mantenimiento. Por lo general (aunque no necesariamente), ésta es la fase más larga del ciclo de vida. El sistema se instala y se pone en funcionamiento práctico. El mantenimiento implica corregir errores no descubiertos en las etapas anteriores del ciclo de vida, mejorar la implementación de las unidades del sistema y resaltar los servicios del sistema una vez que se descubren nuevos requerimientos"[23]

[23] Disponible en Internet: https://hablemosdesoftware.wordpress.com/2014/10/03/metodologias-de-desarrollo-de-software/

4.9 BASES DE DATOS

"Una base de datos es una colección de datos almacenados y organizados de forma que un programa del ordenador pueda seleccionarlos rápidamente y capaces de ser: recobrados, actualizados, insertados y borrados. En un DBMS una base de datos es un sistema de archivos electrónico."[24]

4.10 CICLO DE VIDA DEL PRODUCTO

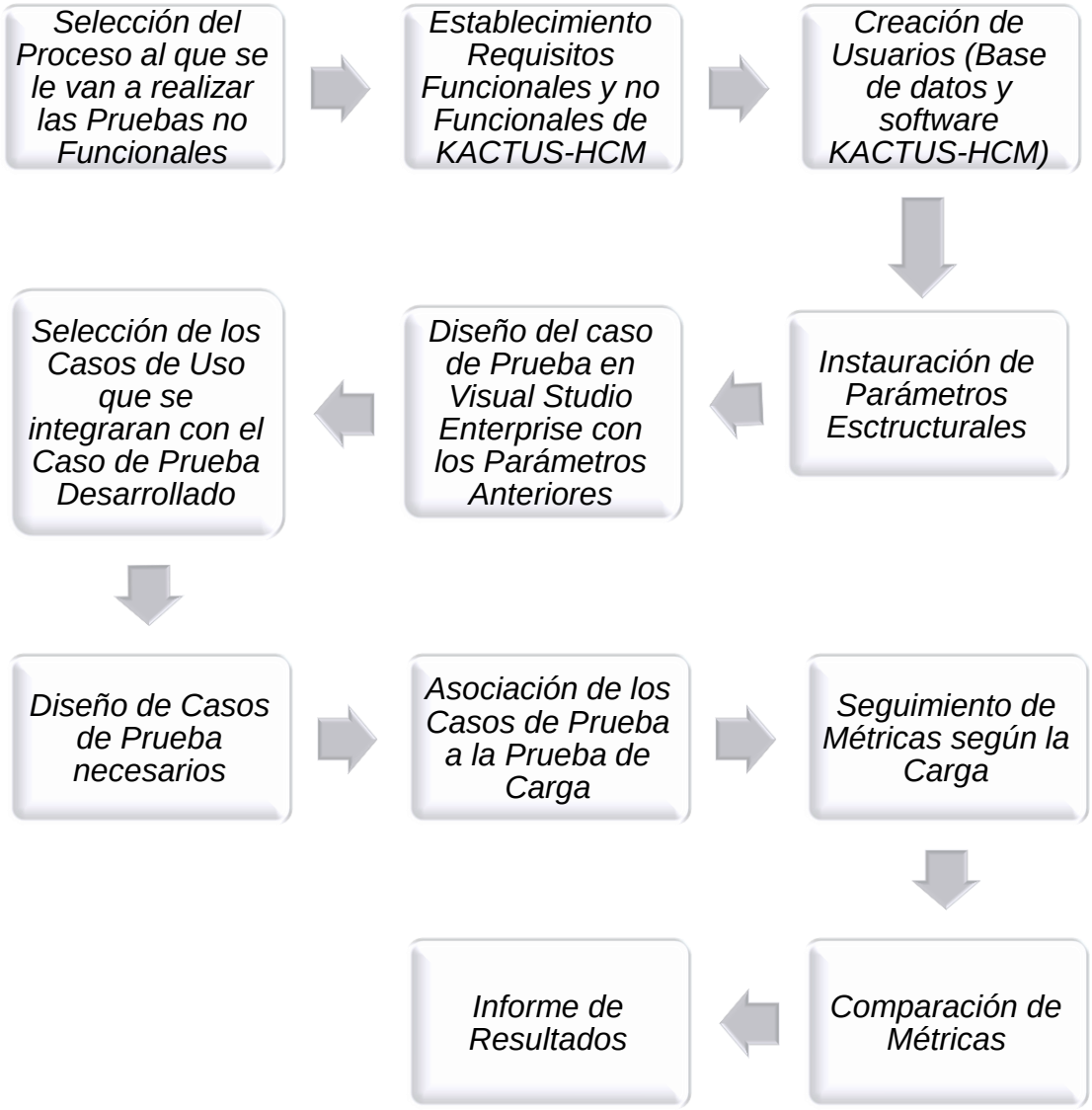
"El ciclo de vida de un proyecto se software, empieza cuando se da la recolección de requerimientos para el programa a desarrollar y termina cuando el producto ha quedado completado y es entregado al cliente que lo pidió. Sin embargo en el intermedio, hay una gran cantidad de fases por las cuales se tiene que pasar y cada metodología tiene fases distintas en su ciclo de desarrollo de programas"[25]

5. METODOLOGÍA

5.1 MANUAL METODOLÓGICO PARA LA EJECUCIÓN DE PRUEBAS NO FUNCIONALES

[24] GUTIÉRREZ DÍAZ, Alejandro. Base de datos. [En línea]. Centro Cultural Itaca S.C.2016.Distrito Federal México. p. 10. MIS 308. [Consultado: 18 de Marzo de 2019]. Disponible en Internet : https://www.aiu.edu/cursos/base%20de%20datos/pdf%20leccion%201/lecci%C3%B3n%201.pdf
[25] Disponible en : https://okhosting.com/blog/el-ciclo-de-vida-del-software/

A continuación, se encontrará un diagrama con de cada uno de los pasos y funciones llevados a cabo en la ejecución de pruebas no funcionales en el software KACTUS-HCM. Es importante resaltar que el mismo puede ser tomando como base para los fines pertinentes de aquellos casos donde se apliquen este



tipo de pruebas.

Figura 3. Diagrama del manual metodológico para la ejecución de pruebas no funcionales

Estos pasos se describen detalladamente a continuación:

5.1.1 SELECCIÓN DEL PROCESO AL QUE SE LE VAN A REALIZAR LAS PRUEBAS NO FUNCIONALES

Inicialmente se debe elegir el programa al que se le van a aplicar las pruebas. Por medio de manuales conocer su funcionamiento y la estructura que le permite ejecutarse. A través de SQL Profiler se puede obtener la consulta realizada, para que finalmente esta pueda ser ingresada al caso de prueba.

5.1.2 REQUISITOS FUNCIONALES Y NO FUNCIONALES DE KACTUS-HCM

Para este paso, los requisitos funcionales y no funcionales del software KACTUS-HCM son manejados de la siguiente manera:

5.1.2.1 REQUISITOS FUNCIONALES

En este apartado se deben especificar los requisitos que deben cumplir los datos de entrada para que la funcionalidad del programa sea correcta. Estos requisitos pueden variar dependiendo del programa, por esto en el apartado 5.7.2.1 se especifican cuales son para el programa KNmLinom.

5.1.2.2 REQUISITOS NO FUNCIONALES

Mientras que para los requisitos no funcionales se debe especificar que se espera del programa entorno a las características de calidad que son: rendimiento, usabilidad, escalabilidad, seguridad y estabilidad. Como ejemplo se puede tomar los establecidos para KNmLinom en el apartado 5.7.2.2

5.1.3 CREACIÓN DE USUARIOS PARA LA BASE DE DATOS Y PARA EL SOFTWARE KACTUS-HCM

Una vez ejecutado el paso anterior, se da inicio a la creación de usuarios tanto para la base de datos como para el software KACTUS-HCM, resaltando que:

5.1.3.1 BASE DE DATOS

La base de datos debe contar con las últimas versiones y con los scripts correspondientes para que el programa o proceso que se vaya a ejecutar no presente inconvenientes al iniciar.

5.1.3.2 USUARIOS PARA BASE DE DATOS

De acuerdo a los casos de uso que se deseen integrar con el caso de prueba se debe crear un usuario que tenga acceso a la base de datos.

5.1.3.3 USUARIOS PARA KACTUS-HCM

De este modo, se deben crear los usuarios (uno por cada caso de uso) en el programa KACTUS-HCM con un usuario administrador en el programa ***KGnUsuar***; estos usuarios deben ser asociados a la empresa o empresas en las que se va a ejecutar la prueba en el programa ***KGnEmpre.***

Asignando los permisos correspondientes para el proceso a los usuarios en el programa ***KGnPerma***, si se trata de una cantidad de usuarios puede ser tedioso la asignación de permisos uno a uno, para este proceso se puede hacer uso del programa ***KGnCpusu*** que permite copiar los permisos asignados a un usuario y asignárselos a un grupo seleccionado.

Estos usuarios también deben asociarse al archivo Conexiones.ini contenido en la carpeta versión en la ruta C:\Kactus\, configurando la siguiente estructura:

[Alias]

DriverID=Motor base de datos (MSSQL u ORA)

DataBase=Nombre de la base datos a la que se direcciona

User_Name=Usuario con el que se va a ingresar al menú Ophelia

Password= Clave generada en el campo pas_usua en la tabla Gn_Usuar (Encriptada por el programa KGnUsuar)

Server=Servidor donde se encuentra la base de datos.

5.1.3.4 VERIFICAR LA CONEXIÓN Y PERMISOS DE LOS USUARIOS CREADOS

Y finalmente para este apartado se debe ingresar a la base de datos con los usuarios creados. Para realizar la verificación de los usuarios establecidos para KACTUS-HCM se ingresa al menú web del mismo (***Ophelia***) donde este permite acceder a los programas y se puede comprobar que los usuarios han quedado con ingreso al software y con los permisos otorgados a los programas.

5.1.4 INSTAURACIÓN DE PARÁMETROS ESTRUCTURALES

Durante este paso se deben establecer los parámetros estructurales que son necesarios para el inicio de sesión y la ejecución del proceso; estos tienen una estructura igual para realizar el Login pero para la ejecución del proceso esta estructura varia.

5.1.4.1 ESTRUCTURA PARA REALIZAR EL LOGIN

Pdata = new string[9];

Pdata[0] = Alias del archivo conexiones.ini

Pdata[1] = Usuario

Pdata[2] =Contraseña cifrada

Pdata[3] = Tipo de Usuario

Pdata[4] = Empresa asociada

Pdata[5] = Código de programa

Pdata[6] =IPcliente

Pdata[7] = "TCP"; Protocolo

Pdata[8] = "N";

5.1.4.2 ESTRUCTURA DEL PROCESO

Debido a que los programas de KACTUS-HCM varían en sus procesos no todos tienen la misma estructura, por esto debe ser solicitada a los Ingenieros de desarrollo ya que esta se encuentra en el código fuente y solo ellos tienen el acceso.

5.1.5 DISEÑO DEL CASO DE PRUEBA EN VISUAL STUDIO ENTERPRISE CON LOS PARÁMETROS ESTRUCTURALES

En este paso, en el proyecto ***KactusLoadTest*** se debe configurar la estructura del Login mencionada en el apartado 5.1.4.1 (el alias es el usuario para acceder a la base de datos) con los datos correspondientes que se han creado en el apartado 5.1.3.3 (Usuario, contraseña, tipo de usuario y empresa asociada), seguidamente se debe colocar la estructura del programa al que se le realizara la prueba de carga y de igual manera configurar los campos de la misma, para observar la configuración de un programa en específico se puede tomar como ejemplo en el apartado 5.7.4.2 la del programa KNmLinom.

5.1.6 SELECCIÓN DE LOS CASOS DE USO QUE SE INTEGRARAN CON EL CASO DE PRUEBA DESARROLLADO

Seguidamente durante este paso, dependiendo del proceso se deben realizar consultas diferentes para obtener un resultado; estas consultas pueden variar con la cantidad de empleados, el tipo de empleado y los números de contrato, entre otros. Como se mencionó anteriormente se puede rescatar esta consulta con la función de SQL Profiler al realizarla por interfaz de usuario.

5.1.7 DISEÑO DE LOS CASOS DE PRUEBA NECESARIOS

Al llevar a cabo el paso anterior se procede a replicar el caso de prueba desarrollado con diferentes casos de uso, definido en el marco teórico como una unidad discreta de interacción entre un usuario (humano o máquina) y el sistema, dependiendo la carga de usuarios a la que será sometido el proceso. Recalcando que por cada usuario se debe haber un caso de prueba.

5.1.8 ASOCIACIÓN DE LOS CASOS DE PRUEBA A LA PRUEBA DE CARGA

Para este paso en la prueba de carga se deberán asociar los casos de prueba con los que se desea aplicar la prueba de carga. Esto se configura en el apartado Escenarios>Combinación de pruebas> Click derecho > Agregar pruebas. Se pueden agregar todos los casos de prueba de una vez o se pueden ir agregando progresivamente.

5.1.9 SEGUIMIENTO DE MÉTRICAS SEGÚN LA CARGA

Seguidamente se procede al paso donde se debe hacer el seguimiento de las métricas a medida que se aumente en la prueba de carga el número de casos de pruebas asociados. Como se menciona en el apartado anterior se pueden ir ingresando una cantidad de pruebas tomar métricas y aumentar o disminuir la cantidad y tomarlas nuevamente.

5.1.10 COMPARACIÓN DE MÉTRICAS

Una vez desarrollado este seguimiento anterior se empiezan a comparar las métricas de diferentes cargas aplicadas, desde la interfaz de usuario (Tiempos de respuesta, consumo de memoria y procesador).

5.1.11 REALIZAR UN INFORME

Finalmente, en esta secuencia de pasos se realiza un informe, dando a conocer los resultados obtenidos que contengan observaciones y sugerencias de los mismos. Este informe también debe incluir la descripción del entorno donde fueron aplicadas las pruebas como las Características específicas del equipo.

5.2 COMPLEMENTOS MANUAL METODOLÓGICO

5.2.3 DESCRIPCIÓN DEL ENTORNO

Para la ejecución de estas pruebas se contó con un servidor de pruebas asignado por DIGITAL WARE con las siguientes características:

5.2.3.1 SOFTWARE EQUIPO

- Sistema Operativo: Windows Server 2012 R2 Standard x64
- Microsoft SQL Server Management Studio 2017
- Ophelia Menu Ver: 18.11.1.1
- Liquidador de Nómina Ver: 19.0.5.1

5.2.3.2 HARDWARE EQUIPO

- 2 Virtual CPU 1.9 GHz
- Memoria RAM 8GB
- Disco C 89.6 GB
- Disco D 49.9 GB
- Tarjeta de red 10 Gbps

5.2.3.3 SOFTWARE UTILIZADO

- Visual Studio 2017 Entreprise Ver: 15.9.7

5.3 MÉTODOS UTILIZADOS

5.3.3 PRUEBAS DE UNIDAD

Tal como lo define en su Hermann las pruebas de unidad son “un procedimiento usado para validar que un módulo o método de un objeto fuente funciona apropiadamente y en forma independiente. A través de ellas se verifica que cierto módulo o método se ejecuta dentro de los parámetros y especificaciones concretadas en documentos tales como los casos de uso Permiten detectar efectivamente la inyección de defectos durante fases sucesivas de desarrollo o

mantenimiento."[26] De este modo, estas pruebas de unidad fueron utilizadas en la verificación de cada prueba por usuario, para constatar que cada método estuviese funcionando y al momento de fusionarlos no presentasen fallas.

5.3.4 PRUEBAS DE INTEGRACIÓN

Calád y Ruiz definen las pruebas de integración de la siguiente manera:

> Las pruebas de integración se realizan para asegurar que varios componentes del sistema interactúen y se comuniquen correctamente. Las pruebas pueden realizarse en varios niveles. En el nivel más bajo, generalmente se realizan por el equipo de desarrollo para asegurar que las unidades juntas funcionen correctamente. En niveles más altos, las pruebas pueden ser hechas por el equipo de pruebas o de desarrollo, y examina como las piezas del sistema funcionan conjuntamente [27]

Por tanto, en este proyecto se hizo uso de ellas para integrar todas las pruebas de unidad y obtener una ejecución exitosa.

5.3.5 PRUEBAS DE DESEMPEÑO

En estas pruebas se categorizaron las pruebas de carga realizadas para descubrir falencias en el sistema y/o infraestructura. Tal como lo definen Calád y Ruiz en su proyecto de metodología de testing de software:

> "Las pruebas de desempeño son usadas para descubrir problemas que pueden resultar debido a factores como: falta de recursos en el servidor, ancho de banda inapropiados, inadecuada capacidad de las

[26] HERMANN, Eliza. Pruebas de Software: Herramientas: Pruebas Unitarias. [Presentación PowerPoint].Madrid, España.Okybele.2011, p. 4. Disponible en Internet : http://www.kybele.etsii.urjc.es/docencia/IS_LADE/2011-2012/Material/Pruebas%20de%20SoftwareHerraminetas.pdf

[27] CALÁD ÁLVAREZ, Alejandro y RUÍZ CALLE, Juan David. Metodologías de testing de software y su aplicación en el centro de informática de la Universidad EAFIT [en línea]. Proyecto de grado Ingeniería de Sistemas. Medellín, Colombia. Universidad EAFIT. Departamento de Ingeniería de Sistemas, 2009. p. 48 [Consultado: 30 de Diciembre de 2018].Disponible en Internet: https://repository.eafit.edu.co/xmlui/bitstream/handle/10784/2744/RuizCalle_JuanDavid_2009.pdf?sequence=1

bases de datos, debilidades del sistema operativo, entre muchos otros factores que pueden llevar a disminuir el desempeño del servidor"[28]

5.4 HERRAMIENTAS UTILIZADAS

5.4.1 VISUAL STUDIO ENTERPRISE 2017

Teniendo en cuenta que "Las herramientas de prueba de Visual Studio pueden ayudarlo a usted y a su equipo a desarrollar y mantener altos estándares de excelencia de código"[29] . Este programa facilitó la creación de la prueba de carga, que contenía las pruebas de unidad e integración. En esta herramienta se pudo visualizar y analizar con diferentes escenarios como se comportaba el proceso de la liquidación de nómina KNmLinom.

5.4.2 KACTUS-HCM

"KACTUS-HCM es la solución necesaria para realizar una administración estratégica de las áreas de gestión humana. Permite despreocuparse de la operatividad del departamento e ir más allá, proporcionando herramientas para realizar estadísticas, gráficas sobre puntos clave y manejar indicadores de gestión"[30] se debe tener en cuenta que este software permitió realizar todo lo relacionado con el diseño de casos de uso para las pruebas no funcionales.

El programa liquidador de nómina KNmLinom del software KACTUS-HCM pertenece al módulo de nómina y este permite realizar la liquidación del empleado recolectando información de otros programas; del programa contratos (KNmContr) se obtiene el tipo de contrato, la periodicidad de pago, el tipo de salario, el sueldo básico, tipo de cotizante, entre otros. Para completar el proceso de nómina este

[28] Ibip., p. 70.

[29] Microsoft. Herramientas de prueba en Visual Studio [En Línea].Estados Unidos.(15 de Marzo de 2018), párr.1. [Consultado 30 de Diciembre de 2018].Disponible en Internet : https://docs.microsoft.com/en-us/visualstudio/test/improve-code-quality?view=vs-2017

[30] Digital Ware Tecnology that changes People´s Lives. [En línea].Bogotá DC. [Consultado: 10 de Enero de 2019]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

programa además revisa el programa novedades (KNmNoved) donde quedan los registros de incapacidades, embargos, días no trabajados, permisos no remunerados, préstamos, y otras novedades que afecten el pago de nómina. El programa KNmLinom permite realizar la liquidación con una fecha de corte y una fecha de pago, así como también se puede establecer por un filtro el tipo de empleados que se desea liquidar; este programa inserta en otros programas las cuales son:

- Prenómina: En este programa se inserta el pago y descuentos realizados para el desprendible de nómina.
- Novedades de seguridad social: Aquí se insertan los pagos realizados la cantidad de días normales (sin novedades), incapacidades (General, hospitalaria, accidente de trabajo, licencia de maternidad, licencia de paternidad, permiso remunerado, entre otros).
- Bases de retención en la fuente: En este se inserta la base de retención que se va a aplicar para cada empleado liquidado y este los clasifica en los diferentes porcentajes que tienen para el pago de retención en la fuente.

5.4.3 SQL SERVER MANAGEMENT STUDIO 2017

SQL Server Management Studio se define como:

> SQL Server Management Studio (SSMS) es un entorno integrado para administrar cualquier infraestructura SQL. Use SSMS para acceder, configurar, administrar, administrar y desarrollar todos los componentes de SQL Server, Azure SQL Database y SQL Data Warehouse. SSMS ofrece una única utilidad integral que combina un amplio grupo de herramientas gráficas con varios editores de scripts enriquecidos para proporcionar acceso a SQL Server para desarrolladores y administradores de bases de datos de todos los niveles.[31]

Esta herramienta fue utilizada para hacer el seguimiento por tablas a los diferentes programas en los que debía insertar ***KNmLinom*** y observar del mismo modo los inicios de sesión y errores controlados del proceso.

[31] Microsoft. SQL Server Management Studio (SSMS) [En Línea]. Estados Unidos. (15 de Octubre de 2018), párr.1 [Consultado 10 de Enero de 2019].Disponible en Internet : https://docs.microsoft.com/en-us/visualstudio/test/improve-code-quality?view=vs-2017

5.5 DESCRIPCIÓN DE PROGRAMAS

5.5.1 KGNUSUAR

Este programa es de uso exclusivo del Administrador del Sistema DBA, y en él, se crean los Usuarios que tienen acceso al aplicativo. Se pueden crear 3 tipos de usuarios:

- ✓ Auditor: Se encarga de la gestión de control y tiene acceso a todo el sistema, pero únicamente para verificación de movimientos en el sistema, ingresos, barrado de datos y generación de reportes de auditoria.
- ✓ D.B.A.: Es aquel que administra todos los recursos del sistema Kactus HCM.
- ✓ Común: Es aquel que puede realizar algunas actividades tales como adición, modificación, consultas y reportes de datos, restringido por el perfil de usuario y definido en el programa de permisos KGnPerma de este mismo módulo.

5.5.2 KGNUEMPR

Este programa permite asignar al usuario el uso del aplicativo (Sistema Kactus HCM) para una o varias empresas de la Organización (Entidades que tienen parametrizadas más de una empresa).

5.5.3 KGNPERMA

Permite realizar la asignación de uso autorizado de los programas a cada usuario, con respeto a las tareas específicas que cumple. Se selecciona de una lista de programas no asociados, el programa que se desea asignar y se relaciona con los permisos del programa (Añadir, modificar, eliminar, entre otros), es decir, muestra una relación completa de todos los programas a los cuales tiene acceso un usuario, y además, el tipo de acceso (permisos asignados)

5.5.4 KGNCPUSU

Este programa permite copiar los permisos establecidos para un usuario en el programa KGnPerma a un grupo de usuarios seleccionados. De este modo el grupo de usuarios tendrá los mismos programas asociados y los mismos permisos.

5.6 CALIDAD

Teniendo en cuenta que los modelos de calidad del software contemplan:

CALIDAD DEL PRODUCTO	Propiedades del producto según Usuario y desarrollador.	**VALOR COMERCIAL**
CALIDAD DEL PROCESO	Actividades que influyen en la calidad del producto	
CALIDAD EN USO	Relación del producto con el ambiente donde se emplea	**VALOR TÉCNICO**

Tabla 2. Modelos de Calidad del Software

Fuente: Elaborada por el Autor.

“En la industria del software se pueden evidenciar necesidades de satisfacción del cliente de productos o servicios de software, de reducción de recursos invertidos en proyectos de software y de la efectiva asignación de recursos humanos. Si hablamos de la calidad del software, una de las primeras definiciones aseguraba que “la calidad de un programa o sistema se evaluaba de acuerdo al número de

defectos por cada mil líneas de código. (KLOC: Kilo Lines Of Code)"[32] . "Es por ello que se hace necesaria la adopción de un estándar de calidad, basado en la experiencia de otras industrias con más tiempo de madurez, aunque basado en las características particulares del software como producto y como servicio".[33]

5.6.1 ISO/IEC 25010

El modelo de calidad representa la piedra angular en torno a la cual se establece el sistema para la evaluación de la calidad del producto. En este modelo se determinan las características de calidad que se van a tener en cuenta a la hora de evaluar las propiedades de un producto software determinado. La calidad del producto software se puede interpretar como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera un valor. Son precisamente estos requisitos (funcionalidad, rendimiento, seguridad, mantenibilidad, etc.) los que se encuentran representados en el modelo de calidad, el cual categoriza la calidad del producto en características y subcaracterísticas.[34]

5.7 MANUAL METODOLÓGICO APLICADO AL PROCESO KNMLINOM

5.7.1 SELECCIÓN DEL PROCESO AL QUE SE LE VAN A REALIZAR LAS PRUEBAS NO FUNCIONALES

Este diseño metodológico fue aplicado priorizando los procesos críticos de ***KACTUS-HCM***, en el cual lidera el proceso Liquidador de Nómina (KNmLinom). En este sentido las pruebas no funcionales se realizaron para conocer la carga que soporta este proceso, certificar que se realice la inserción de datos en los diferentes programas e identificar los factores que afectan el rendimiento del proceso.

[32] Pressman, Roger, Ingeniería de Software 3ª Ed., McGraw Hill, 1993.
[33] LÓPEZ ECHEVERRY Ana María, CABRERA Cesar y VALENCIA AYALA, Luz Estela. Introducción a la calidad de software [En Línea].(Septiembre) ,2008, no.39, p. 328.ISSN 0122-1701 [Consultado : 10 de Enero de 2019]. Disponible en Internet: file:///E:/Dialnet-IntroduccionALaCalidadDeSoftware-4745899.pdf

[34] ISO25000 [En Línea]. [Consultado: 12 de enero de 2019]. Disponible en Internet : https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&start=3

5.7.2 ESTABLECIMIENTO REQUISITOS FUNCIONALES Y NO FUNCIONALES DE KNMLINOM

Partiendo de la idea de que el requisito es una cualidad o característica para que un proceso funcione de una manera adecuada, se toma como referencia a Maximiliano Cristiá en “Introducción a la Ingeniería de Requerimientos”[35] donde expone algunos ejemplos de cómo deben ser estos requerimientos. A continuación, se especifican los requisitos que permiten a KNmLinom ejecutarse

5.7.2.1 REQUISITOS FUNCIONALES

- Los empleados a liquidar deben contar con contratos activos
- Los empleados no deben contar con una liquidación en el mismo rango de fechas con el que se va a liquidar
- Los empleados a liquidar deben ser diferentes para cada usuario que ejecute la liquidación
- Para poder realizar el proceso de liquidación al mismo tiempo los usuarios que ejecuten este proceso deben ser diferentes
- El proceso debe insertar en las siguientes tablas: Prenómina (NM_PRENO), Novedades de seguridad social (NM_NOVSS) y bases de retención en la fuente (NM_BASRE)

5.7.2.2 REQUISITOS NO FUNCIONALES

- El proceso debe ejecutarse con la llave primaria y foránea de la tabla de novedades de seguridad social NM_NOVSS desactivadas
- El proceso no debe demorar más de 3 minutos ejecutándose
- El proceso debe soportar la concurrencia de 30 usuarios
- KNmLinom debe realizar transacciones de todos los usuarios al mismo tiempo insertando los datos en las tablas mencionadas anteriormente.

[35] CRISTIÁ, Maximiliano.Introducción a la Ingenieria de Requerimientos. [En Línea]. Ingeniería de Software Facultad de Ciencias Exactas, Ingeniería y Agrimensura. Universidad Nacional de Rosario.2011. p.4-5. [Consultado: 20 de marzo de 2019]. Disponible en Internet:https://www.fceia.unr.edu.ar/~mcristia/publicaciones/ingreq-a.pdf

5.7.3 CREACIÓN DE USUARIOS (BASE DE DATOS Y SOFTWARE KACTUS-HCM)

5.7.3.1 BASE DE DATOS

La asignación de esta base de datos fue realizada por la Ingeniera encargada del dimensionamiento Andrea Otálora. Esta base de datos se encuentra ubicada en el servidor SQL2014\CLIENTESDW denominada SVPK; la misma fue adecuada con las últimas versiones de los programas involucrados para garantizar su correcto funcionamiento.

5.7.3.2 USUARIOS PARA BASE DE DATOS

En el programa Microsoft SQL Server Management Studio, se realizó la conexión al servidor donde se encontraba la base de datos con autenticación de SQL Server con un usuario administrador. De allí se localizó el apartado seguridad en donde el subapartado inicio sesión permite agregar nuevos inicios de sesión (Ver Figura 4), con los cuales se pudieron ingresar y modificar datos en las diferentes tablas de SVPK. Se debe tener en cuenta que en las propiedades de inicio de sesión de cada usuario se debía realizar la asignación de usuarios con la base de datos. (Ver Figura 5)

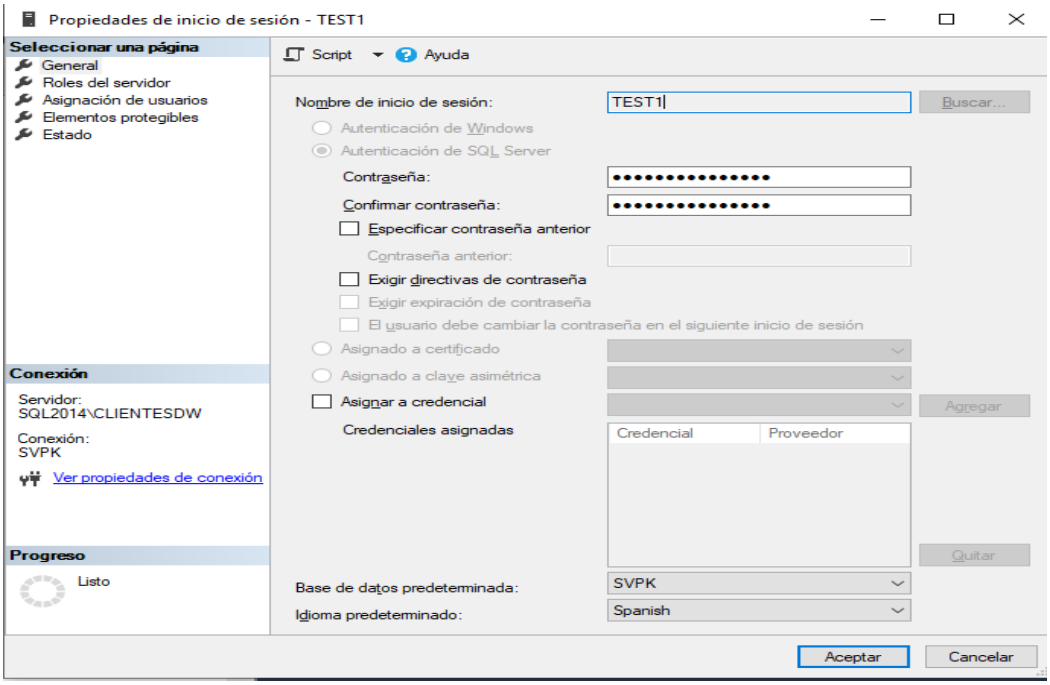


Figura 4. Propiedades de inicio de sesión

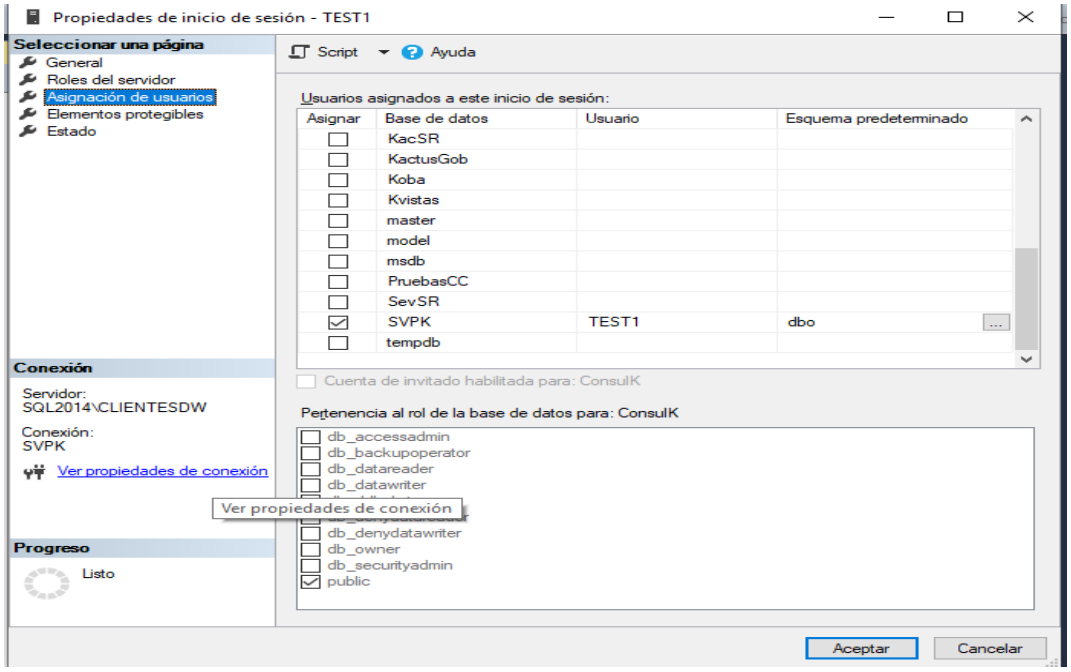


Figura 5. Asignación de usuarios con la base de datos

5.7.3.3 USUARIOS PARA KACTUS-HCM

Esta asignación inició con el ingreso al menú web de KACTUS-HCM (Ophelia) con un usuario existente que contaba con los permisos suficientes para la creación de los 30 usuarios prueba. Seguidamente se crearon los nuevos usuarios en el programa KGnUsuar *(Ver Figura 6),* asignándole a estos la misma clave, para la facilidad de la estructura de las pruebas; incluyendo en los mismos la configuración de tipo de usuario común. Del mismo modo, se asignó la empresa en el programa KGnUempr a los 30 usuarios *(Ver Figura 7)* y en el programa KGnPerma se asociaron todos los programas con permisos a un usuario prueba, para que después se pudiesen copiar estos permisos a los usuarios restantes en el programa KGnCpusu. (*Ver Figura 8).* Por otra parte, se configuró el archivo conexiones.ini como se puede observar en la figura 9 contenido en la carpeta versión en la ruta C:\Kactus\, configurando la siguiente estructura:

[TEST1]

DriverID=MSSQL

DataBase=SVPK

User_Name=TEST1

Password=`MROJJMGGLDDHAFD>C?

Server=SQL2014\CLIENTESDW

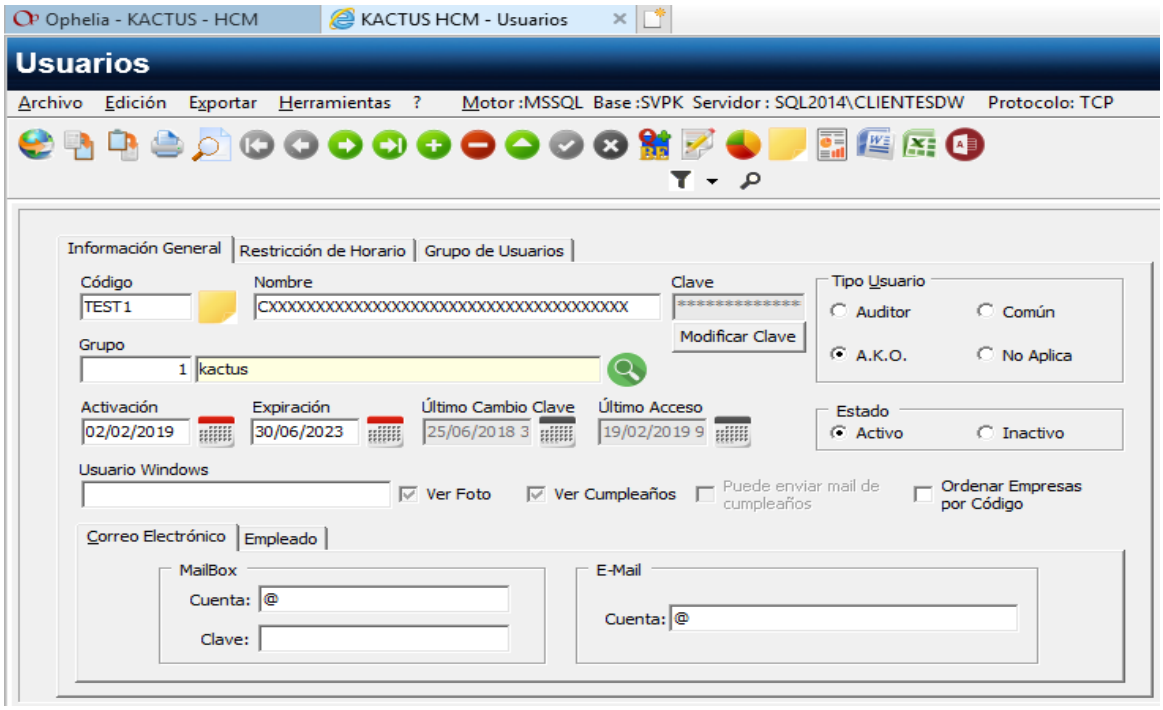


Figura 6. Creación de usuarios KGnUsuar

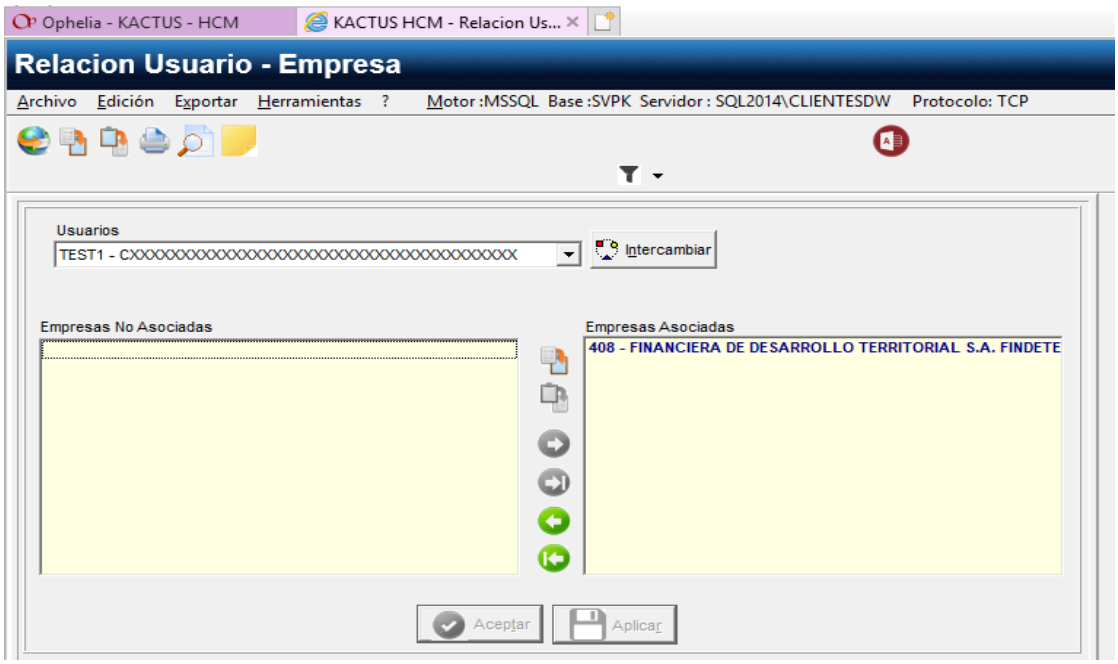


Figura 7. Asignación de empresa a usuarios KGnUempr

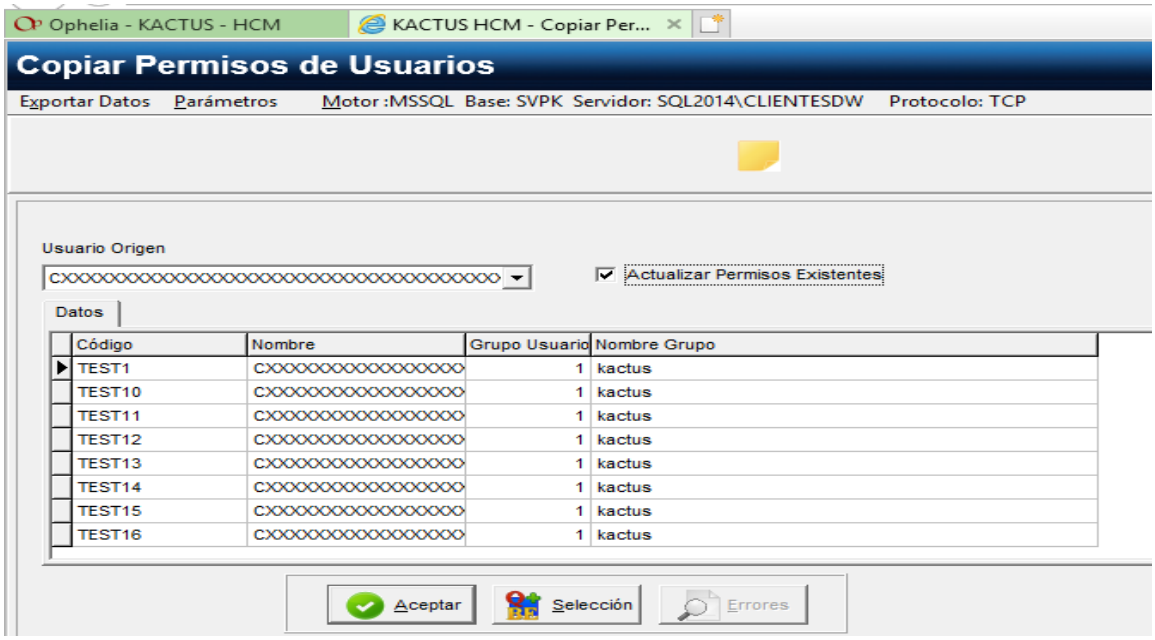


Figura 8. Asignación de permiso de usuarios en KGnCpusu

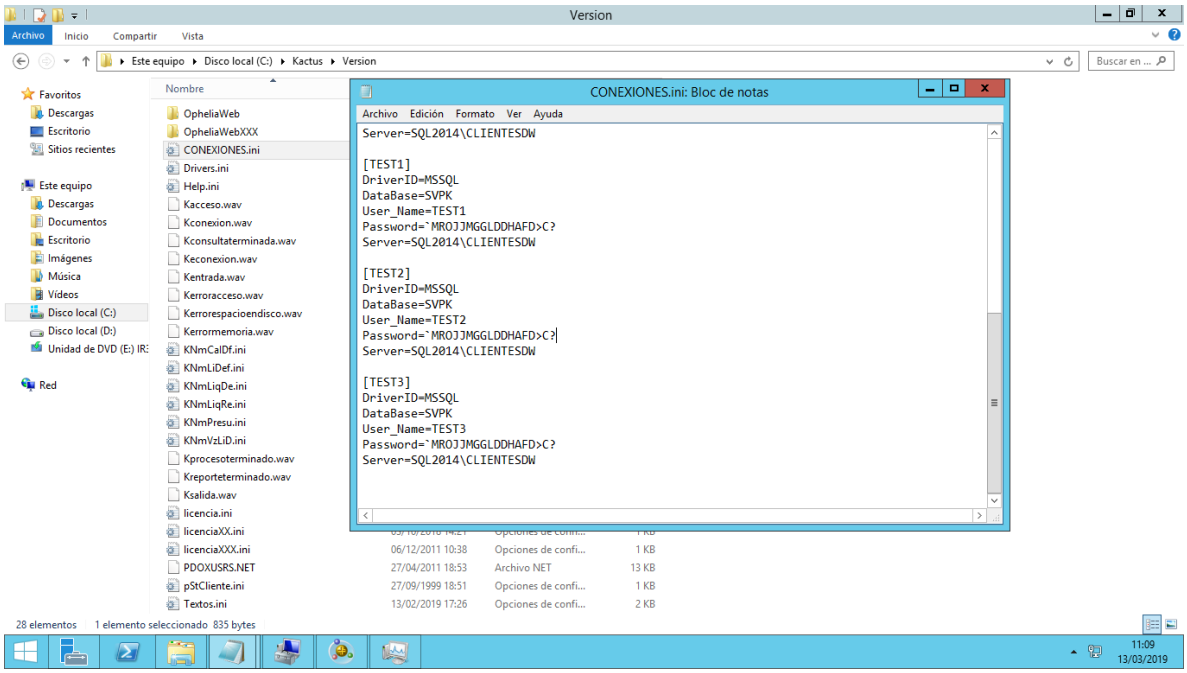


Figura 9. Archivo de configuración Conexiones.ini

5.7.3.4 VERIFICAR LA CONEXIÓN Y PERMISOS DE LOS USUARIOS CREADOS

Con relación a lo anterior en este paso se efectuó el inicio de sesión de cada usuario por el menú web de KACTUS-HCM (*Ophelia)* y se procedió a abrir el programa KNmLinom como se puede observar no presentaron inconveniente al ejecutarse (Ver figuras 10, 11 y 12).

Figura 10. Menú web de KACTUS-HCM (Ophelia)

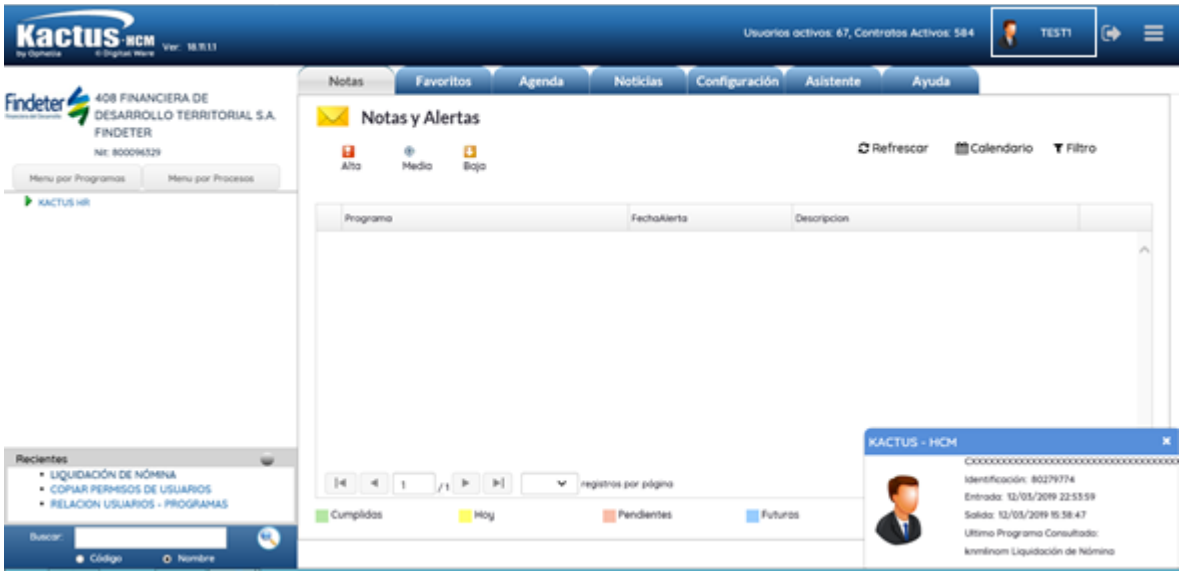


Figura 11. Ingreso de usuario por el menú web de KACTUS-HCM (Ophelia)

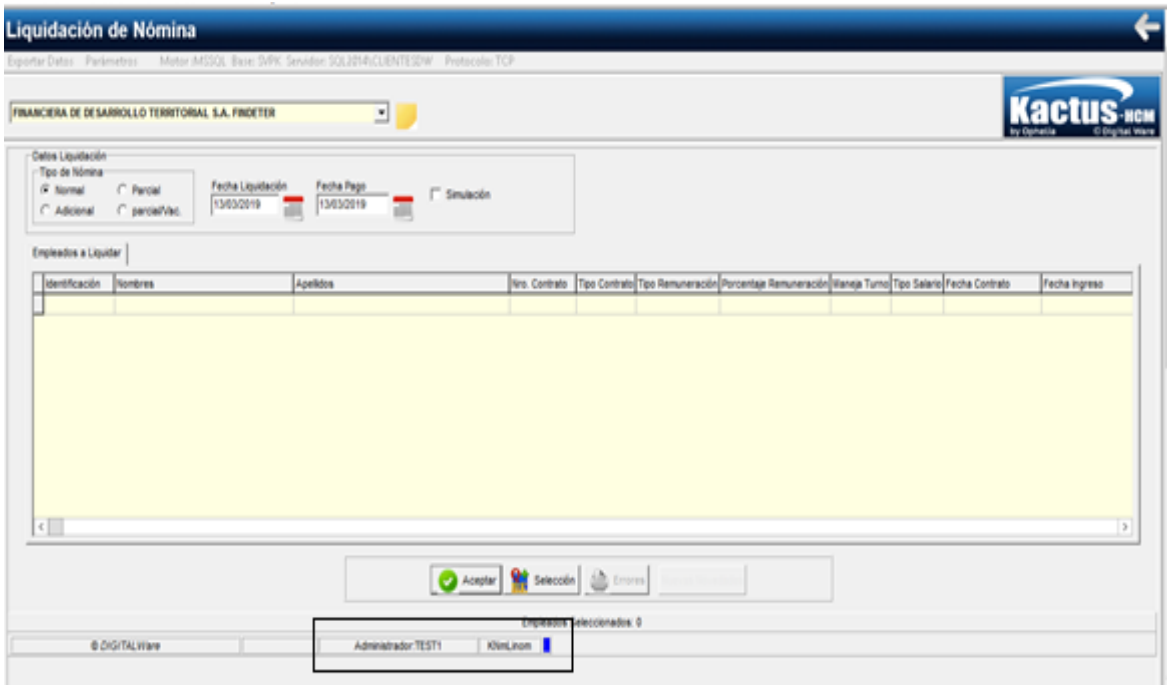


Figura 12. Ingreso al programa KNmLinom con el usuario TEST1.

5.7.4 INSTAURACIÓN DE PARÁMETROS ESTRUCTURALES

5.7.4.1 ESTRUCTURA PARA REALIZAR EL LOGIN

Esta estructura se compone de un vector en el que se ingresaron los datos necesarios para iniciar sesión por el servicio. A este vector lo componen nueve posiciones descritas anteriormente en 5.1.4.1,

```
Pdata = new string[9];
        Pdata[0] = "TEST1";
        Pdata[1] = "TEST1";
        Pdata[2] = "C175126128127176174";
        Pdata[3] = "D";
        Pdata[4] = "408";
        Pdata[5] = "KNmLinom";
        Pdata[6] = "132.147.154.165";
        Pdata[7] = "TCP";
        Pdata[8] = "N";
```

5.7.4.2 ESTRUCTURA DEL PROCESO

Esta estructura se compone de un vector de 16 posiciones en el que se ingresaron los datos necesarios para realizar el proceso de *KNmLinom,* las cuales son:

```
DateTime a = DateTime.Parse("12/02/2019"); Fecha de liquidación
DateTime b = DateTime.Parse("12/02/2019"); Fecha de Pago
pdataprocesar = new object[16];
string lStSqlQbe = Sentencia en SQL para la realización del filtro QBE.
pdataprocesar[0] = lStSqlQbe;
pdataprocesar[1] = a; Trae la fecha de liquidación
pdataprocesar[2] = b; Trae la fecha de pago
pdataprocesar[3] = true; Check de Simulación
pdataprocesar[4] = "N"; Tipo de Nómina
pdataprocesar[5] = "N"; Tipo de Liquidación
pdataprocesar[6] = "LIQUIDACION DE NOMINA";observaciones
pdataprocesar[7] = "N";
pdataprocesar[8] = false;Check de Aportes
pdataprocesar[9] = false;Ver aumentos
```

pdataprocesar[10] = false;Nuevas Novedades
pdataprocesar[11] = 1;Proceso
pdataprocesar[12] = Sesion;
pdataprocesar[13] = 1;Proceso
pdataprocesar[14] = 0;Validación Ley 1986 Educación
pdataprocesar[15] = 0; Validación Ley 1986 Salud

Y la configuración que se hizo para el caso de uso es:

```
DateTime a = DateTime.Parse("12/02/2019");
        DateTime b = DateTime.Parse("12/02/2019");
        pdataprocesar = new object[16];
        string lStSqlQbe = Dependiendo del caso de uso
        pdataprocesar[0] = lStSqlQbe;
        pdataprocesar[1] = a;
        pdataprocesar[2] = b;
        pdataprocesar[3] = true;
        pdataprocesar[4] = "N";
        pdataprocesar[5] = "N";
        pdataprocesar[6] = "LIQUIDACION DE NOMINA";
        pdataprocesar[7] = "N";
        pdataprocesar[8] = false;
        pdataprocesar[9] = false;
        pdataprocesar[10] = false;
        pdataprocesar[11] = 1;
        pdataprocesar[12] = Sesion;
        pdataprocesar[13] = 1;
        pdataprocesar[14] = 0;
        pdataprocesar[15] = 0;
```

5.7.5 DISEÑO DEL CASO DE PRUEBA EN VISUAL STUDIO ENTERPRISE CON LOS PARÁMETROS ANTERIORES

En el proyecto KactusLoadTest se deben configurar la estructura del login con los datos correspondientes seguidamente la estructura de KNmLinom como se muestra en la *figura 13.*

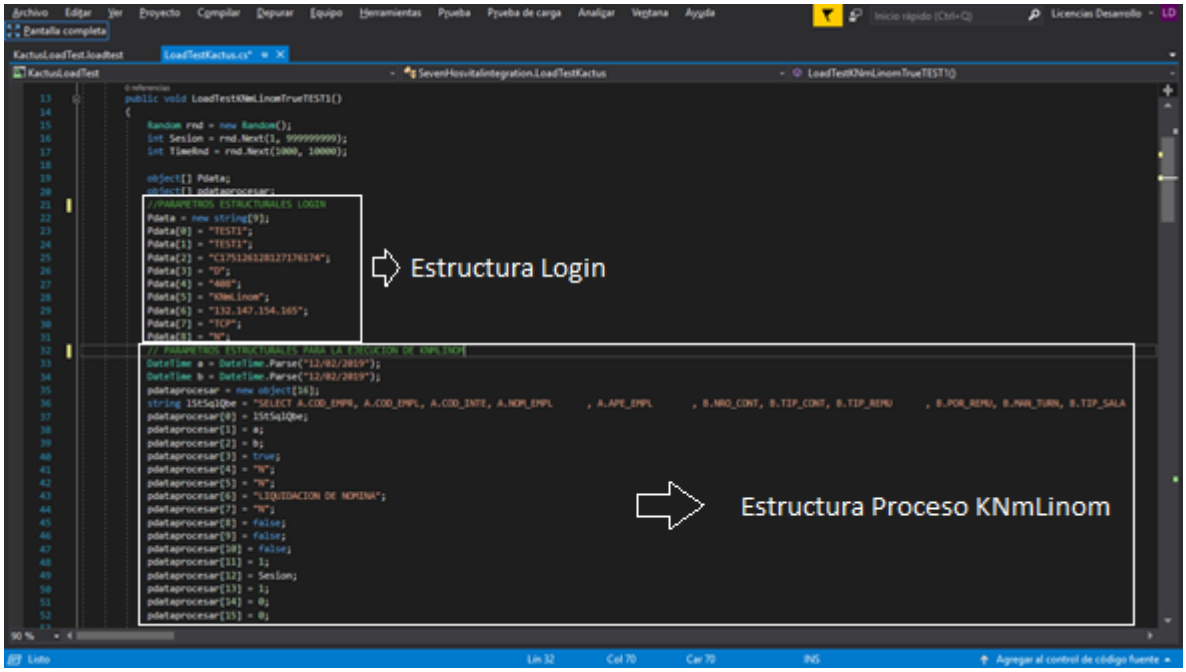


Figura 13. Diseño del caso de prueba en Visual Studio

5.7.6 SELECCIÓN DE LOS CASOS DE USO QUE SE INTEGRARAN CON EL CASO DE PRUEBA DESARROLLADO

Para el proceso de KNmLinom se realizara la selección de casos de uso mediante los requisitos funcionales asi que lo primero es realizar la consulta desde la interfaz grafica teniendo en cuenta que el contrato debe ser indefinio y el indicador de actividad debe estar activo se realiza el filtro con los parametros establecidos y obtenemos 235 empleados que cumplen con esta condición (Ver figura 14,15 y 16).

Figura 14. Filtro QBE con tipo de contrato Indefinido "I"

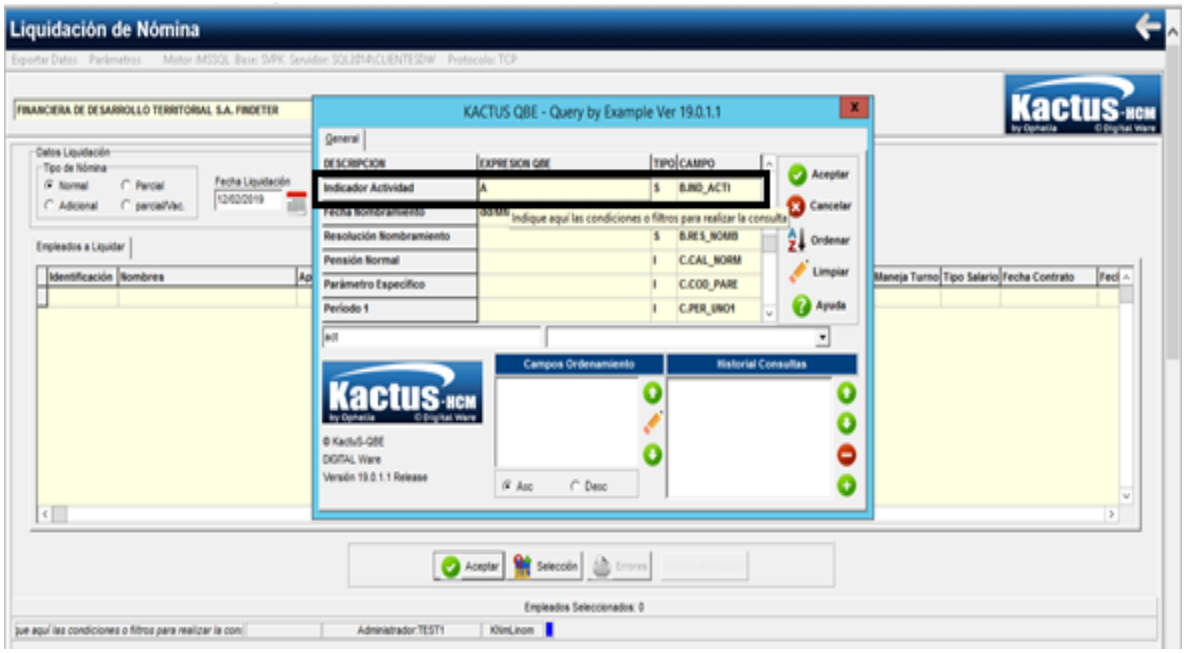


Figura 15. Filtro QBE indicador de actividad Activo "A"

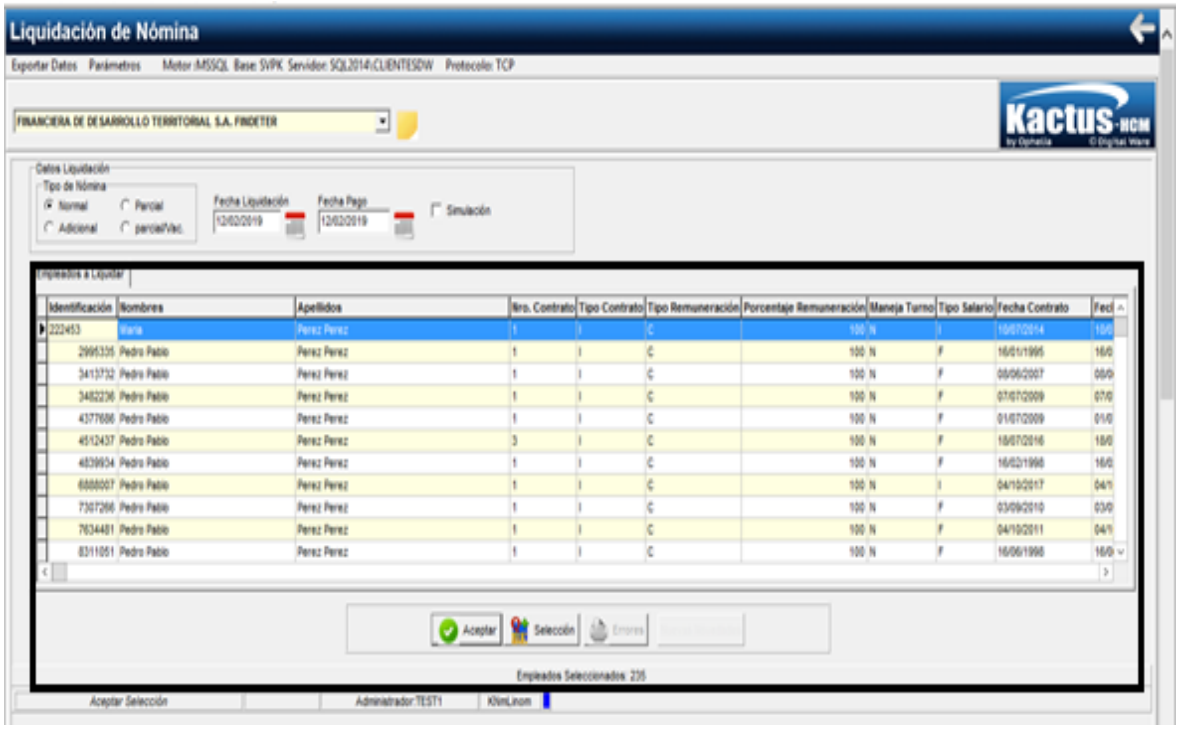


Figura 16. Resultado de la consulta aplicando los filtros

5.7.7 DISEÑO DE CASOS DE PRUEBA NECESARIOS

Para este proceso se aplicó una carga de 30 usuarios y proporcionando los empleados que cumplen la condicion (235) cada usuario tuvo un filtro que selecciono 7 u 8 empleados.

El metodo de prueba diseñado se replicó 30 veces(ver figura 17), al cual se le cambiaron en los parametros estrcuturales del login el usuario variando entre TEST1-TEST30, el alias que tambien varia en el rango anterior y el tipo de usuario que a partir del segundo metodo es ‘C’ por ser usuarios comunes. Y en los parametros del proceso KNmLinom se varia la consulta que hace el caso de uso para obtener empleados diferentes por usuario. (ver figura 18)

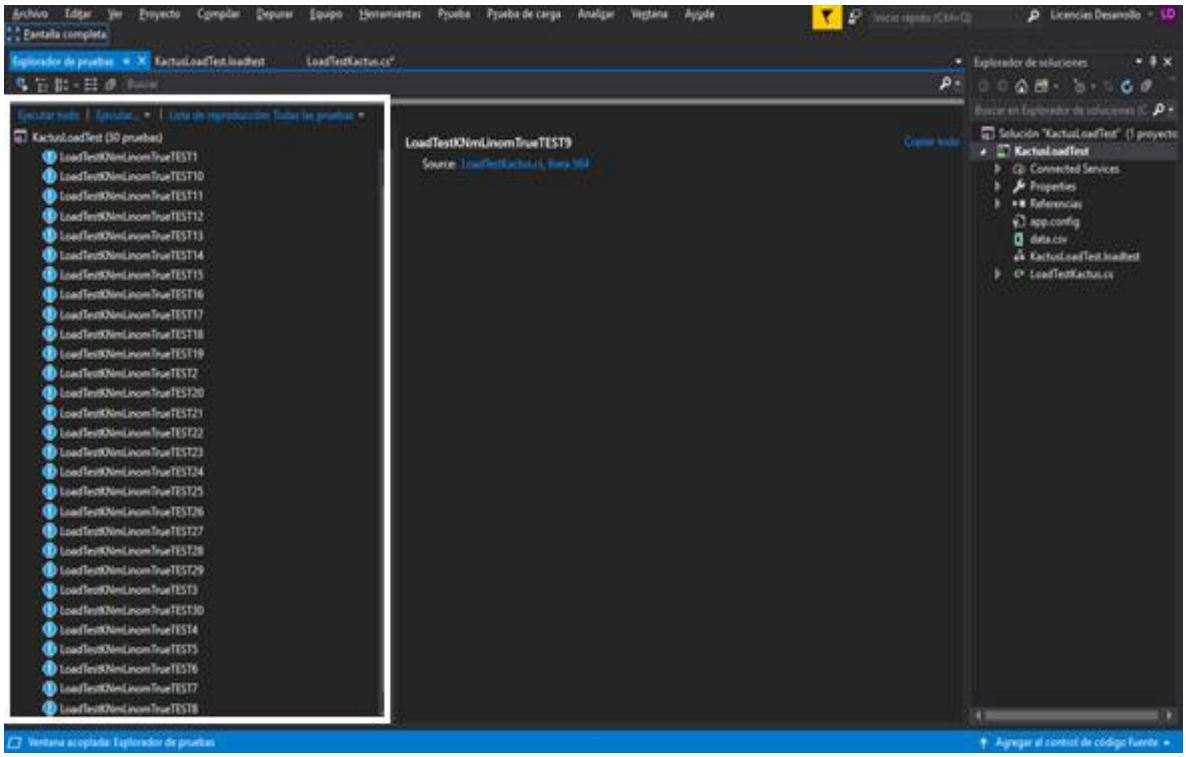


Figura 17. Métodos Replicados para los 30 usuarios.

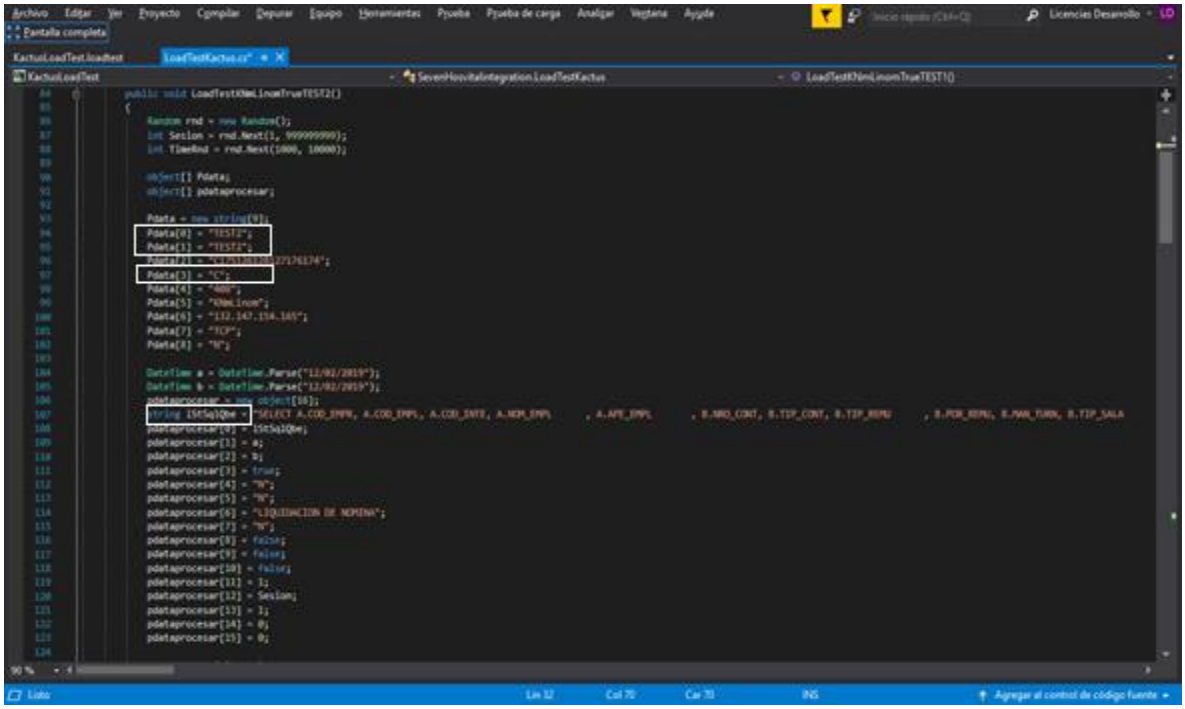


Figura 18. Cambios realizados en los métodos de prueba.

5.7.8 ASOCIACIÓN DE LOS CASOS DE PRUEBA A LA PRUEBA DE CARGA

En la prueba de carga se deberán asociar los casos de prueba con lo que se desea aplicar la prueba de carga. Esto se configura en el apartado Escenarios>Combinación de pruebas> Click derecho > Agregar pruebas (Ver figura 19)

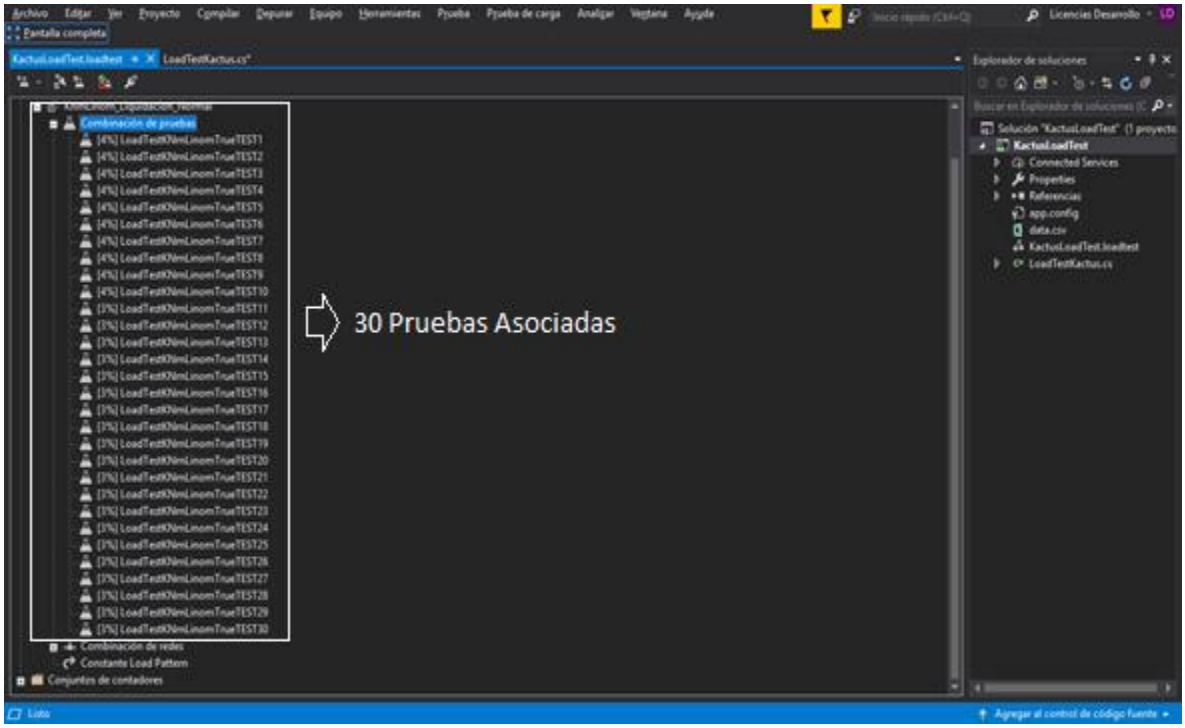


Figura 19. Pruebas asociadas a la prueba de carga

5.7.9 SEGUIMIENTO DE MÉTRICAS SEGÚN LA CARGA

En este paso se toman las métricas en 3 puntos con 1 usuario, 15 usuarios y 30 usuarios asociados a la prueba de carga. A continuación, se muestran las figuras del procedimiento para la configuración de las diferentes cargas.

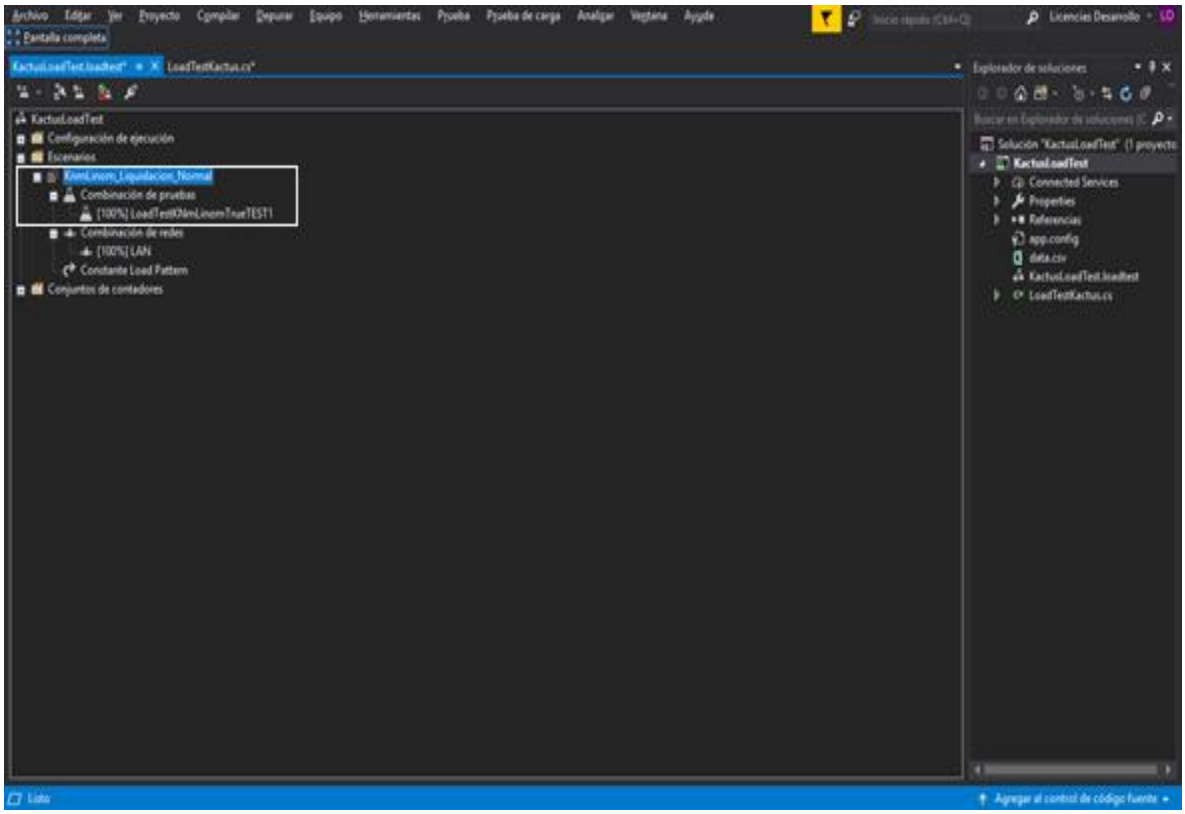


Figura 20. Asociar un método de prueba a la prueba de carga.

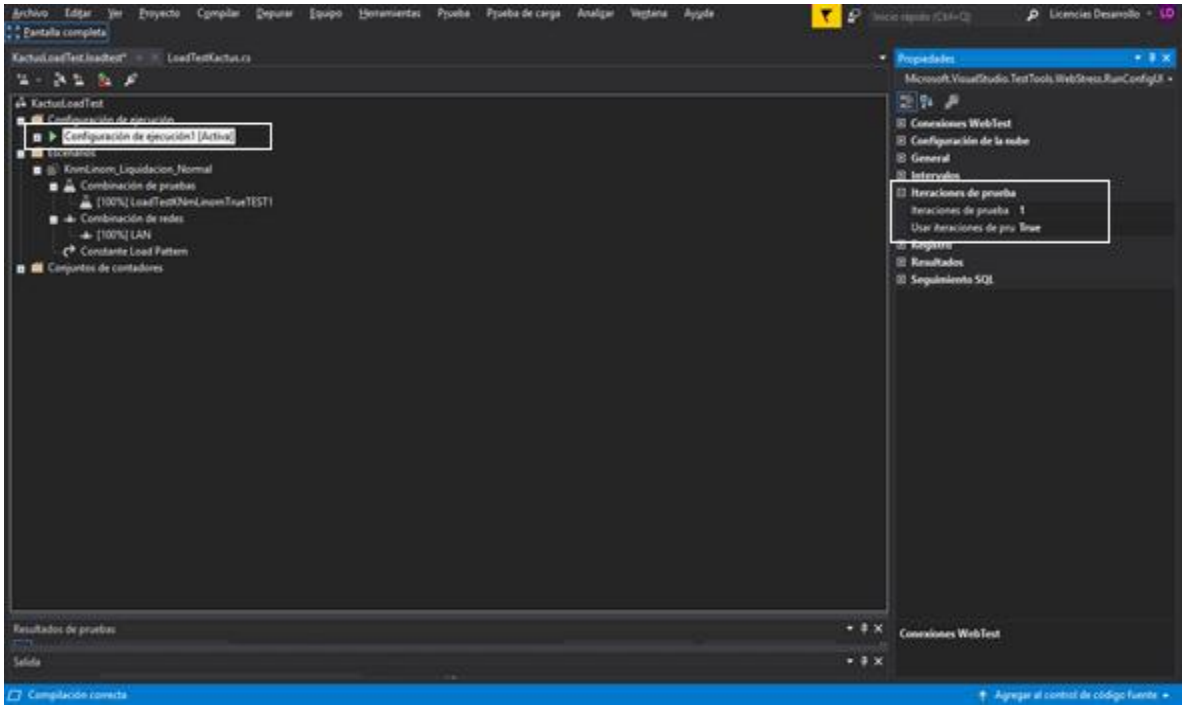


Figura 21. Configuración de las iteraciones de prueba de carga.

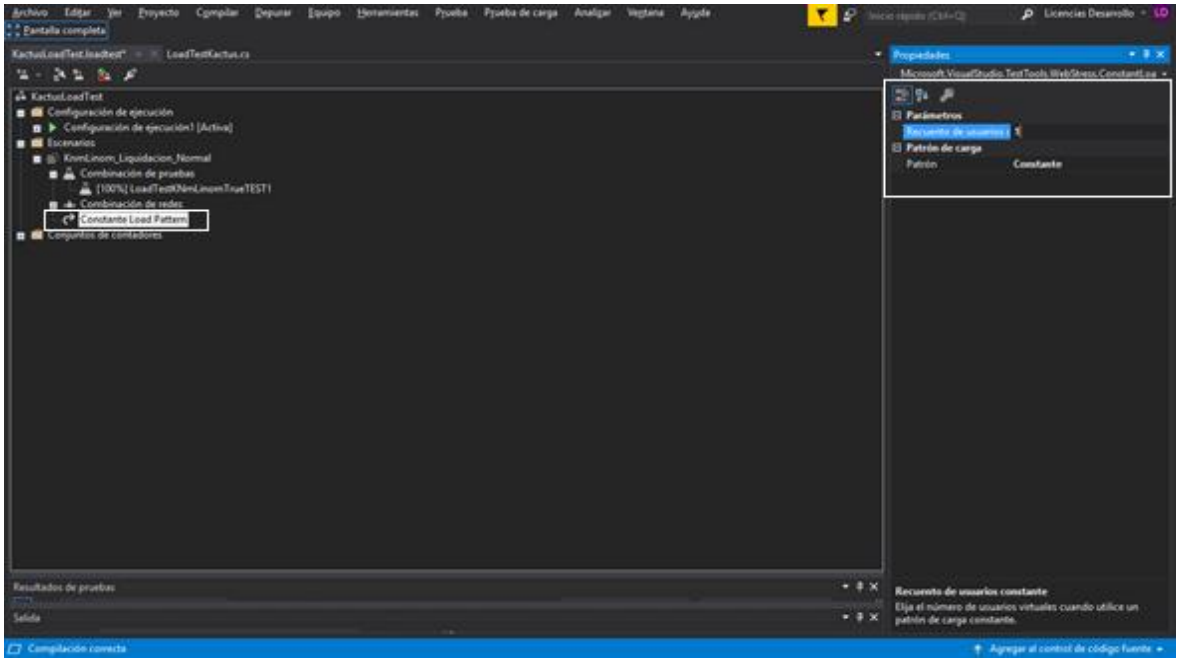


Figura 22. Configuración de la carga constante.

5.7.10 COMPARACIÓN DE MÉTRICAS

En la siguiente tabla se describen los tiempos de ejecución, los promedios de tiempo por prueba, los subprocesos y la memoria RAM que consume el proceso KNmLinom con diferentes cargas aplicadas.

Carga / Caracteristicas no funcionales	1 Usuario	15 Usuarios	30 Usuarios
Tiempo de ejecución de la prueba carga	12 (Seg)	57 (Seg)	95 (Seg)
Pruebas por segundo	0,078 (Seg)	0,26 (Seg)	0,31(Seg)
Promedio de tiempo por prueba	12 (Seg)	52,9 (Seg)	87,4 (Seg)
Subprocesos	9	26	39

Memoria RAM	21.040 KB	165.604 KB	305.510

Tabla 3. Comparación de Métricas

Como se ve reflejado en la tabla anterior los tiempos de ejecución, elproceso empieza a consumir mas memoria y a ejecutar mas subprocesos a medida que hay mas concurrencia.

5.7.11 INFORME DE RESULTADOS

Mediante el programa Visual Studio se pueden generar los informes comparativos en donde se pueden elegir dos resultados de pruebas ejecutadas y se eligen los componentes que se desean comparar. También se puede generar un informe de tendencia donde se muestran en un Excel los resultados obtenidos de las pruebas realizadas y así poder hacer una comparación con más de dos pruebas.

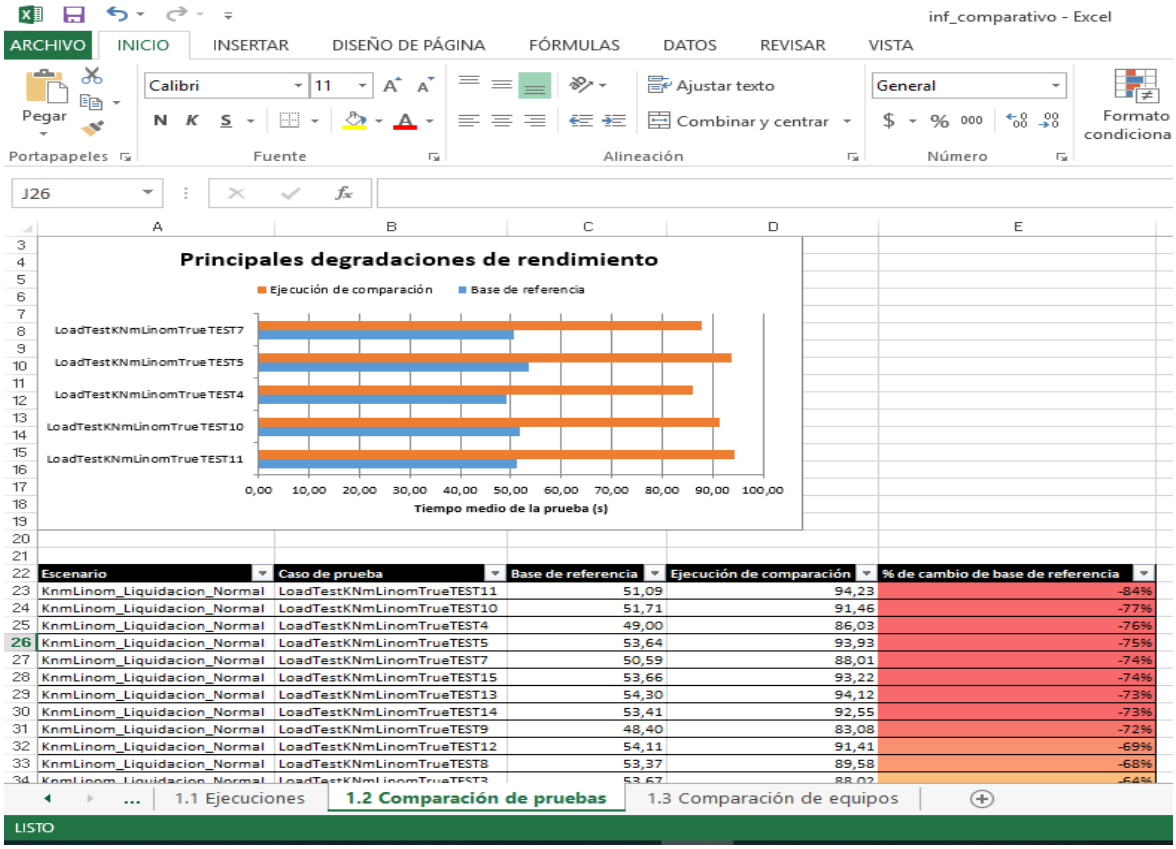


Figura 23. Informe comparativo entre base de referencia (15 usuarios) y ejecución de comparación (30 usuarios)

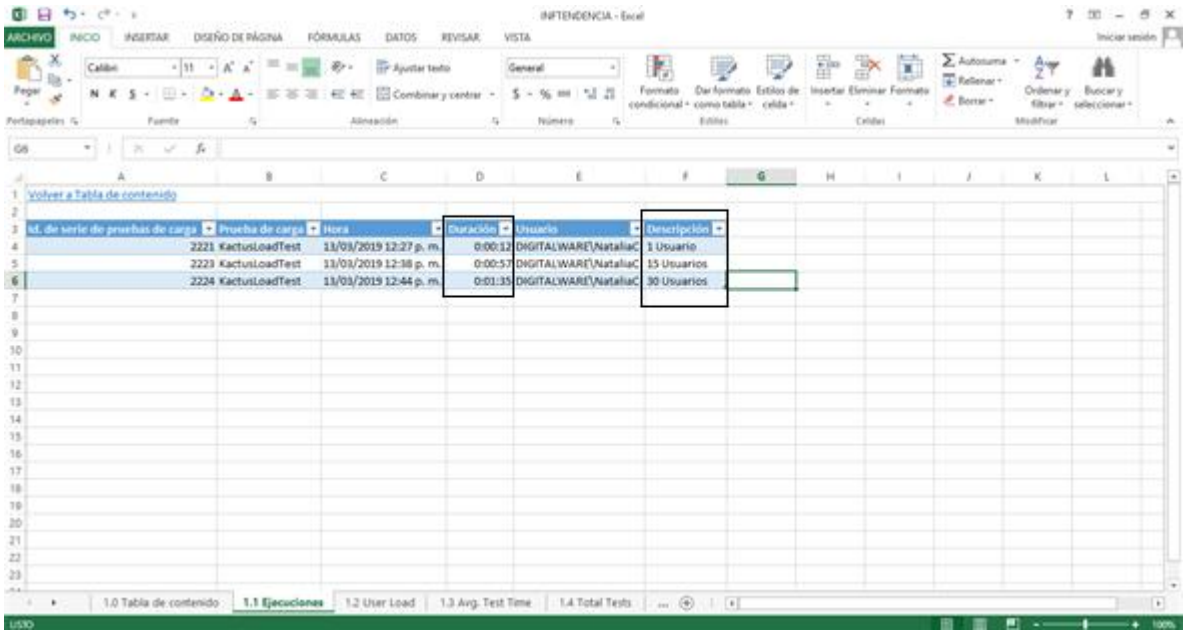


Figura 24. Informe tendencia de las tres pruebas realizadas con diferentes cargas

En estos informes se pueden apreciar varias características de las pruebas de carga y se realizan comparativas de las pruebas unitarias donde se observan las degradaciones de tiempo por la prueba ejecutada con una carga de 15 usuarios (Base de Referencia) contra la prueba con una carga de 30 usuarios (Ejecución de Comparación) y en la tabla que se puede visualizar debajo del gráfico se encuentra la más detallada la comparación de los tiempos (Ver figura 23) y en el informe de tendencia se pueden observar las comparativas de varias pruebas entorno a las características no funcionales, las ejecuciones y su duración (Ver figura 24).

6 IMPACTO DE LA PASANTÍA

Digital Ware por ser una compañía tecnológica permitió aplicar las habilidades y conocimientos propios a la formación académica previa a la realización de la pasantía, en un primer momento formando parte del área de trabajo: Administración de Operaciones liderada por el Ingeniero Edwin Barreto. Siendo este quien diese el punto de partida de esta labor mediante su propuesta de realizar la automatización de procesos tediosos y repetitivos del producto KACTUS-HCM para evaluar la funcionalidad del mismo.

Ser parte de esta área fomentó inicialmente la ejecución de la propuesta del Ingeniero basándose en el aporte constante de ideas que llevaran a cabo la automatización del proceso del liquidador de nómina, así como también el cumplir a cabalidad los objetivos de la misma. Los resultados por el rendimiento laboral brindaron la oportunidad de vinculación a la compañía acompañada del beneficio de expansión de trabajo en otras secciones del producto tales como:

- Selfservice
- Kactus RL (Reclutamiento web)
- Acompañamiento con checklist en programas maestros de nómina y gestión humana para salidas de servicepack (últimas versiones liberadas a los clientes).
- Programa de Incapacidades

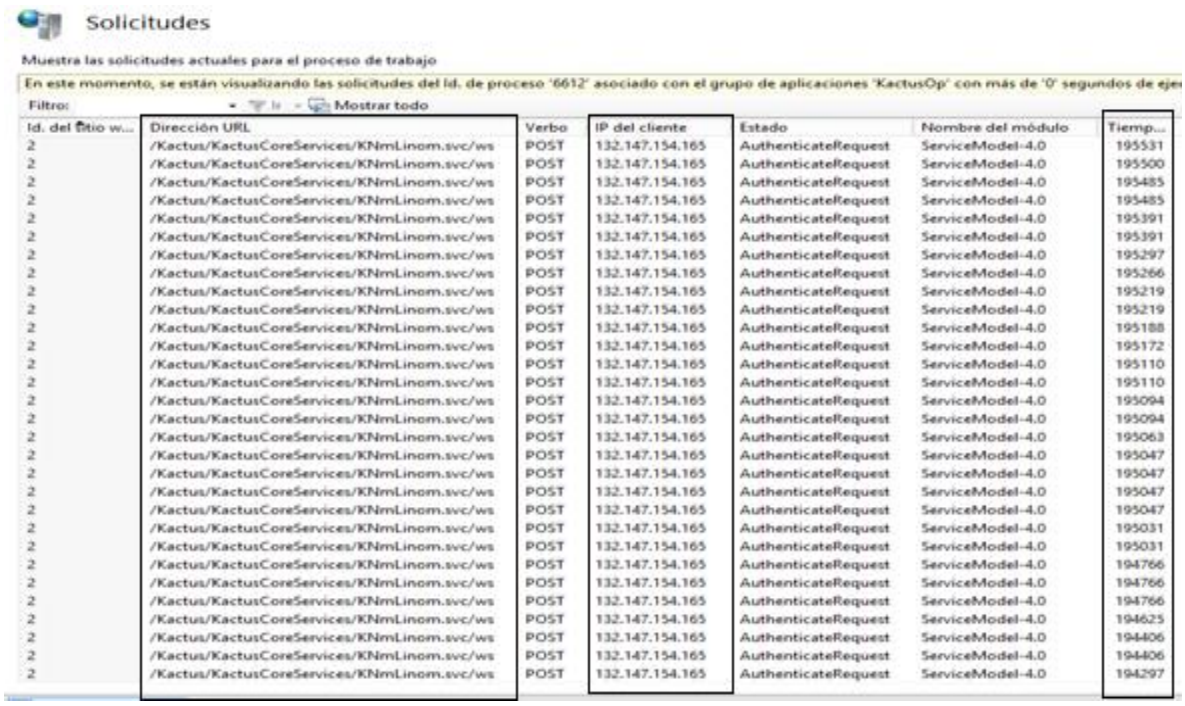
El resultado principal de esta pasantía fue realizar una metodología que complementara este proyecto con la ejecución de pruebas no funcionales para no solo garantizar funcionalidad sino también las condiciones de carga y respuesta que tuviera el proceso.

7 RESULTADOS

El liquidador de nómina (**KnmLinom**) no presenta errores de inserción de datos con carga constante de 1 a 30 usuarios liquidando 7 a 8 empleados por usuario, pero este presenta aumento en las características no funcionales ya que realizando una sola prueba con un usuario se demora 12 segundos y al tener concurrencia de 30 usuarios se llega a demorar hasta 94 segundos.

El correcto funcionamiento del liquidador de nómina está ligado a las condiciones de infraestructura en donde se ejecute y a los usuarios concurrentes en el proceso, por ende, el rendimiento y la eficiencia del liquidador (Tiempo de respuesta) son fuertemente afectados.

Como recomendación se debe ampliar la carga de la prueba en usuarios y empleados a liquidar. Para ampliar la carga de usuarios y empleados a liquidar el aprovisionamiento de la infraestructura es proporcional a la carga aplicar.


Solicitudes

Muestra las solicitudes actuales para el proceso de trabajo

En este momento, se están visualizando las solicitudes del Id. de proceso '6612' asociado con el grupo de aplicaciones 'KactusOp' con más de '0' segundos de eje

Filtro: Mostrar todo

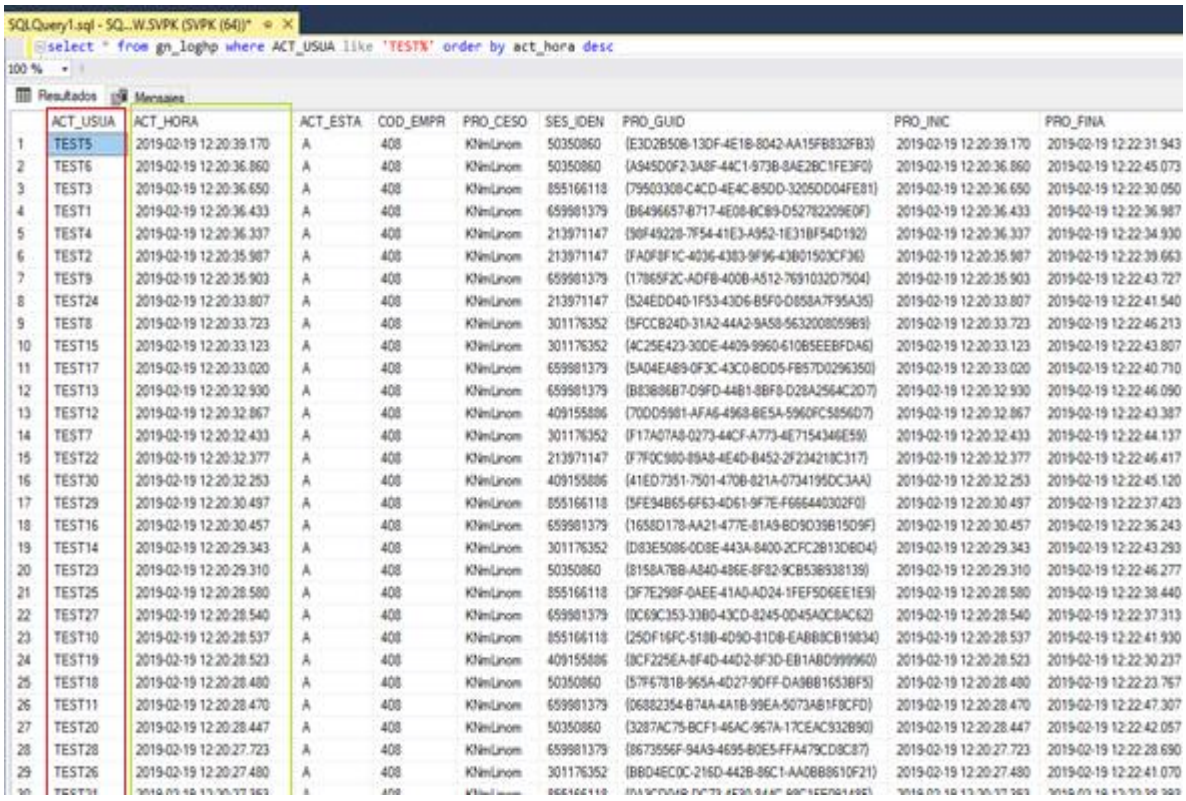
Id. del sitio w...	Dirección URL	Verbo	IP del cliente	Estado	Nombre del módulo	Tiemp...
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195531
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195500
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195485
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195485
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195391
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195391
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195297
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195266
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195219
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195219
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195188
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195172
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195110
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195110
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195094
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195094
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195063
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195047
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195047
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195047
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195047
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195031
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	195031
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194766
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194766
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194766
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194625
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194406
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194406
2	/Kactus/KactusCoreServices/KNmLinom.svc/ws	POST	132.147.154.165	AuthenticateRequest	ServiceModel-4.0	194297

Figura 25. Solicitudes visualizadas desde el software Internet Information Services (IIS)

En la figura anterior (ver *figura 25*) se pueden observar las solicitudes concurrentes que se le realizaron al servicio desde la máquina de ejecución de las pruebas de carga donde se representó que las solicitudes contenían la misma dirección IP solicitante.

Se realizó la validación de que la prueba se estuviera ejecutando correctamente, para esto se observaron las tablas involucradas, en *la figura 26* se relaciona los inicios de sesión de cada usuario y el tiempo ejecutado en esa sesión. Este seguimiento se realizó de esta manera para garantizar que el método de prueba estuviese registrando en cada una de las tablas en donde inserta el proceso los datos correspondientes a una liquidación de nómina exitosa.

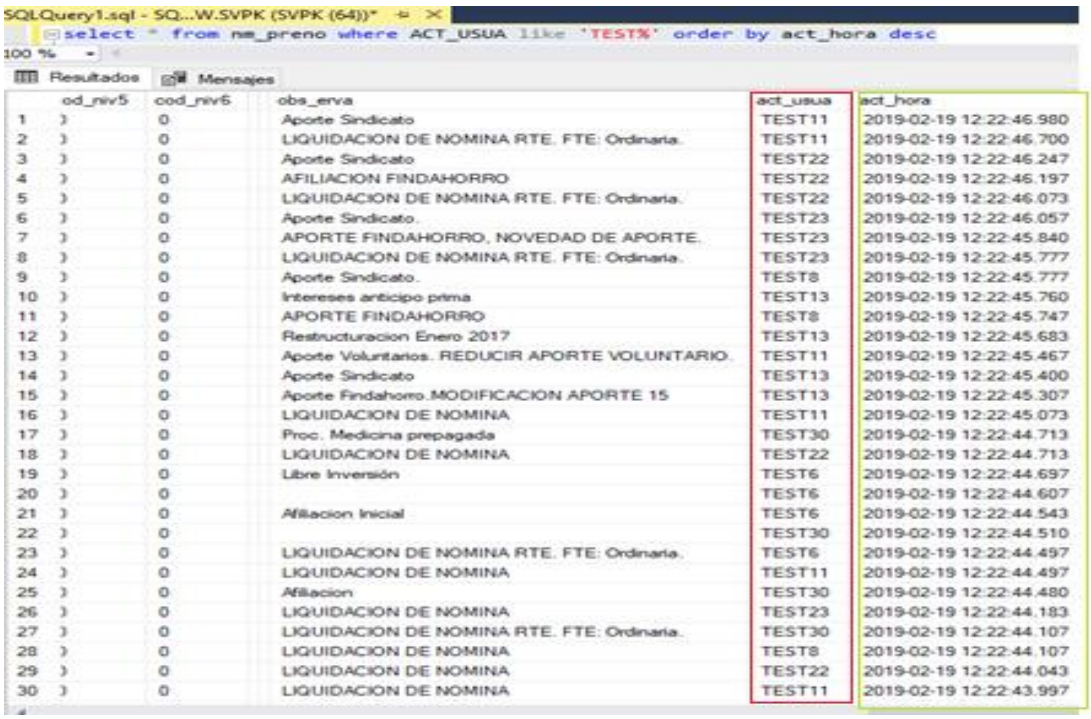
La concurrencia de los usuarios se puede ver reflejada en los campos de las tablas ***ACT_USUA*** Y ***ACT_HORA***, en el primero de ellos aparece el usuario que está actualizando la tabla y en el segundo la hora en la que se realiza dicha inserción. Como se puede observar en las figuras la diferencia suele ser de milisegundos en los que puede haber inserciones de los usuarios TEST1 a TEST30.



	ACT_USUA	ACT_HORA	ACT_ESTA	COD_EMPR	PRO_CESO	SES_IDEN	PRO_GUID	PRO_INIC	PRO_FINA
1	TEST5	2019-02-19 12:20:39.170	A	408	KNmUnom	50350860	{E3D2B50B-13DF-4E1B-8042-AA15FB832FB3}	2019-02-19 12:20:39.170	2019-02-19 12:22:31.943
2	TEST6	2019-02-19 12:20:36.860	A	408	KNmUnom	50350860	{A945D0F2-3A8F-44C1-973B-8AE2BC1FE3F0}	2019-02-19 12:20:36.860	2019-02-19 12:22:45.073
3	TEST3	2019-02-19 12:20:36.650	A	408	KNmUnom	855166118	{79503308-C4CD-4E4C-B5DD-3205DD04FE81}	2019-02-19 12:20:36.650	2019-02-19 12:22:30.050
4	TEST1	2019-02-19 12:20:36.433	A	408	KNmUnom	659981379	{B6496657-B717-4E08-BCB9-D52782209E0F}	2019-02-19 12:20:36.433	2019-02-19 12:22:36.987
5	TEST4	2019-02-19 12:20:36.337	A	408	KNmUnom	213971147	{90F49228-7F54-41E3-A952-1E31BF54D192}	2019-02-19 12:20:36.337	2019-02-19 12:22:34.930
6	TEST2	2019-02-19 12:20:35.987	A	408	KNmUnom	213971147	{FA0F8F1C-4036-4383-9F96-43B01503CF36}	2019-02-19 12:20:35.987	2019-02-19 12:22:39.663
7	TEST9	2019-02-19 12:20:35.903	A	408	KNmUnom	659981379	{17865F2C-ADFB-400B-A512-7691032D7504}	2019-02-19 12:20:35.903	2019-02-19 12:22:43.727
8	TEST24	2019-02-19 12:20:33.807	A	408	KNmUnom	213971147	{524EDD40-1F53-43D6-B5F0-D858A7F95A35}	2019-02-19 12:20:33.807	2019-02-19 12:22:41.540
9	TEST8	2019-02-19 12:20:33.723	A	408	KNmUnom	301176352	{5FCCB24D-31A2-44A2-9A58-563200805989}	2019-02-19 12:20:33.723	2019-02-19 12:22:46.213
10	TEST15	2019-02-19 12:20:33.123	A	408	KNmUnom	301176352	{4C25E423-30DE-4409-9960-610B5EEBFDA6}	2019-02-19 12:20:33.123	2019-02-19 12:22:43.807
11	TEST17	2019-02-19 12:20:33.020	A	408	KNmUnom	659981379	{5A04EAB9-0F3C-43C0-BDD5-FB57D0296350}	2019-02-19 12:20:33.020	2019-02-19 12:22:40.710
12	TEST13	2019-02-19 12:20:32.930	A	408	KNmUnom	659981379	{B83B86B7-D9FD-44B1-8BF8-D28A2564C2D7}	2019-02-19 12:20:32.930	2019-02-19 12:22:46.090
13	TEST12	2019-02-19 12:20:32.867	A	408	KNmUnom	409155886	{70DD5981-AFA6-4968-BE5A-5960FC5856D7}	2019-02-19 12:20:32.867	2019-02-19 12:22:43.387
14	TEST7	2019-02-19 12:20:32.433	A	408	KNmUnom	301176352	{F17A07A8-0273-44CF-A773-4E7154346E59}	2019-02-19 12:20:32.433	2019-02-19 12:22:44.137
15	TEST22	2019-02-19 12:20:32.377	A	408	KNmUnom	213971147	{F7F0C980-89A8-4E4D-B452-2F234218C317}	2019-02-19 12:20:32.377	2019-02-19 12:22:46.417
16	TEST30	2019-02-19 12:20:32.253	A	408	KNmUnom	409155886	{41ED7351-7501-470B-821A-0734195DC3AA}	2019-02-19 12:20:32.253	2019-02-19 12:22:45.120
17	TEST29	2019-02-19 12:20:30.497	A	408	KNmUnom	855166118	{5FE94B65-6F63-4D61-9F7E-F666440302F0}	2019-02-19 12:20:30.497	2019-02-19 12:22:37.423
18	TEST16	2019-02-19 12:20:30.457	A	408	KNmUnom	659981379	{1658D178-AA21-477E-81A9-BD9D39B15D9F}	2019-02-19 12:20:30.457	2019-02-19 12:22:36.243
19	TEST14	2019-02-19 12:20:29.343	A	408	KNmUnom	301176352	{D83E5086-0D8E-443A-8400-2CFC2B13D8D4}	2019-02-19 12:20:29.343	2019-02-19 12:22:43.293
20	TEST23	2019-02-19 12:20:29.310	A	408	KNmUnom	50350860	{8158A7BB-A840-486E-8F82-9CB53B938139}	2019-02-19 12:20:29.310	2019-02-19 12:22:46.277
21	TEST25	2019-02-19 12:20:28.580	A	408	KNmUnom	855166118	{3F7E298F-0AEE-41A0-AD24-1FEF5D6EE1E5}	2019-02-19 12:20:28.580	2019-02-19 12:22:38.440
22	TEST27	2019-02-19 12:20:28.540	A	408	KNmUnom	659981379	{0C69C353-33B0-43CD-8245-0D45A0C8AC62}	2019-02-19 12:20:28.540	2019-02-19 12:22:37.313
23	TEST10	2019-02-19 12:20:28.537	A	408	KNmUnom	855166118	{250F16FC-518B-4D90-81DB-EABB8CB19834}	2019-02-19 12:20:28.537	2019-02-19 12:22:41.930
24	TEST19	2019-02-19 12:20:28.523	A	408	KNmUnom	409155886	{8CF225EA-8F4D-44D2-8F3D-EB1ABD999960}	2019-02-19 12:20:28.523	2019-02-19 12:22:30.237
25	TEST18	2019-02-19 12:20:28.480	A	408	KNmUnom	50350860	{57F6781B-965A-4D27-9DFF-DA9BB1653BF5}	2019-02-19 12:20:28.480	2019-02-19 12:22:23.767
26	TEST11	2019-02-19 12:20:28.470	A	408	KNmUnom	659981379	{06882354-B74A-4A1B-99EA-5073AB1F8CFD}	2019-02-19 12:20:28.470	2019-02-19 12:22:47.307
27	TEST20	2019-02-19 12:20:28.447	A	408	KNmUnom	50350860	{3287AC75-BCF1-46AC-967A-17CEAC932B90}	2019-02-19 12:20:28.447	2019-02-19 12:22:42.057
28	TEST28	2019-02-19 12:20:27.723	A	408	KNmUnom	659981379	{8673556F-94A9-4695-B0E5-FFA479CD8C87}	2019-02-19 12:20:27.723	2019-02-19 12:22:28.690
29	TEST26	2019-02-19 12:20:27.480	A	408	KNmUnom	301176352	{BBD4EC0C-216D-442B-86C1-AA0BB8610F21}	2019-02-19 12:20:27.480	2019-02-19 12:22:41.070

Figura 26. Tabla de inicio de sesión donde se ve reflejada la concurrencia

Del mismo modo en la *figura 27* se muestra la tabla donde se registran los datos del desprendible de nómina,



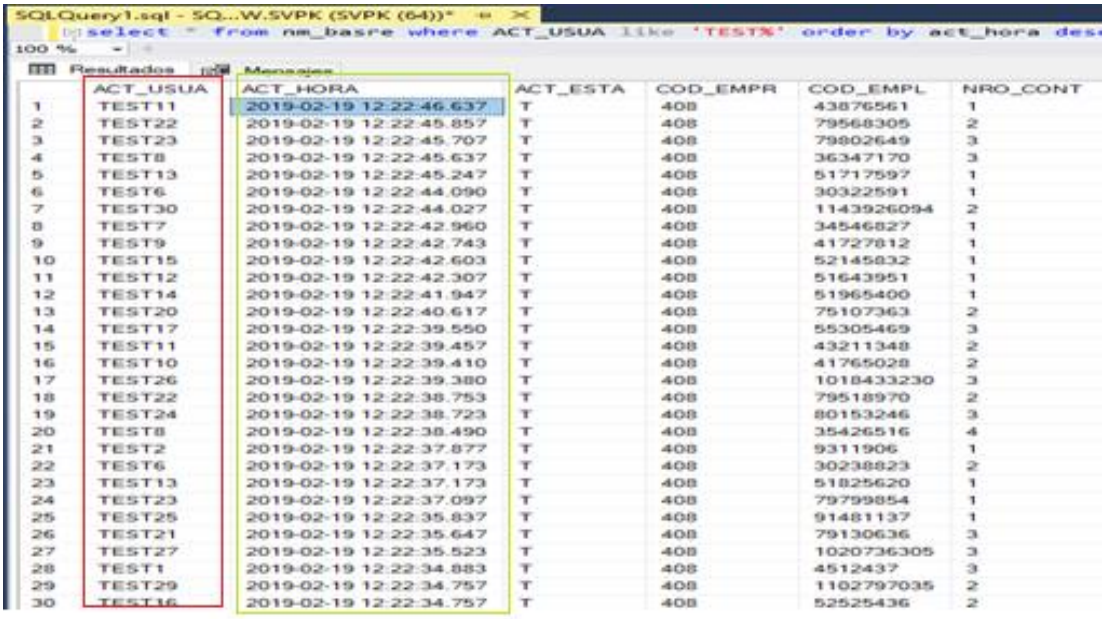
```
SQLQuery1.sql - SQ...W.SVPK (SVPK (64))*
select * from nm_preno where ACT_USUA like 'TEST%' order by act_hora desc
```

Resultados | Mensajes

	od_niv5	cod_niv6	obs_erva	act_usua	act_hora
1	)	0	Aporte Sindicato	TEST11	2019-02-19 12:22:46.980
2	)	0	LIQUIDACION DE NOMINA RTE. FTE: Ordinaria.	TEST11	2019-02-19 12:22:46.700
3	)	0	Aporte Sindicato	TEST22	2019-02-19 12:22:46.247
4	)	0	AFILIACION FINDAHORRO	TEST22	2019-02-19 12:22:46.197
5	)	0	LIQUIDACION DE NOMINA RTE. FTE: Ordinaria.	TEST22	2019-02-19 12:22:46.073
6	)	0	Aporte Sindicato.	TEST23	2019-02-19 12:22:46.057
7	)	0	APORTE FINDAHORRO, NOVEDAD DE APORTE.	TEST23	2019-02-19 12:22:45.840
8	)	0	LIQUIDACION DE NOMINA RTE. FTE: Ordinaria.	TEST23	2019-02-19 12:22:45.777
9	)	0	Aporte Sindicato.	TEST8	2019-02-19 12:22:45.777
10	)	0	Intereses anticipo prima	TEST13	2019-02-19 12:22:45.760
11	)	0	APORTE FINDAHORRO	TEST8	2019-02-19 12:22:45.747
12	)	0	Restructuracion Enero 2017	TEST13	2019-02-19 12:22:45.683
13	)	0	Aporte Voluntarios. REDUCIR APORTE VOLUNTARIO.	TEST11	2019-02-19 12:22:45.467
14	)	0	Aporte Sindicato	TEST13	2019-02-19 12:22:45.400
15	)	0	Aporte Findahorro.MODIFICACION APORTE 15	TEST13	2019-02-19 12:22:45.307
16	)	0	LIQUIDACION DE NOMINA	TEST11	2019-02-19 12:22:45.073
17	)	0	Proc. Medicina prepagada	TEST30	2019-02-19 12:22:44.713
18	)	0	LIQUIDACION DE NOMINA	TEST22	2019-02-19 12:22:44.713
19	)	0	Libre Inversión	TEST6	2019-02-19 12:22:44.697
20	)	0		TEST6	2019-02-19 12:22:44.607
21	)	0	Afiliacion Inicial	TEST6	2019-02-19 12:22:44.543
22	)	0		TEST30	2019-02-19 12:22:44.510
23	)	0	LIQUIDACION DE NOMINA RTE. FTE: Ordinaria.	TEST6	2019-02-19 12:22:44.497
24	)	0	LIQUIDACION DE NOMINA	TEST11	2019-02-19 12:22:44.497
25	)	0	Afiliacion	TEST30	2019-02-19 12:22:44.480
26	)	0	LIQUIDACION DE NOMINA	TEST23	2019-02-19 12:22:44.183
27	)	0	LIQUIDACION DE NOMINA RTE. FTE: Ordinaria.	TEST30	2019-02-19 12:22:44.107
28	)	0	LIQUIDACION DE NOMINA	TEST8	2019-02-19 12:22:44.107
29	)	0	LIQUIDACION DE NOMINA	TEST22	2019-02-19 12:22:44.043
30	)	0	LIQUIDACION DE NOMINA	TEST11	2019-02-19 12:22:43.997

Figura 27 Tabla pre-nómina donde el proceso KNmLinom debe insertar datos

Por otro lado, en *la figura 28* el proceso inserta los registros de las bases de retención en la fuente aplicada



```
SQLQuery1.sql - SQ...W.SVPK (SVPK (64))*
select * from nm_basre where ACT_USUA like 'TEST%' order by act_hora desc
```

Resultados | Mensajes

	ACT_USUA	ACT_HORA	ACT_ESTA	COD_EMPR	COD_EMPL	NRO_CONT
1	TEST11	2019-02-19 12:22:46.637	T	408	43876561	1
2	TEST22	2019-02-19 12:22:45.857	T	408	79568305	2
3	TEST23	2019-02-19 12:22:45.707	T	408	79802649	3
4	TEST8	2019-02-19 12:22:45.637	T	408	36347170	3
5	TEST13	2019-02-19 12:22:45.247	T	408	51717597	1
6	TEST6	2019-02-19 12:22:44.090	T	408	30322591	1
7	TEST30	2019-02-19 12:22:44.027	T	408	1143926094	2
8	TEST7	2019-02-19 12:22:42.960	T	408	34546827	1
9	TEST9	2019-02-19 12:22:42.743	T	408	41727812	1
10	TEST15	2019-02-19 12:22:42.603	T	408	52145832	1
11	TEST12	2019-02-19 12:22:42.307	T	408	51643951	1
12	TEST14	2019-02-19 12:22:41.947	T	408	51965400	1
13	TEST20	2019-02-19 12:22:40.617	T	408	75107363	2
14	TEST17	2019-02-19 12:22:39.550	T	408	55305469	3
15	TEST11	2019-02-19 12:22:39.457	T	408	43211348	2
16	TEST10	2019-02-19 12:22:39.410	T	408	41765028	2
17	TEST26	2019-02-19 12:22:39.380	T	408	1018433230	3
18	TEST22	2019-02-19 12:22:38.753	T	408	79518970	2
19	TEST24	2019-02-19 12:22:38.723	T	408	80153246	3
20	TEST8	2019-02-19 12:22:38.490	T	408	35426516	4
21	TEST2	2019-02-19 12:22:37.877	T	408	9311906	1
22	TEST6	2019-02-19 12:22:37.173	T	408	30238823	2
23	TEST13	2019-02-19 12:22:37.173	T	408	51825620	1
24	TEST23	2019-02-19 12:22:37.097	T	408	79799854	1
25	TEST25	2019-02-19 12:22:35.837	T	408	91481137	1
26	TEST21	2019-02-19 12:22:35.647	T	408	79130636	3
27	TEST27	2019-02-19 12:22:35.523	T	408	1020736305	3
28	TEST1	2019-02-19 12:22:34.883	T	408	4512437	3
29	TEST29	2019-02-19 12:22:34.757	T	408	1102797035	2
30	TEST16	2019-02-19 12:22:34.757	T	408	52525436	2

Figura 28. Tabla bases de retención donde el proceso KNmLinom debe insertar datos

Por último en la *figura 29* se muestra la tabla de novedades de seguridad social donde quedan los registros insertados por el proceso KNmLinom

```
select * from nm_novss where ACT_USUA like 'TEST%' order by act_hora desc
```

100 %

Resultados | Mensajes

	ACT_USUA	ACT_HORA	ACT_ESTA	CAS_CONT	COD_EMPR	FEC_ACUM
1	TEST11	2019-02-19 12:22:43.150	T	NULL	408	2019-02-12 00:00:00.000
2	TEST22	2019-02-19 12:22:43.023	T	NULL	408	2019-02-12 00:00:00.000
3	TEST8	2019-02-19 12:22:42.977	T	NULL	408	2019-02-12 00:00:00.000
4	TEST23	2019-02-19 12:22:42.930	T	NULL	408	2019-02-12 00:00:00.000
5	TEST13	2019-02-19 12:22:42.743	T	NULL	408	2019-02-12 00:00:00.000
6	TEST6	2019-02-19 12:22:41.540	T	NULL	408	2019-02-12 00:00:00.000
7	TEST30	2019-02-19 12:22:40.117	T	NULL	408	2019-02-12 00:00:00.000
8	TEST9	2019-02-19 12:22:38.847	T	NULL	408	2019-02-12 00:00:00.000
9	TEST7	2019-02-19 12:22:37.737	T	NULL	408	2019-02-12 00:00:00.000
10	TEST14	2019-02-19 12:22:34.663	T	NULL	408	2019-02-12 00:00:00.000
11	TEST15	2019-02-19 12:22:34.007	T	NULL	408	2019-02-12 00:00:00.000
12	TEST12	2019-02-19 12:22:32.130	T	NULL	408	2019-02-12 00:00:00.000
13	TEST20	2019-02-19 12:22:30.127	T	NULL	408	2019-02-12 00:00:00.000
14	TEST24	2019-02-19 12:22:27.377	T	NULL	408	2019-02-12 00:00:00.000
15	TEST17	2019-02-19 12:22:27.317	T	NULL	408	2019-02-12 00:00:00.000
16	TEST26	2019-02-19 12:22:27.020	T	NULL	408	2019-02-12 00:00:00.000
17	TEST11	2019-02-19 12:22:26.753	T	NULL	408	2019-02-12 00:00:00.000
18	TEST2	2019-02-19 12:22:26.473	T	NULL	408	2019-02-12 00:00:00.000
19	TEST22	2019-02-19 12:22:25.863	T	NULL	408	2019-02-12 00:00:00.000
20	TEST10	2019-02-19 12:22:25.800	T	NULL	408	2019-02-12 00:00:00.000
21	TEST8	2019-02-19 12:22:25.660	T	NULL	408	2019-02-12 00:00:00.000
22	TEST6	2019-02-19 12:22:25.177	T	NULL	408	2019-02-12 00:00:00.000
23	TEST5	2019-02-19 12:22:24.300	T	NULL	408	2019-02-12 00:00:00.000
24	TEST13	2019-02-19 12:22:24.207	T	NULL	408	2019-02-12 00:00:00.000
25	TEST23	2019-02-19 12:22:23.253	T	NULL	408	2019-02-12 00:00:00.000
26	TEST21	2019-02-19 12:22:23.207	T	NULL	408	2019-02-12 00:00:00.000
27	TEST25	2019-02-19 12:22:23.160	T	NULL	408	2019-02-12 00:00:00.000
28	TEST27	2019-02-19 12:22:23.127	T	NULL	408	2019-02-12 00:00:00.000
29	TEST16	2019-02-19 12:22:22.627	T	NULL	408	2019-02-12 00:00:00.000
30	TEST29	2019-02-19 12:22:22.127	T	NULL	408	2019-02-12 00:00:00.000

Figura 29. Tabla novedades de seguridad social donde el proceso KNmLinom debe insertar datos

En las figuras 30,31 y 32 se observa la ejecución de las pruebas realizadas con diferentes cargas aplicadas progresivamente 1, 15 y 30 usuarios donde se pueden apreciar los mínimos y máximos de cada característica no funcional. A continuación se describe brevemente las características y los sus etiquetas para apreciarlas en las figuras.

User Load (Carga de usuario): Número de usuarios que se va a cargar a la prueba

Tests/sec (Pruebas/segundo): Pruebas que se realizan por segundo

Avg. Test Time (Promedio Tiempo de prueba): Tiempos promedio de cada prueba

Thread Count (Recuento de Hilos): Subprocesos en el servidor donde se ejecuta la prueba

Private Bytes (Bytes privados): Consumo memoria RAM en el servidor donde se ejecuta la prueba

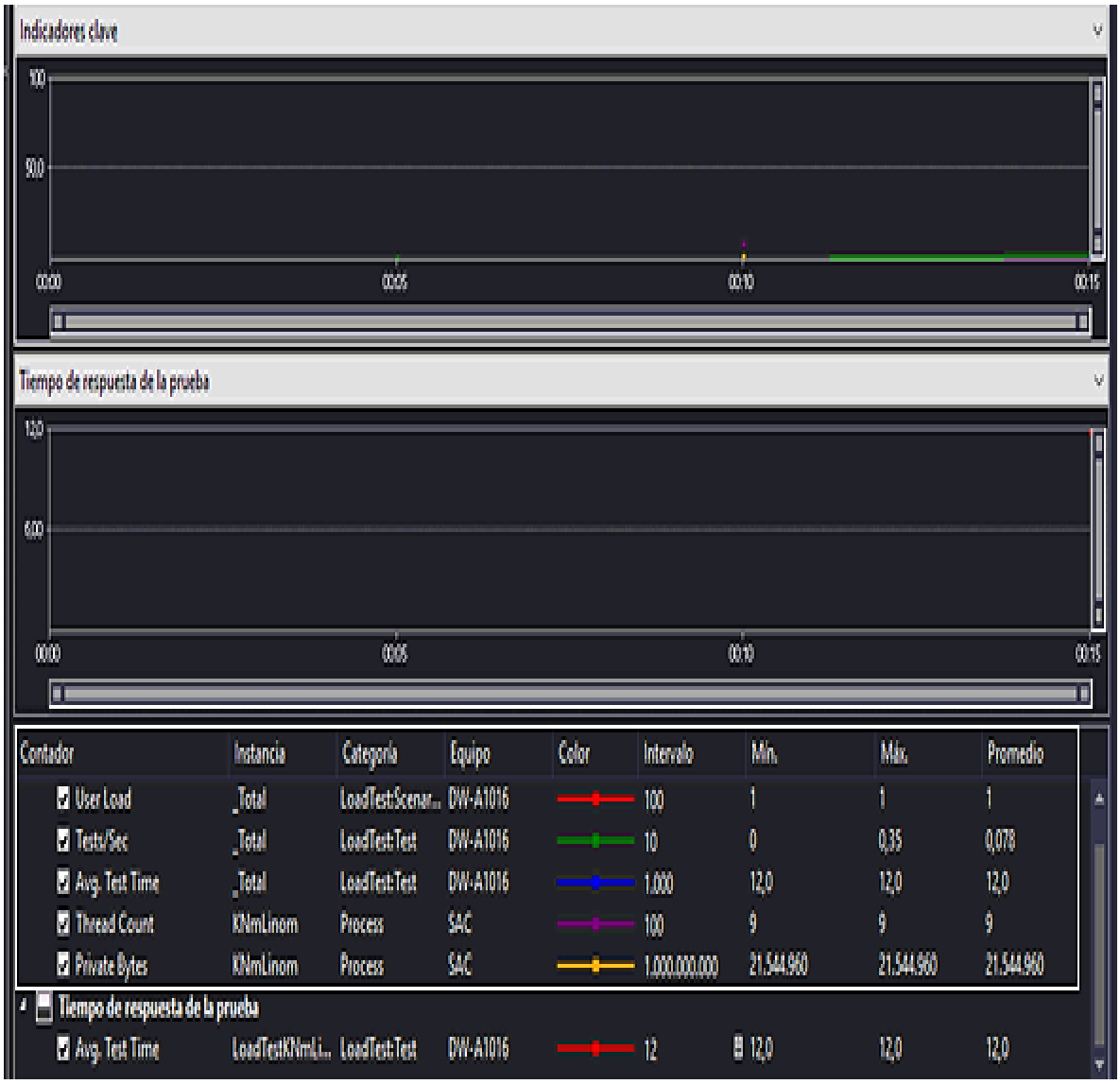


Figura 30. Resultados de prueba de carga con un usuario

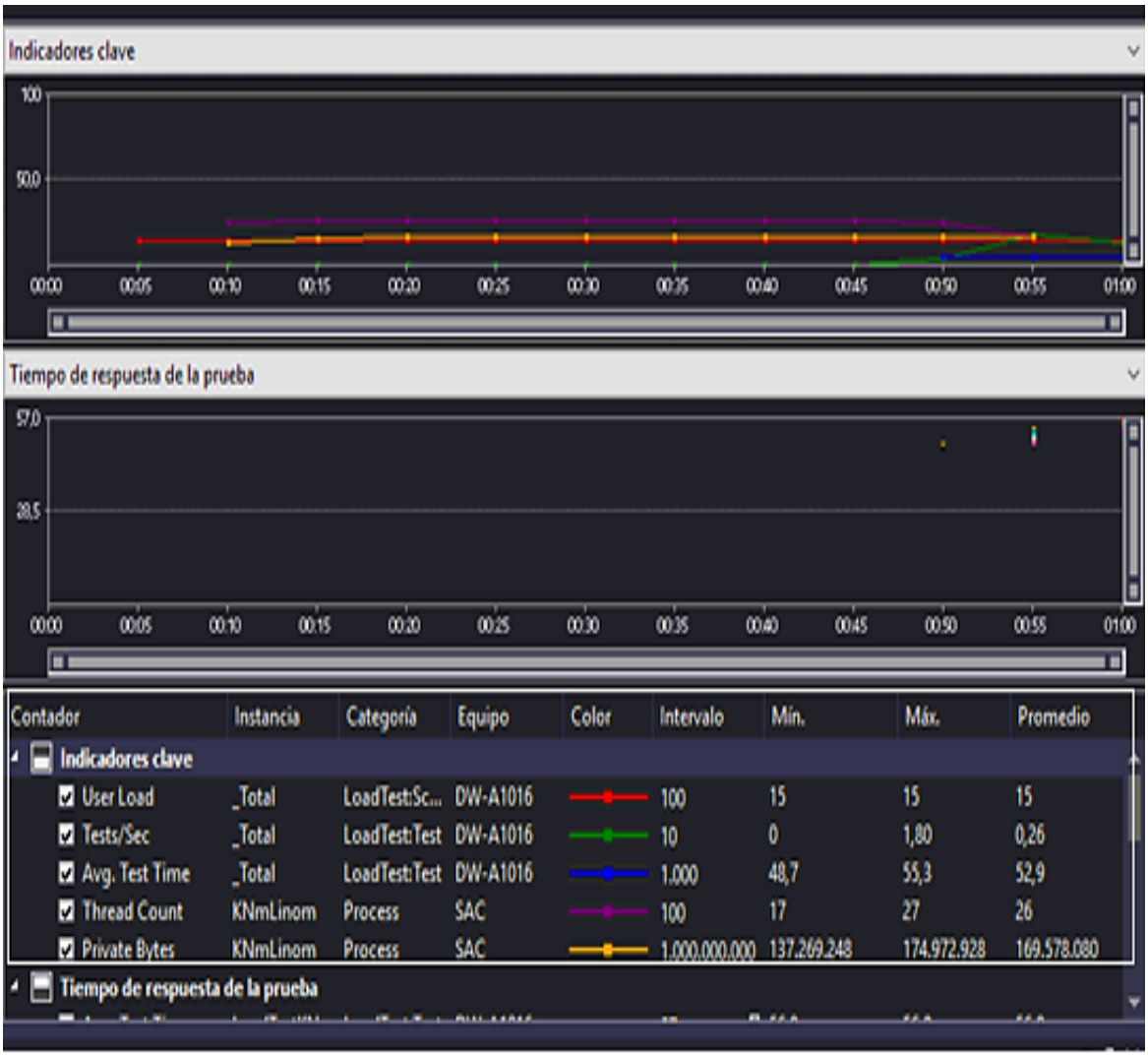


Figura 31. Resultados de prueba de carga con 15 usuarios

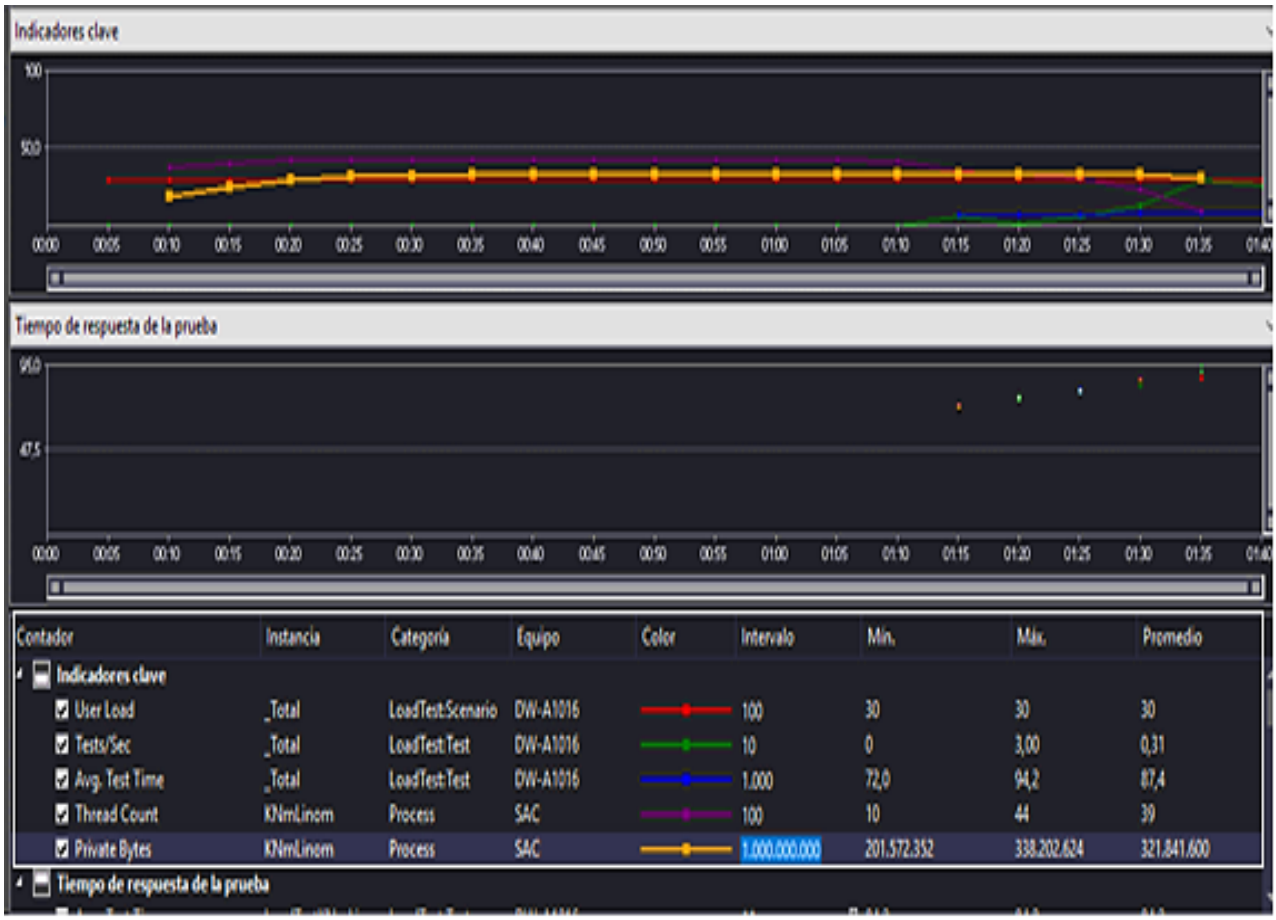


Figura 32. Resultados de prueba de carga con 30 usuarios

Se debe tener presente que la aplicación de estas pruebas no realizara ninguna mejora al software, pero con ellas se podrá tener un informe de rendimiento con el cual se pueden tomar decisiones como realizar seguimiento de la ejecución en la base de datos, mejoras de infraestructura, detección de errores en la programación, entre otros.

Por realizarse una carga de empleados tan pequeña no se recomienda realizar ningun cambio en hardware puesto que los recursos con los que cuenta el servidor no se vieron saturados. Para poder dar este tipo de sugerencias las pruebas deben ser sometidas a escenarios mas reales en los que se puedan variar las configuraciones de las mismas y poder comparar resultados en diferentes escenarios. En cuanto al software la sugerencia que se da es encontrar una mejora a nivel de codigo para que el tiempo de ejecución no se altere de gran manera al tener concurrencia.

Dependiendo de los problemas encontrados se debe realizar un análisis de lo que representarían estos cambios a la compañía, ya que si las pruebas no funcionales

arrojan resultados de que los tiempos de ejecución aumentan de una versión del programa a otra se debe solucionar inmediatamente ya que al ser liberada esta versión ocasionara problemas en todos los clientes, pero si los problemas son propios de infraestructura del cliente se le debe dar un informe y que el mismo decida si va a realizar o no los cambios ya que allí solo ellos pueden tomar la decisión y la empresa solo hace el acompañamiento con la asesoría de lo que se debería mejorar.

La priorización de los cambios estará ligada con los resultados obtenidos, serán de prioridad del lado de la empresa si la demora de ejecución o error en la inserción de datos sea propiamente del programa, pero si esto es de parte del cliente será la prioridad que ellos mismos le den.

CONCLUSIONES

Se logró identificar el funcionamiento del proceso KNmLinom del software KACTUS-HCM con la ejecución de las pruebas no funcionales denotando que el mismo al ser concurrido, tiene un aumento en sus tiempos de ejecución, pero continúa funcionando correctamente sin generar cruces de inserción de datos. Los resultados de los procesos tienden a depender de las condiciones de infraestructura en donde se realicen por esto no siempre tendrán el mismo comportamiento así que es recomendable que en el momento de realizar la instalación del producto en otras compañías se establezcan unas condiciones previas de dimensionamiento para que no se generen errores o colapsos en los procesos.

Mediante el análisis de las metodologías que se pueden aplicar a las pruebas no funcionales se determinó que la metodología en cascada se ajustaba más a la necesidad de la empresa, y esta se tomó como base para determinar las etapas que son: la definición de los requerimiento funcionales y no funcionales, el diseño de los casos de uso para la integración con las pruebas, el desarrollo de las pruebas unitarias, la asociación de estas pruebas a la prueba de carga, la ejecución de las mismas y el análisis de resultados mediante informes que comparan las métricas.

Esta metodología permitirá continuar con la ejecución de las pruebas de carga en otros procesos críticos (Liquidadores) puesto que, en cada uno de ellos con diferentes parámetros estructurales, se dará seguimiento al paso a paso propuesto por la misma. También podrá ser aplicada bajo otras condiciones de hardware o software modificando diferentes parámetros de ejecución como lo son: la carga máxima de usuarios y la cantidad de empleados a liquidar por usuario.

Finalmente, esta metodología permite a la compañía Digital Ware ofrecer un producto con mejores estándares de calidad ya que con esto se estará en constante evaluación de procesos críticos donde no solo aseguren la funcionalidad, sino que también se confirme que los procesos no se vean afectados en tiempos de respuesta.

REFERENCIAS

[1] Digital Ware Tecnology that changes People´s Lives. [En línea].Bogotá DC. [Consultado: 19 de Septiembre de 2018]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

[4] Digital Ware Tecnology that changes People´s Lives. [En línea].Bogotá DC.Reglamento interno [Consultado: 09 de Noviembre de 2018]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

[5] ÁLVAREZ HERNÁNDEZ, Jorge Eduardo y PONCE CRUZ, Francisco Javier. Metodología para el desarrollo de pruebas de software basada e métricas de funcionalidad y rendimiento [en línea].Trabajo Terminal 2013-B061.México, DF. Instituto Politécnico Nacional Escuela Superior de Computo. Departamento de Formación Integral e Institucional. Comisión Académica de Trabajo Terminal, 2015. p. 36. [Consultado 17 Agosto de 2018]. Disponible en Internet: https://goo.gl/hhLJgK

[6] ECHEVERRÍA PEREZ, Devis y AVELLA PAUMIER, Ariannis. Testing como Práctica para Evaluar la Eficiencia en Aplicaciones Web [en línea]. En: Centro Nacional de Calidad de Software: Revista Latinoamericana de Software.2014, vol.2 , no. 5. p. 307-308. [Consultado: 17 Agosto de 2018]. Disponible en Internet: file:///E:/255-Texto%20del%20art%C3%83_culo-551-1-10-20141027.pdf

[7] ACOSTA MESA, Ingrid Ximena, NIETO GUEVARA, Erika Alejandra y BARAHONA RODRÍGUEZ, Cesar Yesid. Metodología para la evaluación de calidad de los productos software de la Universidad de Cundinamarca [en línea]. En: ENGI: Revista Electrónica de la Facultad de Ingeniería. Universidad de Cundinamarca. 2015, vol. 3, no 2. p. 15-16. [Consultado 18 de Agosto de 2018]. Disponible en Internet: file:///E:/157-412-1-SM%20(2).pdf

[8] PARADA GELVÉZ, Hugo Alexer. Contribución a la gestión de los procesos de pruebas de software y servicios [En línea]. Tesis doctoral Ingeniería de Sistemas

Telemáticos. Madrid, España. Universidad Politécnica de Madrid. Departamento de ingeniería de Sistemas Informáticos. Escuela Técnica Superior de Ingenieros de Telecomunicación ,2010. p. 82.[Consultado 18 Agosto de 2018] Disponible en Internet : http://oa.upm.es/4019/1/HUGO_ALEXER_PARADA_GELVEZ.pdf

[9] KANER, Cem. Actualización de algunos conceptos básicos en pruebas de software [En Línea].Cem Kaner, JD, Ph.D. Rockledge FL, Estados Unidos. (21 de Noviembre de 2006), p. 1. [Consultado: 24 de Noviembre de 2018]. Disponible en internet: http://kaner.com/?p=30

[10] International Software Testing Qualifications Board. Probador Certificado Programa de estudio de nivel básico. [Medio Electrónico]. Versión 2011. Copyright © ISTQB®. 31 de Marzo de 2011. p.11. [Consultado 24 de Noviembre de 2018]. Disponible en: http://www.sstqb.es/ficheros/sstqb_file95-637831.pdf

[11] SÁNCHEZ PEÑO, José Manuel. Pruebas de Software fundamentos y técnicas [En Línea]. Madrid, España. Universidad Politécnica de Madrid. Departamento de Teoría de la Señal y Comunicaciones, 2015. p. 22. [Consultado: 26 de Noviembre de 2018]. Disponible en: http://oa.upm.es/40012/1/PFC_JOSE_MANUEL_SANCHEZ_PENO_3.pdf

[12] T. Müller, Libro Programa de Estudio de Nivel Básico para Probador Certificado, istqb, Version 2013, p. 14. Disponible en: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

[13] ZAMORA HERNÁNDEZ, Jorge. Análisis de los procesos de verificación y validación en las organizaciones software [En Línea]. Proyecto fin de carrera Ingeniería del Sotfware.Madrid, España. Universidad Carlos III de Madrid. Departamento de Ingeniería del Software, 2011. p.91. [Consultado: 20 de Diciembre de 2018]. Disponible en Internet: https://e-archivo.uc3m.es/bitstream/handle/10016/12880/PFC_Jorge_Zamora_Hernandez.pdf

[14] TERRERA, Gustavo. Testing de caja blanca–parte I [En Línea].TestingBaires.3 de Junio del 2014, p.1. [Consultado: 20 de Diciembre de 2018]. Disponible en Internet: https://testingbaires.com/2014/01/03/testing-de-caja-blanca-parte/

[15] SÁNCHEZ PEÑO, Pruebas de Software fundamentos y técnicas, Op. Cit., p. 23.

[16] OCAMPO ACOSTA, Alejandro y CORREA TAPASCO, Luisa Marcela. Impacto de las pruebas no funcionales en la medición de la calidad del producto software desarrollado [En Línea].Tesis de Ingeniería de sistemas y computación. Pereira, Colombia. Universidad tecnológica de Pereira. Facultad de ingenierías eléctrica, electrónica, física y Ciencias de la computacion.Programa de ingeniería de sistemas y computación 2011. p.16. [Consultado: 21 de Diciembre de 2018].
Disponible en Internet: https://n9.cl/cVGb

[17] MICROFOCUS. Informe oficial pruebas de carga la clave para garantizar la ejecución ininterrumpida de las aplicaciones empresariales [En Línea].Microfocus.Colombia. (02 de febrero de 2017), p.3. [Consultado: 13 de Enero de 2019]. Disponible en Internet: https://www.microfocus.com/es-es/media/white-paper/load_testing_wp_es.pdf

[19] 4R SOLUCIONES. Pruebas de stress sobre aplicaciones web [En Línea].4r Productora Digital y Software Factory. Argentina 03 de Octubre de 2012. p.1. [Consultado: 13 de Enero de 2019]. Disponible en: http://www.4rsoluciones.com/blog/pruebas-de-stress-sobre-aplicaciones-web-2/

[20] OCAMPO ACOSTA, Alejandro y CORREA TAPASCO, Luisa Marcela. Impacto de las pruebas no funcionales en la medición de la calidad del producto software desarrollado, Op. Cit., p. 47.

[21] SPARKS, Geoffrey. El Modelo de Casos de Uso. Introducción al modelado de sistemas de software usando el Lenguaje Unificado de Modelado (UML) [En Línea]. Solus - Craftware Consultores Ltda. p.3[Consultado: 12 de Marzo de 2019]. Disponible en internet: http://www.sparxsystems.com.ar/downloads/whitepapers/El_Modelo_de_Casos_de_Uso.pdf

[22] RODRÌGUEZ FLORIDO, Miguel Ángel. Introducción a la Programación Concurrente. [En Línea]. Universidad de Las Palmas de Gran Canaria. p.4. [Consultado: 12 de Marzo de 2019]. Disponible en internet: https://www2.ulpgc.es/hege/almacen/download/20/20233/tema1.pdf

[23] HERMANN, Eliza. Pruebas de Software: Herramientas: Pruebas Unitarias. [Presentación PowerPoint].Madrid, España.Okybele.2011, p. 4. Disponible en Internet: http://www.kybele.etsii.urjc.es/docencia/IS_LADE/2011-2012/Material/Pruebas%20de%20SoftwareHerraminetas.pdf

[24] GUTIÉRREZ DÍAZ, Alejandro. Base de datos. [En línea]. Centro Cultural Itaca S.C.2016.Distrito Federal México. p. 10. MIS 308. [Consultado: 18 de Marzo de 2019]. Disponible en Internet: https://www.aiu.edu/cursos/base%20de%20datos/pdf%20leccion%201/lecci%C3%B3n%201.pdf

[25] CALÁD ÁLVAREZ, Alejandro y RUÍZ CALLE, Juan David. Metodologías de testing de software y su aplicación en el centro de informática de la Universidad EAFIT [en línea]**.** Proyecto de grado Ingeniería de Sistemas. Medellín, Colombia. Universidad EAFIT**.** Departamento de Ingeniería de Sistemas**,** 2009**.** p. 48 [Consultado**:** 30 de Diciembre de 2018]**.**Disponible en Internet**:** https://repository.eafit.edu.co/xmlui/bitstream/handle/10784/2744/RuizCalle_JuanDavid_2009.pdf?sequence=1

[26] Microsoft. Herramientas de prueba en Visual Studio [En Línea].Estados Unidos.(15 de Marzo de 2018), párr.1. [Consultado 30 de Diciembre de 2018].Disponible en Internet: https://docs.microsoft.com/en-us/visualstudio/test/improve-code-quality?view=vs-2017

[27] Digital Ware Tecnology that changes People´s Lives. [En línea].Bogotá DC. [Consultado: 1 de Enero de 2019]. Disponible en Internet: http://www.digitalware.com.co/index.php/acerca-de-digital-ware/nosotros-

[28] Microsoft. SQL Server Management Studio (SSMS) [En Línea].Estados Unidos. (15 de Octubre de 2018), párr.1. [Consultado 1 de Enero de 2019].Disponible en Internet: https://docs.microsoft.com/en-us/visualstudio/test/improve-code-quality?view=vs-2017

[29] Disponible en: https://okhosting.com/blog/el-ciclo-de-vida-del-software/

[30] Disponible en: https://docs.microsoft.com/en-us/visualstudio/test/analyze-load-test-results-in-the-graphs-view?view=vs-2017

[31] Disponible en: https://docs.microsoft.com/en-us/visualstudio/test/analyze-load-test-results-in-the-graphs-view?view=vs-2017

[32] Pressman, Roger, Ingeniería de Software 3ª Ed., McGraw Hill, 1993.

[33] LÓPEZ ECHEVERRY Ana María, CABRERA Cesar y VALENCIA AYALA, Luz Estela. Introducción a la calidad de software [En Línea].(Septiembre) ,2008, no.39, p. 328.ISSN 0122-1701 [Consultado : 10 de Enero de 2019]. Disponible en Internet: file:///E:/Dialnet-IntroduccionALaCalidadDeSoftware-4745899.pdf

[34] ISO25000 [En Línea]. [Consultado: 12 de enero de 2019]. Disponible en Internet: https://iso25000.com/index.php/normas-iso-25000/iso-25010?limit=3&start=3

[35] CRISTIÁ, Maximiliano.Introducción a la Ingenierıa de Requerimientos. [En Línea]. Ingeniería de Software Facultad de Ciencias Exactas, Ingenier´ıa y Agrimensura.Universidad Nacional de Rosario.2011. p.4-5. [Consultado: 20 de marzo de 2019]. Disponible en Internet:https://www.fceia.unr.edu.ar/~mcristia/publicaciones/ingreq-a.pdf