UNIVERSIDAD CATÓLICA LOS ÁNGELES CHIMBOTE

FACULTAD DE INGENIERÍA

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

IMPLEMENTACIÓN DE COMPONENTES DE SOFTWARE PARA MEJORAR EL DESARROLLO DE LOS APLICATIVOS WEB EN EL MINISTERIO DE ECONOMÍA Y FINANZAS – LIMA; 2020.

TESIS PARA OPTAR EL GRADO ACADÉMICO DE MAESTRO EN INGENIERÍA DE SISTEMAS CON MENCIÓN EN TECNOLOGÍA DE INFORMACIÓN Y COMUNICACIÓN

AUTOR

TOMÁS PASCUAL, GUIDO JHONNY
ORCID: 0000-0002-3807-8116

ASESOR

GARCÍA CÓRDOVA, EDY JAVIER
ORCID: 0000-0001-5644-4776

CHIMBOTE – PERÚ

2021

Equipo de Trabajo

AUTOR

Tomás Pascual, Guido Jhonny

ORCID: 0000-0002-3807-8116

Universidad Católica Los Ángeles de Chimbote, Estudiante de Posgrado, Chimbote, Perú

ASESOR

García Córdova, Edy Javier

ORCID: 0000-0001-5644-4776

Universidad Católica Los Ángeles de Chimbote, Facultad de Ingeniería, Escuela Profesional de Ingeniería de Sistemas, Piura, Perú

JURADO

Sullón Chinga, Jennifer Denisse

ORCID: 0000-0003-4363-0590

Sernaqué Barrantes, Marleny

ORCID: 0000-0002-5483-4997

Coronado Zuloeta, Oswaldo Gabiel

ORCID: 0000-0002-0708-2286

Hoja de firma del jurado y asesor

Sullón Chinga, Jennifer Denisse
Presidente

Sernaqué Barrantes, Marleny
Miembro

Coronado Zuloeta, Oswaldo Gabiel
Miembro

García Córdova, Edy Javier
Asesor

Agradecimiento

En primer lugar, agradecer a Dios quien hace posible que cada cosa exista.

Le doy gracias a mis padres, quienes dieron lo mejor de sí inculcándome valores y poder ser mejor cada día.

A mi esposa, mi hija y mi nieto, que son la razón de ser de todos mis esfuerzos y sacrificios.

Gracias a todas las personas que con sus aportes ayudaron a la elaboración de esta tesis.

Guido

Dedicatoria

Dedico el presente trabajo a mi nieto Dylan, quien vino al mundo a renovar mi alegría.

Guido

Resumen

La presente tesis se realizó en la Maestría en Ingeniería de Sistemas con mención en Tecnología de Información y Comunicación de la Escuela de Ingeniería de Sistemas de la Universidad Católica Los Ángeles de Chimbote, tomando como problemática la complejidad y tiempo de desarrollo de las aplicaciones web del Ministerio de Economía y Finanzas-Lima. Frente a esta situación se propuso como objetivo general Implementar Componente de Software para Mejorar el Desarrollo de los Aplicativos Web en dicha institución. El tipo de investigación es cuantitativa, de nivel aplicativo y con un diseño pre-experimental. La población está conformada por 6 desarrolladores del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima evaluando 12 procesos de búsqueda de información de las opciones del aplicativo SIGAWEB. La muestra ocupará a toda la población, por lo que la denominaremos una población muestral. Se usó como técnica para la recolección de datos la observación y como instrumento las notas de campo. Así mismo, para la implementación del componente de software se utilizó como instrumentos: Eclipse IDE como herramienta de desarrollo, Framework ZK para la capa de presentación y JAVA como lenguaje de programación. Llevar a cabo el presente trabajo dotó a la institución de una herramienta de software que simplificará el proceso de desarrollo de las aplicaciones web, reduciendo las líneas de código y los tiempos de entrega de los productos.

Palabras clave: Componente de software, Aplicativo web, Framework ZK, Frontend, Backend.

Abstract

This thesis was carried out in the Master's Degree in Systems Engineering with a mention in Information and Communication Technology of the School of Systems Engineering of The Angels de Chimbote Catholic University, taking as a problem the complexity and development time of the web applications of the Ministry of Economy and Finance-Lima. Faced with this situation, the general objective of Implementing a Software Component to Improve the Development of Web Applications in said institution was proposed. The type of research is quantitative, application level and with a pre-experimental design. The population is made up of 6 developers of the SIGAWEB application of the Ministry of Economy and Finance - Lima evaluating 12 information search processes of the SIGAWEB application options. The sample will occupy the entire population, so we will call it a sample population. Observation was used as a technique for data collection and field notes as an instrument. Likewise, for the implementation of the software component, the following instruments were used: Eclipse IDE as a development tool, Framework ZK for the presentation layer and JAVA as a programming language. Carrying out this work provided the institution with a software tool that will simplify the web application development process, reducing lines of code and product delivery times.

Keywords: Software component, Web application, ZK Framework, Frontend, Backend.

Contenido

Índice de Gráficos

Índice de Tablas

I. Introducción

El Perú es un país con alto potencial para el desarrollo de software, pues requiere cubrir necesidades de aplicaciones informáticas en entidades públicas y privadas. Para ello, es necesario implementar componentes de software reutilizables, que reduzcan los tiempos de desarrollo y abarate los costos.

El Ministerio de Economía y Finanzas no es ajeno a esta realidad pues, a través de la Oficina General de Tecnologías de la Información (1), desarrolla aplicaciones informáticas en plataforma cliente/servidor y web para uso interno y otras instituciones del país, como son el SIAF y el SIGA. Sin embargo, dichas aplicaciones llevan varios años de haber sido implementadas y han sufrido múltiples cambios a través del tiempo. Esto ha conducido a incrementar la complejidad del código, siendo difícil su mantenimiento y los tiempos de entrega de las nuevas versiones son cortos.

Adaptando determinados procesos del SIGA a un modelo de desarrollo de software basado en componentes es posible optimizar el desarrollo del aplicativo en su versión web. Esto va a permitir reducir significativamente la complejidad y tiempo de desarrollo; así mismo, aliviar la sobrecarga laboral.

De lo manifestado anteriormente se planteó el siguiente problema de investigación: ¿De qué manera la implementación de componentes de software permite mejorar el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020?

El objetivo planteado en la presente investigación fue implementar un Componente de Software para Mejorar el Desarrollo de los Aplicativos Web en el Ministerio de Economía y Finanzas – Lima; 2020.

En función al objetivo general se plantearon los siguientes objetivos específicos:

1. Identificar procesos de desarrollo de software a cambiar a modelo basado en componentes.
2. Medir el tiempo de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas – Lima; 2020 antes de implementar el componente de software.
3. Determinar la complejidad de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas – Lima; 2020 antes de implementar el componente de software.
4. Implementar componente de software para reducir el tiempo y complejidad del Desarrollo de los Aplicativos Web del Ministerio de Economía y Finanzas – Lima; 2020.
5. Comprobar y contrastar el tiempo y complejidad de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas - Lima; 2020 después de implementar el componente de software.

La investigación se justifica académicamente porque permite comprobar las habilidades y conocimientos adquiridos en el desarrollo de la Maestría en Ingeniería de Sistemas en la Universidad Católica los Ángeles de Chimbote.

Desde el punto de vista operativo se justifica porque la implementación del componente de software dota a la institución de una herramienta que simplifica el proceso de desarrollo de las aplicaciones web, reduciendo las líneas de código y tiempos de entrega de los productos.

Tecnológicamente la investigación se justifica porque el componente de software implementado hace uso de tecnologías como frameworks, lenguajes de programación y otras herramientas tecnológicas disponibles en el mercado, que

como producto final, puede usarse no solo dentro de la institución estudiada, sino en otras instituciones y empresas que usen tecnologías similares.

En cuanto al aspecto económico se justifica porque la implementación de un componente de software especializado, al reducir la complejidad y tiempo de los procesos de desarrollo, permite optimizar las horas-hombre, reduciendo así el costo de los proyectos.

La metodología de la investigación ha sido de tipo cuantitativa, de nivel aplicativo y con un diseño pre-experimental. La población está conformada por 6 desarrolladores del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima evaluando 12 procesos de búsqueda de información de las opciones del aplicativo SIGAWEB. La muestra ocupó a toda la población de seis programadores, por lo que se denomina población muestral. Se usó como técnica para la recolección de datos la observación y como instrumento las notas de campo. Así mismo, para la implementación del componente de software se utilizó como instrumentos: Eclipse IDE como herramienta de desarrollo, Framework ZK para la capa de presentación y JAVA como lenguaje de programación. Los datos obtenidos fueron procesados, codificados e ingresados en el Software de Ofimática Excel de Microsoft 365, el cual también se usó para el análisis de los datos y la obtención de tablas y gráficos de las variables en estudio. Como resultados se aprecia que usando el componente de software desarrollado, hay una disminución significativa en los tiempos de desarrollo, así como en las líneas de código empleadas. El componente de búsqueda de información SigaBandbox permite reducir al 36.2% el tiempo promedio que se demoraría elaborar el mismo componente usando el código original. Así mismo, la complejidad expresada en líneas de código se reduce al 21.7% con respecto al código original.

Para la implementación del Componente de software denominado SigaBandbox, que permite la búsqueda de información, se ha seguido el “Proceso de Solución

Técnica e Integración del Producto" de la Oficina General de Tecnologías de la Información del Ministerio de Economía y Finanzas, el cual está enmarcado en el modelo Capability Maturity Model Integration (CMMI) v1.3., llegando a la conclusión de que la Implementación de componentes de software mejoran el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020;

II. Marco teórico

2.1. Bases teóricas relacionadas con el estudio

2.1.1. Antecedentes

2.1.1.1. Antecedentes a nivel internacional

Hincapié y Pinto (2) en el año 2019, en su tesis para optar el grado de Magister en Ingeniería de Sistemas y Computación, titulado: "Análisis y Prototipado de un Componente de Software de Exploración de Datos, Integrado a la Arquitectura de Visualización utilizando Dashboards", desarrollado en la ciudad de Pereira – Colombia, tiene por objetivo general el Análisis e implementación de un componente de software a través de un prototipo orientado a la exploración de datos integrado a la arquitectura de visualización propuesta en el proyecto denominado "Diseño de una arquitectura por componentes para la visualización de datos utilizando dashboards". El trabajo es de tipo pre-experimental.

Analiza e implementación un componente de software a través de un prototipo orientado a la exploración de datos integrado a la arquitectura de visualización propuesta en el proyecto denominado "Diseño de una arquitectura por componentes para la visualización de datos utilizando dashboards" de Betancurt Obando & Abril Pérez, 2018, concluye que el componente de exploración de datos ayudó a la arquitectura y al usuario obtener un conocimiento más amplio sobre los datos, dando así un aporte para soportar de mejor manera el futuro proceso de toma de decisiones; así mismo, las arquitecturas construidas por componentes ofrecen a los desarrolladores o diseñadores de software posibilidades infinitas a la hora de extender sus funcionalidades e integrar nuevos componentes.

Cazalilla (3) en el año 2017, en su tesis doctoral denominada: "Diseño e implementación de un sistema de control de robots mediante la ingeniería del software basada en componentes. Aplicación a un robot

paralelo de 3DOF", en la ciudad de Valencia – España, tiene por objetivo diseñar controladores para un robot paralelo de tres grados de libertad aplicando los principios de la ingeniería del software basado en componentes. Su trabajo es de tipo experimental. Concluye demostrando la eficacia del desarrollo de controladores robóticos en forma de componentes software independientes y reusables, siguiendo las metodologías de la Ingeniería del Software Basada en Componentes. Además, el hecho de haber escogido Orocos como middleware de control en tiempo real y orientado a componentes, ha permitido la ejecución en paralelo de diversos controladores avanzados, distribuyendo la carga computacional de una forma mucho menos compleja. Por otro lado, al haber diseñado los controladores de forma modular, también ha permitido poder reusar partes comunes a la hora de realizar nuevos controladores.

Silvera y Sayas (4) en el año 2016, en su tesis de grado: "Elaboración de un Componente de Software para el Cálculo de Métricas de Diseño a partir de XMI", en la ciudad de Cartagena de Indias – Colombia, tiene como objetivo general desarrollar un componente reutilizable que permita a los desarrolladores de software realizar cálculo de métricas de diseño a partir de diagramas en formato XMI. Su tipo de investigación es descriptivo, como metodología usó al Proceso Racional Unificado o RUP, dividiendo el proceso de desarrollo del producto en cinco fases, de acuerdo con los objetivos específicos. Como resultado obtuvo un componte para el cálculo de métricas de diseño, que facilita al ingeniero de software el análisis sus proyectos a partir de diagramas de clases representados en XMI, como herramienta de control de calidad del diseño para optimizar y mejorar en el producto software. Concluye que la herramienta desarrollada brinda una ayuda importante al ingeniero de software presenta información del estado general de sus proyectos. Dicho componente hace parte de la tesis doctoral del profesor Martín Monroy Ríos, titulada «Marco de

referencia para la recuperación y análisis de vistas arquitectónicas de comportamiento», que plantea la elaboración de un entorno de ingeniería inversa.

2.1.1.2. Antecedentes a nivel nacional

Palomino (5) en el año 2018, en su tesis: "Componente Software para Controlar las Versiones del Esquema de Base de Datos Relacional, AHREN Contratistas Generales SAC - Ayacucho, 2017", en la ciudad de Ayacucho, tiene por objetivo general: Desarrollar el componente software mediante técnicas e instrumentos, la metodología ágil Scrum, control de versiones y patrón Code First de Entity Framework 6, con la finalidad de controlar las versiones del esquema de base de datos relacional, AHREN Contratistas Generales SAC, 2017. Su tipo de investigación es aplicada, nivel de investigación descriptiva y diseño no experimental. Desarrolla un componente software de control de versiones para administrar eficientemente las versiones del esquema de base de datos relacional en AHREN Contratistas Generales SAC a través de una investigación es aplicada. El componente software, permite controlar las versiones del esquema de bases de datos relacionales de manera eficiente, permitiendo la administración de los cambios y actualizaciones de la aplicación en la base de datos relacional, sin tener la necesidad de la intervención del desarrollador, optimizando el tiempo y la corrección de errores durante el despliegue de la aplicación. Concluye que se logró desarrollar el componente software mediante técnicas e instrumentos como el análisis documental, la observación, metodología ágil scrum, control de versiones y patrón code first de entity framework 6, para controlar las versiones del esquema de base de datos relacional; se logró analizar y diseñar la capa de modelo de datos para configurar las versiones del esquema de base de datos relacional, en base a al código fuente del componente; se logró implementar y probar la capa de contexto para administrar las versiones del esquema de base de datos relacional.

Cáceres (6) en el año 2016, en su tesis de grado denominada: "Componente Software para Mejorar el Acceso a las Bases de Datos Distribuida, Software Middleware, 2016", en la ciudad de Ayacucho, su objetivo general es: Desarrollar un componente software mediante técnicas e instrumentos, el proceso Scrum, la notación UML, sistemas distribuidos, arquitectura orientada a servicios, lenguaje orientado a objetos, base de dato relacional y tecnologías de internet, con la finalidad de mejorar el acceso a las bases de datos distribuidas, software middleware, 2016. Su tipo de investigación es observacional, el nivel de investigación es aplicada, el diseño de la investigación es no experimental, observacional y prospectivo transversal. Desarrolla un componente software mediante técnicas e instrumentos, el proceso Scrum, la notación UML, sistemas distribuidos, arquitectura orientada a servicios, lenguaje orientado a objetos, base de dato relacional y tecnologías de internet, con la finalidad de mejorar el acceso a las bases de datos distribuidas, software middleware, 2016 ,concluye que con el modelo arquitectónico analizado y diseñado, mejora la eficiencia de acceso a las bases de datos distribuidas basado en técnicas de optimización y el principio del software middleware, logrando delegar la responsabilidad a un componente software el acceso de forma transparente y homogénea a las fuentes de datos heterogéneos y distribuidos; así mismo, cumple con la premisa de que la organización, que el nivel de abstracción del componente, permite mantener la autonomía, trasparencia, aminorar el inconveniente de las equivalencias de los tipos de datos y la adaptación al entorno multiplataforma para la manipulación de los datos en diferentes motores relacionales logrando mejorar el acceso a las bases de datos distribuía. Concluye y corrobora que si desarrollamos software usando el componente software denominado Middleware SQL entonces mejoramos la eficiencia de acceso a las bases de datos distribuida.

Aroni (7) en el año 2015, en su tesis de grado denominada: “Desarrollo de un Componente de Software COM+ para Monitoreo y Control de un Equipo Informático bajo la Plataforma .NET”, en la ciudad de Juliaca, su objetivo general es: Diseñar un componente de seguridad COM+ bajo la plataforma .NET reduciendo las fallas en seguridad de la información y el uso de aplicaciones indebidas teniendo un correcto control y monitoreo sobre un equipo informático. El tipo de investigación es aplicativo, el nivel de investigación es descriptiva – explicativa y el diseño de investigación es cuasi-experimental. Concluye que el componente de software de seguridad permite la reutilización del mismo, basándonos en modelos comprobados; así mismo, se desarrollaron clases y librerías reutilizables que servirán como punto de partida para el desarrollo de otros tipos de componentes pudiendo utilizar el marco de trabajo como base para otros proyectos similares.

2.1.1.3. Antecedentes a nivel regional

López (8) en el año 2018, en su tesis de grado denominada: “Diseño e Implementación de un Componente en la Plataforma .NET bajo la Metodología SCRUM para la Creación y Modificación de Planos Mediante la Teoría de Grafos”, en la ciudad de Lima, su objetivo general es: Diseñar e implementar un componente en la plataforma .NET bajo la metodología Scrum para la creación y modificación de planos mediante la teoría de grafos. El tipo de investigación es descriptiva y aplicada y el diseño es documental por objetivos. Concluye que: Primero, demuestra que el marco de trabajo Scrum puede adaptarse para todo tipo de proyectos, no solo de desarrollo tradicional, con las capas tradicionales de una aplicación de software, sino también de un componente como con una arquitectura diferente y con especificaciones bastante singulares como el presente; segundo, logra desarrollar un componente para la creación y modificación de

objetos 2D y 3D en AutoCAD usando la metodología Scrum, dando como resultado la modificación de planos 2D para la agregación de instalaciones eléctricas; y tercero, realizando los casos de prueba pudo concluir que el componente reduce significativamente el tiempo que toma la creación de planos eléctricos, dichos tiempos fueron reducidos de aproximadamente de diez minutos a una hora que le toma a un operario humano, de acuerdo con su pericia, reducirlo a menos de veinte segundos.

Castro y Exebio (9) en el 2019, en su trabajo de investigación de grado denominada: "Métodos y Herramientas de Trabajo para Arquitecturas Basado en Componentes de Desarrollo de Software: Una revisión Sistemática de la Literatura", su tipo de investigación es descriptiva no experimental. Tiene por objetivo principal Revisar métodos y herramientas de trabajo para Arquitecturas Basado en Componentes de Desarrollo de Software. Concluyen que a través una revisión sistemática de la literatura realizada a 18 artículos académicos encontrados en 3 bases de datos indexadas de gran relevancia en el ámbito académico y científico y análisis bibliométrico de clasificación de los estudios por año de publicación, hubo un incremento en el año 2008, luego en el año 2016 y 2017 en el interés constante de la optimización de un software basado en componentes, así mismo pudieron notar que la mayor cantidad de artículos fueron en idioma inglés. Finalmente, se espera que, por la arquitectura de componentes, surjan temas de interés sobre la mantenibilidad y la gestión de componentes, temas que serán de mucha importancia para poder tener un mejor control de estos componentes a la hora de implementarlos con el fin de hacerlo comercial.

Rojas (10) en el año 2018, en su tesis de grado denominada: "Póliza Electrónica usando Componente Backend para Reducir Costos de

Despachos en Rímac Seguros y Reaseguros", en la ciudad de Lima, su objetivo general es: Implementar la Póliza Electrónica usando componente Backend para reducir costos de despachos en Rímac Seguros y Reaseguros, el tipo de investigación es aplicada. Describe el desarrollo de la Póliza Electrónica para los productos de salud en Rímac Seguros y Reaseguros, la cual surgió por la necesidad de agilizar el proceso de despacho físico de las pólizas emitidas hacia los clientes/corredores, que ocasionaban gastos de mensajería economato, alquiler de impresoras, servicios de impresión, compaginación y archivo. Para suplir esta necesidad, desarrolló un componente Backend que contribuye a implementar un modelo de entrega electrónica en la compañía. El componente, también, almacena los documentos legalizados en un repositorio digital, para que puedan ser remitidas digitalmente a los clientes/corredores. Finalmente, el componente puede remitir los links de los documentos de la póliza mediante un envió electrónico al cliente/corredor, usando un servicio de envío electrónico que cuente con las certificaciones y regulaciones que garanticen la validez legal de los documentos enviados. El componente Backend desarrollado automatizó el despacho de pólizas de los productos EPS y AM con éxito, logrando reducir gastos de despacho físico de documentos, acortar tiempos de entrega y así lograr la reducción de reclamos de los clientes/corredores por demora o no entrega de documentos. Concluye que el desarrollo del componente backend, garantiza las políticas y procedimientos de ventas y servicios existentes en la compañía; así mismo, realizó las pruebas del componente de manera satisfactoria, consiguiendo la aprobación del área usuaria para la puesta en producción.

Carbajal (11) en el año 2013, en su tesis de grado denominada: "Construcción de un Componente de Software para la Búsqueda del Camino más Corto y el Control de Movimiento en un Videojuego de Estrategia en Tiempo Real", en la ciudad de Lima, su objetivo general

es: Realizar el análisis, diseño e implementación de un componente de búsqueda de caminos y control de movimiento, para implementar un mecanismo de navegación autónoma de los personajes en un videojuego de estrategia en tiempo real en dos dimensiones. El tipo de investigación es aplicada con diseño experimental. Concluye que se estableció la importancia de describir sistemáticamente las características y funcionalidades del componente desarrollado, pues gracias a esta, se pudo validar que cada uno de los resultados esperados cumpla con las principales características de un videojuego RTS; así mismo, demostró con éxito el acoplamiento del componente de búsqueda de caminos y control de movimiento con los componentes encargados de implementar la lógica y gestionar los gráficos del videojuego, los cuales fueron desarrollados en otros proyectos, Por último, utilizó dicha adaptación en la construcción del videojuego ”1814, La Rebelión del Cuzco”.

2.2. Marco teórico

2.2.1. Información del: Ministerio de Economía y Finanzas

2.2.1.1. Información general

El Ministerio de Economía y Finanzas es un organismo del Poder Ejecutivo, cuya organización, competencia y funcionamiento está regido por el Decreto Legislativo Nº 183 y sus modificatorias. Está encargado de planear, dirigir y controlar los asuntos relativos a presupuesto, tesorería, endeudamiento, contabilidad, política fiscal, inversión pública y política económica y social. Asimismo, diseña, establece, ejecuta y supervisa la política nacional y sectorial de su competencia asumiendo la rectoría de ella.

El actual ministro de economía es el Sr. Waldo Mendoza Bellido. El Sr. Waldo Mendoza Bellido juró como ministro de Economía y Finanzas

del Consejo de Ministros el 18 de noviembre de 2020. Antes de asumir el cargo se venía desempeñando como presidente del Consejo Fiscal, además de ser miembro del Consejo Consultivo de la Presidencia del Poder Judicial, y del Consejo Directivo de la Superintendencia Nacional de Educación Superior Universitaria (SUNEDU) (1).

2.2.1.2. Visión

Sector que promueve el crecimiento económico sostenido, que contribuye en la mejora de la calidad de vida de los peruanos, garantizando transparencia y responsabilidad en la política fiscal, enmarcado en la estabilidad macroeconómica.

2.2.1.3. Misión

Mantener la armonía de la política económica y financiera, por medio de la transparencia y responsabilidad fiscal, fomentando al crecimiento económico del país de manera sostenida.

2.2.1.4. Objetivos estratégicos institucionales

- Velar por el equilibrio fiscal y su sostenibilidad.
- Incrementar los ingresos públicos y su estabilidad.
- Conseguir mejor apertura económica del mercado de bienes y servicios.
- Ampliar la cobertura de los mercados financieros y previsional privado.
- Reducir las brechas de infraestructura social y productiva reactivando la inversión en ellas.

- Velar por la calidad del gasto público del gobierno en todos sus niveles.
- Transparentar la rendición de cuentas en el sector público.
- Modernizar la gestión del Ministerio como institución.

2.2.1.5. Organización

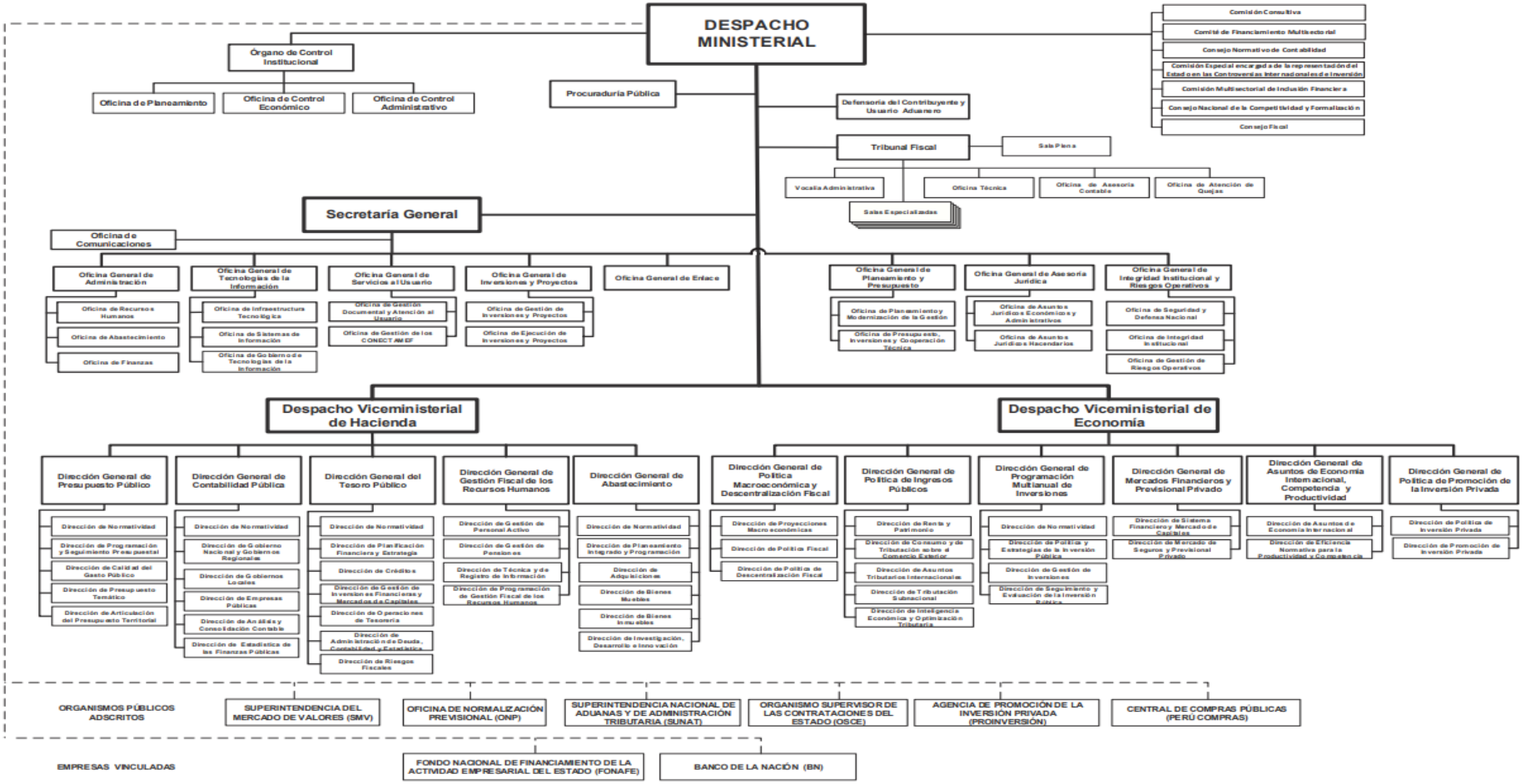
- **Alta Dirección:** se encarga de conducir y tomar decisiones en el Ministerio de Economía y Finanzas. Está formada por: Despacho Ministerial, Despacho Viceministerial de Economía, Despacho Viceministerial de Hacienda y Secretaría General.

  Tiene a su disposición un equipo de asesores especializados que se encargan de dar un juicio técnico en los temas que se les encomiende.

- **Órgano de Control y Defensa Judicial**: conformado por el Órgano de Control Institucional y la Procuraduría Pública.
- **Órgano Resolutivo / Defensoría:** formado por el Tribunal Fiscal y la Defensoría del Contribuyente y Usuario Aduanero.
- **Órganos de Administración Interna:** formada por los Órganos de Asesoramiento y Órganos de Apoyo.
- **Órganos de línea:** formada por los Órganos dependientes del Viceministro de Hacienda y los Órganos que dependen del Viceministro de Economía.

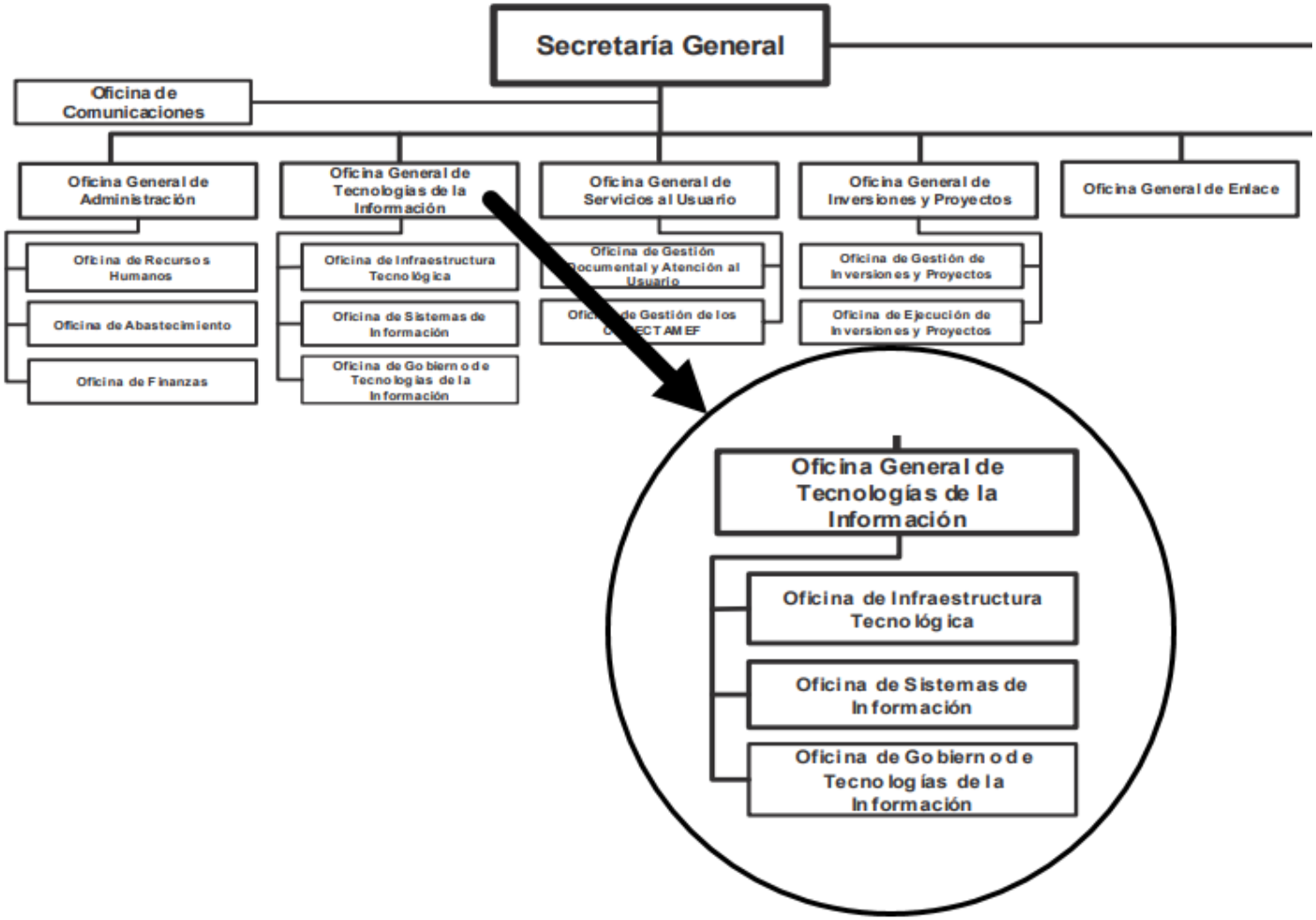
2.2.1.6. Organigrama

Gráfico 1. Organigrama del Ministerio de Economía y Finanzas



Fuente: Ministerio de Economía y Finanzas

Gráfico 2. Organigrama de la Oficina General de Tecnologías de la Información



Fuente: Ministerio de Economía y Finanzas

2.2.1.7. Funciones

Entre las principales funciones del Ministerio de Economía y Finanzas tenemos las siguientes:

a) Formular, planear, dirigir, coordinar, ejecutar, supervisar y evaluar la política económica y financiera nacional y sectorial, aplicable a todos los niveles de gobierno, en el marco de las políticas de Estado;

b) Dictar normas y lineamientos técnicos para la adecuada ejecución y supervisión de la política económica y financiera, la gestión de los recursos públicos, así como para el otorgamiento y reconocimiento de derechos, la fiscalización y la sanción, en materias de su competencia;

c) Ejercer la rectoría de los Sistemas Administrativos de Presupuesto Público, Tesorería, Endeudamiento Público, Contabilidad, Abastecimiento, Programación Multianual y Gestión de Inversiones; así como del Sistema Funcional de Promoción de la Inversión Privada;

d) Formular, proponer, ejecutar y evaluar los lineamientos de política económica y financiera a través del Marco Macroeconómico Multianual (MMM), en consistencia con el marco normativo de la responsabilidad y transparencia fiscal;

e) Evaluar la integralidad y consistencia de las políticas públicas en relación con la política económica y financiera en general, en el corto, mediano y largo plazo;

f) Formular, proponer, ejecutar y evaluar las políticas, normas y lineamientos técnicos en materia de descentralización fiscal, con el objetivo de propiciar la responsabilidad fiscal y la equidad en la transferencia de recursos a los gobiernos subnacionales;

g) Establecer los principios, procesos, normas, procedimientos, técnicas e instrumentos que conducen el proceso presupuestario de las Entidades Públicas;

2.2.1.8. Marco Legal

- Creación del Ministerio de Hacienda por Don José de San Martín, Protector del Perú, mediante decreto del 03 de agosto de 1821.
- Aprobación de la Ley Orgánica del Ministerio de Hacienda mediante Decreto Ley Nº 17521, del 02 de marzo de 1969, que define su estructura y funciones.
- Se modifica el nombre de Ministerio de Hacienda por la de Ministerio de Economía y Finanzas a través del Decreto Ley Nº 17703, del 13 de junio de 1969.
- Se cambia la Estructura Orgánica del Ministerio de Hacienda por medio del Decreto Ley Nº 22196, del 30 de mayo de 1978.
- Se establece la denominación de Ministerio de Economía, Finanzas y Comercio por medio del Decreto Ley Nº 23123, del 09 de julio de 1980, al agregar la Secretaría de Estado de Comercio que pertenecía al Ministerio de Industria, Comercio, Turismo e Integración.
- Se aprueba la Ley Orgánica del Ministerio de Economía y Finanzas y Comercio por medio del Decreto Legislativo N° 183, del 12 de junio de 1981.
- Se modifica la denominación del Ministerio de Economía, Finanzas y Comercio por la de Ministerio de Economía y Finanzas a través del Decreto Legislativo 325, del 30 de enero de 1985.

- Se aprueba una nueva estructura orgánica del Ministerio de Economía y Finanzas, mediante Resolución Ministerial Nº 455-91-EF/43, del 28 de noviembre de 1991, como consecuencia de la aplicación del Decreto Supremo Nº 004-91-PCM que declaró en reorganización todas las Entidades Públicas.
- Se aprueba una nueva estructura orgánica del Ministerio de Economía y Finanzas mediante Resolución Ministerial Nº 223-2011-EF/43, del 01 de abril de 2011.
- A través del Decreto Supremo N° 117-2014-EF, del 23 de mayo de 2014, se aprueba la nueva estructura Orgánica del Ministerio de Economía y Finanzas.
- Decreto Supremo N° 221-2016-EF, del 22 de julio de 2016, que modificó el ROF
- Resolución Ministerial N° 009-2017-EF/41, del 12 de enero de 2017, que modificó el ROF
- Decreto Supremo N° 256-2019-EF, del 08 de agosto de 2019, que aprobó la Sección Primera del ROF
- Resolución Ministerial N° 292-2019-EF/41, del 09 de agosto de 2019, que aprobó la Sección Segunda del ROF
- Resolución Ministerial N° 301-2019-EF/41, del 18 de agosto de 2019, que aprobó el Texto Integrado del ROF

2.2.1.9. OGTI

La Oficina General de Tecnologías de la Información (OGTI) es un órgano de administración interna de la Secretaría General del Ministerio de Economía y Finanzas, responsable de planificar, desarrollar, implantar y gestionar sistemas de información, infraestructura tecnológica y telecomunicaciones que brinden el soporte de las

funciones desarrolladas por los diferentes órganos y unidades orgánicas del Ministerio de Economía y Finanzas –MEF.

Uno de los sistemas informáticos desarrollados por este órgano es el SIGA, en sus versiones Cliente/Servidor y Web.

2.2.2. Sistema Integrado de Gestión Administrativa SIGA

2.2.2.1. Concepto

El Sistema Integrado de gestión Administrativa del MEF, es un sistema informático que contribuye al ordenamiento y simplificación de los procesos administrativos enmarcados en las normas establecidas por los Órganos Rectores de los Sistemas Administrativos del Estado (12), permitiendo:

- Manejo ordenado de los procesos previos al registro en el Sistema SIAF.
- Simplificación de los procesos administrativos.
- Información oportuna de calidad.
- Disponibilidad de información de costos.

2.2.2.2. Datos históricos

Entre los años 1999 y 2000, se reúne un equipo multisectorial conformado por la Presidencia del Consejo de Ministros – PCM, Ministerio de Economía y Finanzas – MEF, Ministerio de Salud, entre otros; con el objetivo de integrar programas de soporte informático de los sistemas administrativos: planillas, abastecimientos, control patrimonial, adquisiciones y contrataciones, tomando la decisión de desarrollar e implantar un sistema informático que incluya estas necesidades; producto de ello nace el Sistema Informático de Gestión

Administrativa – SIGA, implantándose su primera versión en el año 2002 en 14 unidades ejecutoras piloto de diferentes sectores. Posteriormente el número de unidades ejecutoras aumentó significativamente con el ingreso al proceso de más de 100 UEs del Sector Salud a partir del cuarto trimestre del año 2004.

El SIGA fue desarrollado usando la herramienta de desarrollo PowerBuilder, con bases de Datos Oracle y SQL Server; por lo que sus primeras versiones solo era para plataformas cliente servidor bajo entorno Windows. Cabe indicar que este proyecto fue financiado por el Banco Interamericano de Desarrollo – BID.

En el 2012, luego de una versión previa del SIGA para plataforma web desarrollada en el 2005, se empieza un nuevo desarrollo del SIGA Web debido a limitaciones del desarrollo previo, pues sólo era compatible con Internet Explorer 6.0 a 9.0. Este nuevo desarrollo hace uso del framework ZK, está desarrollado en lenguaje Java y soporta múltiples navegadores.

2.2.2.3. Características del SIGA-MEF

Es un sistema de registro único que permite realizar:

- Programación multianual de Cuadro de Necesidades de bienes y servicios y, Plan Anual de Adquisiciones y Contrataciones.
- Registro de procesos de procesos de selección en sus distintas etapas.
- Generación de órdenes de compra y de servicio.
- Atención de pedidos de bienes y servicios

- Registro y contabilización de los movimientos en el almacén.
- Registro y control de bienes patrimoniales.
- Seguimiento a través de consultas y reportes.
- Contiene un catálogo único de bienes y servicios.
- Utiliza motores de base de datos Oracle y SQL Server.
- Contribuye al ordenamiento y simplificación de los procesos de la gestión administrativa, en el marco de las normas vigentes.
- Es un sistema integrado que abarca procesos técnicos de abastecimiento, gestión patrimonial, presupuestos por resultados y tesorería.
- Está estructurado en submódulos.
- Tiene versiones cliente servidor de escritorio y web.
- Está en constante revisión y actualización.
- Cuenta con soporte técnico a través de un área especializada del MEF.

2.2.2.4. Módulos del SIGA-MEF

- Módulo de Logística: SIGA-ML
- Módulo de Patrimonio: SIGA-MP
- Módulo de Presupuesto por Resultados: SIGA-PpR
- Módulo de Tesorería: SIGA-MT
- Módulo de Bienes Corrientes: SIGA-MBC

También cuenta con módulos complementarios: Módulo Administrador, Módulo de Configuración y Módulo Utilitarios.

El SIGA-MEF también cuenta con una versión web denominada SIGA-Web, y solo presentan las principales opciones de la versión cliente/servidor de escritorio y están en permanente implementación con cada versión.

2.2.2.5. SIGA-Web

El SIGA-Web es un aplicativo informático web desarrollado por la Oficina General de Tecnología de la Información – OGTI, del Ministerio de Economía y Finanzas, con el fin de contar con una herramienta en plataforma web del Sistema Integrado de Gestión Administrativa – SIGA, permitiendo así acceder a la información de entidades a través de internet.

La versión actual del SIGA-Web es la v20.06.00. En el presente estudio se utilizará la versión v20.05.00 del año 2020 como base para el experimento a desarrollar.

2.2.2.6. Problemática del SIGA-Web

El SIGA-Web está desarrollado bajo el patrón arquitectónico Modelo-vista-controlador utilizando el lenguaje Java para su desarrollo, con JDK 1.6.033, Spring 3.0.5 para la inyección de dependencias, Hibernate 3.3.1.GA para la persistencia de datos, ZK 6.5.3 para la capa de presentación y Maven 2.3.2 para la gestión de los proyectos. Así mismo, se utiliza el IDE de desarrollo Eclipse Helios y Tomcat 7.0.50 como contenedor de aplicaciones.

El SIGA-Web en su arquitectura actual ha sido desarrollado en el año 2012 y los framework y herramientas de desarrollo que usa no han sido actualizados por lo que puede estar expuesta a riesgos de seguridad pues ya no cuentan con soporte técnico. Por otro lado, los continuos cambios que ha sufrido el aplicativo a través de los años, debido a nuevos requerimientos, cambios y mejoras, ha llevado a incrementar la complejidad del código siendo difícil su mantenimiento. A esto se suma el hecho de que gran parte del código ha sido migrado del aplicativo SIGA Cliente/Servidor, el cual tiene mayor antigüedad y cierta parte de su código ha quedado en desuso y que también ha sido migrado a la plataforma web pues no hubo un análisis detallado de la funcionalidad a migrar.

Otro de los factores que complica el mantenimiento del SIGA-Web, es la reducción del personal por salida voluntaria de los mismo pero que sus plazas no han sido cubiertas por nuevos integrantes. Esto conlleva a la sobrecarga laborar del equipo de desarrollo disponible pues sus responsabilidades y tareas diarias se han visto incrementadas con cada baja presentada, así mismo, la perdida de personal experimentado en el negocio del sistema y desarrollo del mismo.

Debido a los continuos requerimientos y cambios al cual es sometido el SIGA-Web, pues es usado por los diferentes ministerios e instituciones del estado, es necesario sacar un promedio de cinco o seis versiones cada año, lo que significa un promedio de dos o tres meses para todo el ciclo de vida de desarrollo del software de cada versión. Si tomamos solo el proceso de desarrollo, estamos hablando de un mes aproximadamente para terminar un entregable y pasarlo al área de Control de Calidad.

Estos y otros factores no descritos aquí hacen que urja la necesidad de acelerar el proceso de desarrollo para poder cumplir con requerimientos solicitados pues los plazos son cortos y muchas veces no pueden ser ampliados.

2.2.2.7. Optimizando el proceso de desarrollo del SIGA-Web

Mucha de la problemática del SIGA-Web no es de solución rápida, o en su defecto no va a acelerar el proceso de desarrollo. Si se aumentara el número de integrantes del equipo de desarrollo podríamos bajar los tiempos, pero no necesariamente disminuiríamos el nivel de complejidad de desarrollo del mismo.

A través de la experiencia en el desarrollo del aplicativo SIGA-Web se ha podido determinar que ciertas partes del desarrollo de las interfaces gráficas podrían ser rediseñadas para convertirlas en un componente que podría usarse fácilmente en otras opciones. Un de ellas es el proceso de búsqueda de información, consistente en una caja de texto desplegable que presenta en su interior una grilla (filas y columnas) con datos. Sobre la cabecera de cada columna se presenta una caja de texto para filtrar la información presentada.

Gráfico 3. Interfaz de búsqueda de información en el SIGAWEB

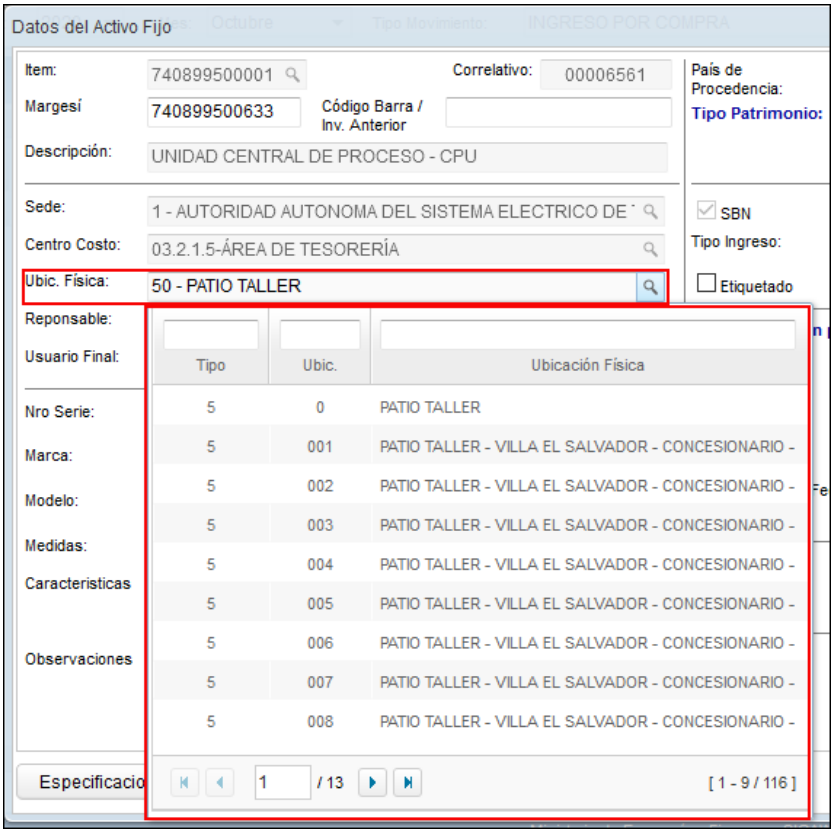
Código	Empleado	C. Costo	Nombre Centro Costo	Tipo Empleado
0000234	SOTOMAYOR MORALES RONNY FRANCISCO	04.2	DIRECCIÓN DE INGENIERÍA	Nombrado
002218	MAGAN MAREOVICH LUIS GUILLERMO	04.1	DIRECCIÓN DE DESARROLLO	Contrato-CAS
002318	NAVARRETE MOSCOSO MILAGROS YULIANA	03.1	UNIDADES DE ASESORAMIENTO	Contrato-CAS
002334	CAJAHUANCA CHAVARRIA SONIA BEATRIZ	04.1	DIRECCIÓN DE DESARROLLO	Contrato-CAS
002335	MAS VILCHEZ PALMIRA MELVA	03.2	UNIDADES DE APOYO	Contrato-CAS
002340	PAUCCAR ALVES WILBER JUAN	04.3	DIRECCIÓN DE ADQUISICIÓN DE PREDIOS Y LIBERACIÓN DE INTERFERENCIAS	Contrato-CAS

Fuente: SIGA-Web

Este proceso de búsqueda es frecuente en las diversas opciones y ventanas de los módulos del aplicativo web, pudiendo presentarse hasta cinco o más veces dentro de una misma ventana, cada uno con su propio código similar.

Por ejemplo, para renderizar en el navegador la siguiente opción de búsqueda se requiere la siguiente cantidad de código: Capa de presentación (vista), 34 líneas de código. En el controlador, 80 líneas de código.

Gráfico 4. Implementación de una interfaz de búsqueda de información



Fuente: SIGA-Web

Gráfico 5. Sección de código fuente frontend de una opción del SIGAWEB

```
<bandbox id="bbxUbicFisica" width="356px" readonly="true" autodrop="true" tooltip="desc" >
    <bandpopup>
        <listbox id="lsbUbicFisica" height="345px"
            emptyMessage="No se encontraron ubicaciones fisicas." width="450px" mold="paging"
            pageSize="10" autopaging="true" onSelect="bbxUbicFisica.close();">
            <listhead>
                <listheader width="80px" align="center">
                    <vbox align="center">
                        <intbox id="txtBusqTipoUbicFisica" maxlength="10" width="60px"/>
                        <label value="Tipo"></label>
                    </vbox>
                </listheader>
                <listheader width="70px" align="center">
                    <vbox align="center">
                        <intbox id="txtBusqUbic" maxlength="10" width="50px"/>
                        <label value="Ubic."></label>
                    </vbox>
                </listheader>
                <listheader width="300px">
                    <vbox align="center">
                        <textbox id="txtBusqUbicFisica" maxlength="10" width="280px"></textbox>
                        <label value="Ubicación Física"></label>
                    </vbox>
                </listheader>
            </listhead>
            <template name="model">
                <listitem value="${each}">
                    <listcell label="${each.tipoUbicac}"></listcell>
                    <listcell label="${each.codUbicac}"></listcell>
                    <listcell tooltiptext="${each.ubicacFisica}"
                        label="${each.ubicacFisica}"></listcell>
                </listitem>
            </template>
        </listbox>
    </bandpopup>
</bandbox>
```

Fuente: SIGA-Web

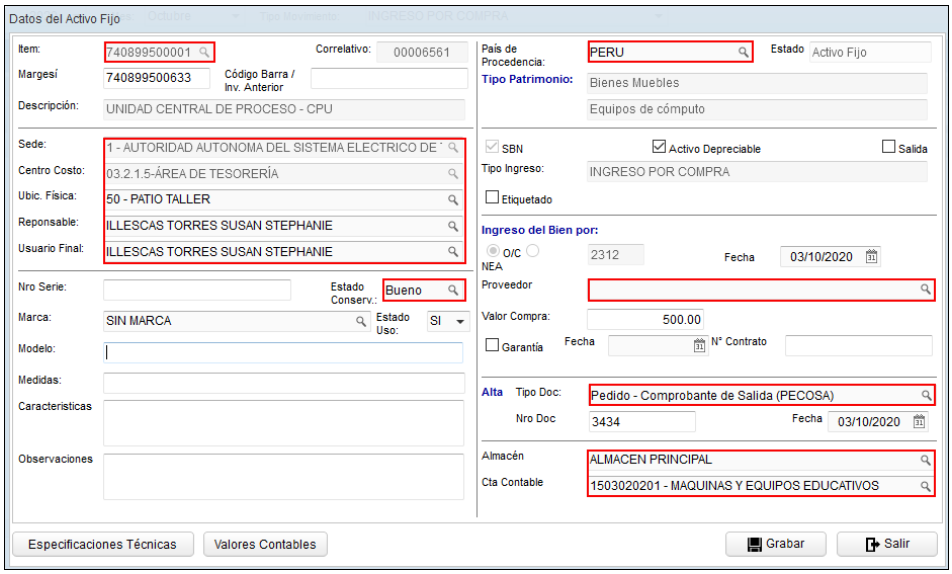
Gráfico 6. Sección de código fuente del backend de una opción del SIGAWEB

```
public void filtrarUbicFisica() {
    List<Map<String, Object>> filtro = new ArrayList<Map<String, Object>>();
    Integer intBusqTipoUbicFisica = this.getTxtBusqTipoUbicFisica().getValue();
    Integer intBusqUbic = this.getTxtBusqUbic().getValue();
    String strBusqUbicFisica = UtilPatrimonio.
        trim(this.getTxtBusqUbicFisica().getValue()).toUpperCase();

    this.getLsbUbicFisica().setModel(new ListModelList<Map<String, Object>>(filtro));

    if (intBusqTipoUbicFisica == null
            && intBusqUbic == null
            && Util.esVacio(strBusqUbicFisica)) {
        filtro = this.getListaUbicFisica();
    } else {

        intBusqTipoUbicFisica = Util.valInteger(intBusqTipoUbicFisica,
                PatrimonioWebConstant.INT_NULL);
        intBusqUbic = Util.valInteger(intBusqUbic,
                PatrimonioWebConstant.INT_NULL);
```

Fuente: SIGA-Web

Cabe decir que a medida que aumenta el número de columnas la complejidad de desarrollo también aumenta. Observe la cantidad de componentes de búsqueda que puede presentar una sola ventana:

Gráfico 7. Búsqueda de información en una ventana típica del SIGAWEB

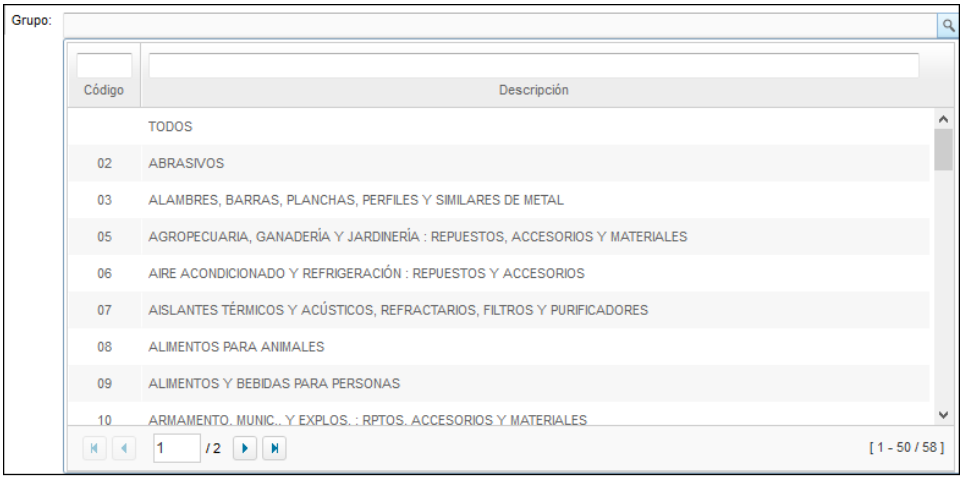


Fuente: SIGA-Web

Con el rediseño del proceso de búsqueda de información en un componente (Componente de búsqueda de información), bastaría con invocarlo y hacer uso de sus puertos de entrada y salida para tener la funcionalidad completa acelerando considerablemente el desarrollo de software. Este es el objeto del presente estudio.

Este sería un prototipo del componente a implementar:

Gráfico 8. Prototipo de componente de búsqueda de información




```
<bandbox id="bnbGrupoItem" width="735px" readonly="true" disabled="false"/>

public void renderBnbGrupoItem() {
    String[] columnas = {"id.grupoBien", "nombreGrupo" };
    String[] etiquetas = { "Código", "Descripción" };
    String[] alineacion = { "center", "left" };
    int[] ancho = {40, 625};
    this.sbbGrupoItem.setColumnas(columnas, etiquetas, ancho, 725);
    this.sbbGrupoItem.setAlinearColumnas(alineacion);
    this.sbbGrupoItem.setAnchoAltoLista(725, 350);
    this.sbbGrupoItem.setTextoOnSelect("id.grupoBien - nombreGrupo");
    this.sbbGrupoItem.setPageSize(50);
    this.sbbGrupoItem.pintar(bnbGrupoItem, this.listaGrupoItem, new Callable(){pul
}
```

Fuente: Propia

2.2.3. Ingeniería del software basado en componentes

La ingeniería de software basada en componentes surgió a finales de los años noventa con la idea de desarrollar aplicaciones que permitan reutilizar componentes de software; esto debido a que muchos diseñadores se sintieron frustrados con el desarrollo orientado a objetos pues, según ellos, no permitía una reutilización más amplia de lo que esperaban. Las clases de cada objeto tenían mucho detalle, eran muy específicas y demasiado acopladas a la aplicación en tiempo de compilación. Esto implicaba conocer detalladamente las clases al usarlas, lo que obligaba a tener el código fuente del componente, dificultando la venta y distribución de los objetos como componentes individuales reutilizables (13).

2.2.3.1. Conceptos

Para Council y Heineman (14) un componente es: “Un elemento de software que se conforma a un modelo de componentes estándar y puede desplegarse y componerse independientemente sin modificación, de acuerdo con un estándar de composición”.

Szyperski (15) dice que: “Un componente de software es una unidad de composición con interfaces especificadas contractualmente y sólo con dependencias de contexto explícitas. Un componente de software puede implementarse de manera independiente y está sujeto a composición por terceras partes”.

Podemos decir también que un componente de software es un bloque de construcción de software, parte modular, desplegable y sustituible de un sistema, que incluye implantación y un conjunto de interfaces.

2.2.3.2. Principios

Para un desarrollo basado en componentes se debe seguir los siguientes principios:

- Los componentes deben ser independientes y sus ejecuciones no deben interferir entre sí. Su implementación debe estar oculta y su cambio no debe afectar las demás partes del sistema.
- Los componentes deben comunicarse por medio de interfaces claramente definidas. Manteniendo dichas interfaces, es posible sustituir un componente por otro con funcionalidades adicionales.

- Las infraestructuras de componentes cuentan con servicios estandarizados que permiten su uso en sistemas de aplicación, reduciendo sustancialmente el código nuevo.

2.2.3.3. Interfaces de un componente

Los componentes tienen dos interfaces:

- Interfaz “proporciona”: precisa los servicios que brinda el componente; es decir, los métodos que puede solicitar el usuario al componente.

Interfaz “requiere”: indica los servicios que ofrecen otros componentes al sistema para que un componente funcione adecuadamente.

Gráfico 9. Interfaces de un Componente

Fuente: Sommerville (13)

2.2.3.4. Procesos

Existen dos tipos de procesos de la ingeniería del software basado en componentes:

- **Desarrollo para reutilización:** Por medio de este proceso se desarrollan componentes o servicios que se usarán en otras aplicaciones.

  La finalidad de este proceso es producir uno o más componentes reutilizables, los componentes que usa son conocidos y el

código fuente está disponible para generalizarlos. Este es el proceso que ocupará el presente estudio.

- **Desarrollo con reutilización:** Si se va a desarrollar aplicaciones nuevas usando componentes y servicios existentes; entonces estamos frente al desarrollo con reutilización.

  En este proceso no se sabe cuáles componentes están disponibles; por tanto, es necesario descubrirlos y el sistema debe diseñarse haciendo uso efectivo de los mismos. El código fuente del componente no está disponible.

  Luego de desarrollar un componente para reutilización, podemos utilizarlos en otros módulos o sistemas de la institución.

2.2.3.5. Componentes de software para la reutilización

En la elaboración de componentes de software de reutilización, debe extenderse y adaptarse componentes específicos de la aplicación y elaborar versiones más genéricas y puedan ser reutilizables. Debe determinarse la probabilidad de que dicho componente sea usado en otros proyectos y si el costo en tiempo y recursos invertidos justificará el hacerlo reutilizable.

Los cambios a un componente, que se pueden hacer, para hacerlo más reutilizable son los siguientes:

- Suprimir los métodos propios de la aplicación.
- Modificar los nombres de los métodos para que sean más generales.

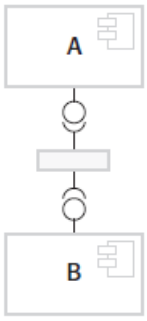
- Incorporar nuevos métodos que mejoren la funcionalidad.
- Hacer uso de excepciones en todos los métodos consistentemente.
- Agregar una interfaz que permita configurar los componentes y se adapten a diferentes situaciones de uso.

2.2.3.6. Composición de componentes

La composición de componentes consiste en integrar dos o más componentes a través de "código pegamento" que permite crear un nuevo componente. Pueden ser de las siguientes formas:

1. **Composición secuencial**: El nuevo componente es creado partiendo de dos componentes ya existentes. Estos componentes no se llaman entre sí, sino a través de código pegamento que llama a los servicios del componente en un orden determinado, asegurando que los resultados brindados por el componente A sean compatibles con las entradas que espera el componente B.

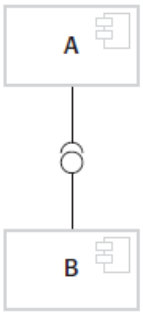
Gráfico 10. Composición secuencial de componentes



Fuente: Ingeniería del Software – Sommerville

2. **Composición jerárquica:** Se da cuando los servicios ofrecidos por un componente son llamados directamente por otro componente. Si las interfaces coinciden, puede que no haya necesidad de un código adicional.

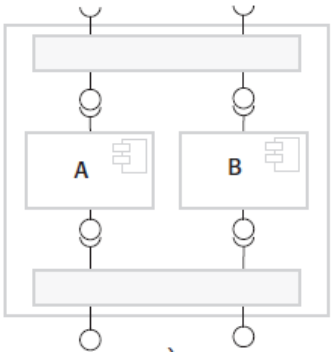
Gráfico 11. Composición jerárquica de componentes



Fuente: Ingeniería del Software – Sommerville

3. **Composición aditiva:** Se da mediante la unión dos o más componentes para crear uno nuevo, lo que combina su funcionalidad. Los componentes se llaman por separado mediante la interfaz externa del componente compuesto, no son dependientes y no se llaman mutuamente. Este tipo de composición puede usarse con componentes que son unidades de programa o con componentes que son servicios.

Gráfico 12. Composición aditiva de componentes



Fuente: Ingeniería del Software – Sommerville

2.2.4. Java

2.2.4.1. Concepto

Java es un lenguaje de programación de propósito general, de alto nivel, orientado a objetos e independiente de la plataforma en la que se programe; es decir, el código que es ejecutado en una plataforma no tiene que ser recompilado para correr en otra. Puede usarse para desarrollar aplicaciones cliente-servidor, web y aplicaciones móviles.

2.2.4.2. Antecedentes

Con el desarrollo de los microprocesadores, que dio origen a las computadoras personales, Sun Microsystems patrocinó en 1991 un proyecto interno de investigación, que resultó en un lenguaje de programación orientado a objetos y basado en C++, al que llamó Java.

El objetivo de Java es que los programas escritos en este lenguaje puedan ejecutarse en una gran variedad de sistemas computacionales.

En 1993, con la popularidad de la web, Sun vio el potencial de Java y poder agregar contenido dinámico a las páginas web.

En el 2009 Oracle adquirió Sun Microsystems, anunciando en el 2010 que el 97% de todas las computadoras de escritorio, tres mil millones de portátiles y 80 millones de dispositivos de televisión ejecutaban Java. Actualmente Java es uno de los lenguajes de desarrollo de software más utilizados en el mundo (16).

2.2.4.3. Máquina Virtual de Java

La máquina virtual Java (JVM por sus siglas en inglés) es una máquina virtual de proceso nativo, es decir, ejecutable en una plataforma específica, capaz de interpretar y ejecutar instrucciones expresadas en un código binario especial (bytecode Java), el cual es generado por el compilador del lenguaje Java. El bytecode Java no es un lenguaje de alto nivel, sino código máquina de bajo nivel.

La JVM actúa como un puente que entiende tanto el bytecode como el sistema sobre el que se pretende ejecutar permitiendo la portabilidad del lenguaje. Existen diferentes máquinas virtuales java para diferentes arquitecturas. Un programa escrito en Windows puede ser interpretado en un entorno Linux usando una JVM para ese entorno.

2.2.4.4. Java Development Kit

Java Development Kit (JDK), es un software que provee herramientas de desarrollo para la creación de programas en Java, pudiendo instalarse en una computadora local o en una unidad de red.

Principales programas que incluye:

- appletviewer.exe: es un visor de applets para generar vistas previas.
- javac.exe: es el compilador de Java.
- java.exe: es el intérprete de Java.
- javadoc.exe: genera la documentación de las clases Java de un programa.

Para el presente trabajo se hará uso de la versión JDK-6u33 de 64 bits, debido a que es la versión que usa el aplicativo de la institución en estudio, y JDK 1.8.0_261 por ser la anterior obsoleta y puede presentar riesgos de seguridad y de rendimiento.

2.2.5. Eclipse IDE

Eclipse es una plataforma de software formado por un conjunto de herramientas de programación de código abierto multiplataforma. Fue desarrollado inicialmente por IBM, ahora es desarrollado por la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto. Eclipse fue liberado con licencia de software libre.

Eclipse dispone de un Editor de texto con un analizador sintáctico, compilación en tiempo real, tiene pruebas unitarias con JUnit, control de versiones con CVS y GIT, asistentes para creación de proyectos, clases, refactorización, etc.

Entre sus principales versiones tenemos:

- Eclipse IDE 2020-12, diciembre 2020
- Eclipse IDE Photon, 27 de junio de 2018
- Mars, 24 de junio de 2015
- Luna, 25 de junio de 2014
- Kepler, 26 de junio de 2013
- Juno, 27 de junio de 2012
- Índigo, 22 de junio de 2011

- Helios, 23 de junio de 2010
- Galileo, 24 de junio de 2009

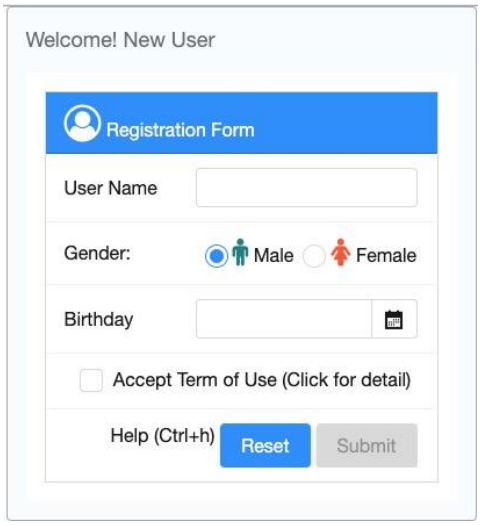
Para el presente trabajo de investigación se usó Eclipse Helios (IDE de la institución en estudio) y Eclipse IDE 2020-12 (última versión disponible), como plataforma de desarrollo JAVA.

2.2.6. Framework ZK

ZK es un framework de interfaz de usuario que permite desarrollar aplicaciones web y aplicaciones móviles sin tener que aprender JavaScript o AJAX.

Diseñar una interfaz de usuario con ZK es fácil. Se puede crear rápidamente la interfaz de usuario usando los diversos componentes XUL/XHTML con que cuenta ZK. El estilo, comportamiento y la función de cada componente pueden configurarse según las necesidades del programador.

Gráfico 13. Interfaz de usuario en ZK



Fuente: zkoss.org

ZK usa archivos ZUL para guardar la programación de interfaz de usuario; estos ZUL son similares a los XML, siendo fáciles de construir y leer. Los componentes de interfaz de usuario de ZK son como bloques de construcción; se pueden combinar, mezclar o heredar para convertirlos en un nuevo componente según los requerimientos del programador; permitiendo la reutilización y la modularidad.

ZK es framework basado en componentes con el modelo de programación orientada a eventos; por lo tanto, los programadores pueden añadir funciones para responder a los eventos de los componentes que se activan por interacción de los usuarios.

ZK soporta un lenguaje de marcación para la definición de una potente interfaz de usuario llamada ZUML.

- ZUML está diseñado para que desarrolladores no expertos diseñen interfaces de usuario de forma eficiente.
- ZUML permite a un desarrollador mezclar diferentes tipos de lenguaje de marcación, tales como el lenguaje XUL de Mozilla y XHTML, todos ellos en la misma página.
- ZUML permite a los desarrolladores embeber scripts en lenguaje Java y usar expresiones JSP para manipular los componentes y acceder a los datos.

2.2.6.1. Ventajas del Framework ZK

- ZUML permite a los no expertos diseñar eficientemente interfaces de usuario.
- Usar scripts en Java ayuda al prototipado rápido y a las personalizaciones.

- No es necesario que el desarrollador tenga conocimientos de Ajax o JavaScript.
- Modelo basado en componentes intuitivo dirigido por eventos.
- Permite centrar toda la lógica de programación en el servidor.
- Permite controlar la interfaz de usuario a través de un controlador en el lado del servidor (17)

2.2.6.2. Ediciones

El Framework ZK se ofrece en 3 ediciones: CE, PE y EE. La edición comunitaria CE es abierta y gratuita bajo LGPL para sus proyectos de código abierto y propietarios. ZK PE y EE vienen con funciones avanzadas, mejor rendimiento y soporte profesional para respaldarlo en la creación de sus aplicaciones profesionales y empresariales.

Tabla 1. Ediciones del Framework ZK

Nº	Características	CE	PE	EE
1	Fusión servidor + cliente	x	x	x
2	Basado en componentes del lado del servidor y controlado por eventos	x	x	x
3	Widget y control del lado del cliente en jQuery orientado a objetos	x	x	x
4	Declaración de UI en ZUML (ZUL)	x	x	x
5	Programación de UI en Java	x	x	x
6	IU basada en modelos y enlace de datos sin programación	x	x	x
7	MVC, MVVM	x	x	x
8	Múltiples presentaciones y estilo 100% basado en CSS	x	x	x
9	Componentes polimórficos y composición recursiva.	x	x	x
10	Ajax transparente y JSON	x	x	x
11	Websocket			x

12	Enlace de datos: formulario de validación de beans			x
13	Componentes sensibles			x
14	Paquete temático ZK			x
15	Tema ZK-Bootstrap			x
16	Subtítulo (Tabbox, Borderlayout)			x
17	Componentes de sombra			x

Fuente: zkoss.org

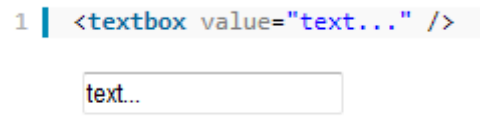
Para el presente trabajo se usó la versión gratuita CE versión 6.5.3.

2.2.6.3. Algunos componentes de ZK

En el presente trabajo de investigación trabajaremos con algunos componentes de ZK:

- **Textbox:** Se utiliza para el ingreso de datos textuales. Se le puede asignar valor, tipo, restricción, filas, columnas, etc. utilizando las propiedades correspondientes.

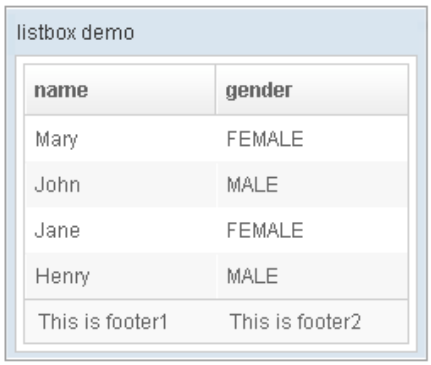
Gráfico 14. Componente Textbox de ZK



Fuente: zkoss.org

- **Listbox:** Se utiliza para mostrar una serie de elementos en una lista. El usuario puede seleccionar un elemento de la lista.

Gráfico 15. Componente Listbox de ZK



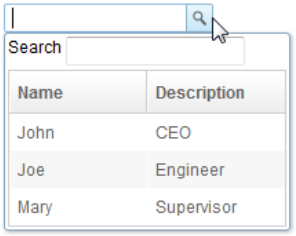

```
<window title="listbox demo" border="normal" width="250px">
    <listbox id="box">
        <listhead sizable="true">
            <listheader label="name" sort="auto" />
            <listheader label="gender" sort="auto" />
        </listhead>
        <listitem>
            <listcell label="Mary" />
            <listcell label="FEMALE" />
        </listitem>
        <listitem>
            <listcell label="John" />
            <listcell label="MALE" />
        </listitem>
        <listitem>
            <listcell label="Jane" />
            <listcell label="FEMALE" />
        </listitem>
        <listitem>
            <listcell label="Henry" />
            <listcell label="MALE" />
        </listitem>
        <listfoot>
            <listfooter>
                <label value="This is footer1" />
            </listfooter>
            <listfooter>
                <label value="This is footer2" />
            </listfooter>
        </listfoot>
    </listbox>
</window>
```

Fuente: zkoss.org

- Bandbox: Es un cuadro de texto especial que incorpora una ventana emergente personalizable (también conocida como una ventana desplegable). Al igual que los cuadros combinados, un bandbox consta de un cuadro de entrada y una ventana emergente. La ventana emergente se abre automáticamente cuando los usuarios presionan Alt + ABAJO o hacen clic en el botón de lupa.

La ventana emergente de un bandbox puede ser cualquier cosa. Está diseñado para brindar a los desarrolladores la máxima flexibilidad. Un uso típico es representar la ventana emergente como un diálogo de búsqueda.

Gráfico 16. Componente Bandbox de ZK




```
<bandbox id="bd">
    <bandpopup>
        <vbox>
            <hbox>
                Search
                <textbox />
            </hbox>
            <listbox width="200px"
                onSelect="bd.value=self.selectedItem.label;bd.close();">
                <listhead>
                    <listheader label="Name" />
                    <listheader label="Description" />
                </listhead>
                <listitem>
                    <listcell label="John" />
                    <listcell label="CEO" />
                </listitem>
                <listitem>
                    <listcell label="Joe" />
                    <listcell label="Engineer" />
                </listitem>
                <listitem>
                    <listcell label="Mary" />
                    <listcell label="Supervisor" />
                </listitem>
            </listbox>
        </vbox>
    </bandpopup>
</bandbox>
```

Fuente: zkoss.org

2.2.7. Aplicaciones Web

2.2.7.1. Concepto

Se entiende por aplicación web a las herramientas que pueden usarse mediante el acceso a un servidor web por medio de internet o de una intranet usando un navegador. Podemos decir también que es un

programa codificado en un lenguaje interpretable por los navegadores web y permite su ejecución.

Las aplicaciones web son independientes del sistema operativo, y facilitan su mantenimiento y actualización pues no requieren distribuir e instalar nada adicional en los equipos de los usuarios que van a usarlo. Algunos ejemplos de aplicaciones web son correos electrónicos, wikis, blogs, tiendas en línea, redes sociales, etc.

2.2.7.2. Internet

Se puede definir a Internet como una red global de computadoras con la finalidad de intercambiar libremente información entre todos sus usuarios.

Internet nace a mediados de la década de los setenta, bajo los auspicios de DARPA, la Agencia de Proyectos Avanzados para la Defensa de Estados Unidos. DARPA inició un programa de investigación de técnicas y tecnologías para unir diversas redes de conmutación de paquetes, permitiendo así a los ordenadores conectados a estas redes comunicarse entre sí de forma fácil y transparente (18).

Posteriormente se crea el protocolo de comunicaciones de datos, IP o Internet Protocol, que permitía a computadoras diversos comunicarse a través de una red, Internet, formada por la interconexión de diversas redes.

A mediados de los ochenta la Fundación Nacional para la Ciencia norteamericana, la NSF, creó una red, la NSFNET, que se convirtió en el backbone (el troncal) de Internet junto con otras redes similares, apareciendo también los primeros proveedores de acceso a Internet.

El boom de Internet se da a mediados de los noventa. El número de proveedores de acceso privado se disparó, permitiendo a millones de personas acceder a Internet y empezó a conocérsela como la web.

La WWW (World Wide Web), junto con el correo electrónico, son dos de los principales servicios que actualmente ofrece Internet; permitiendo, además, acceder a multitud de prestaciones y funciones, así como a infinidad de servicios, programas, tiendas, etc.

2.2.7.3. Fundamentos de la web

La web se fundamenta en dos elementos principales: el protocolo HTTP y el lenguaje HTML.

- **El protocolo HTTP** (hypertext tranfer protocol) es un protocolo simple, orientado a conexión y sin estado. Emplea para su funcionamiento un protocolo de comunicaciones (TCP, transport control protocol) de modo conectado, un protocolo que establece un canal de comunicaciones de extremo a extremo (entre el cliente y el servidor) por el que pasa el flujo de bytes que constituyen los datos que hay que transferir. El protocolo no mantiene estado, es decir, cada transferencia de datos es una conexión independiente de la anterior, sin relación alguna entre ellas. Para transferir una página web tenemos que enviar el código HTML del texto, así como las imágenes que la componen.

- **El lenguaje HTML** (hypertext mark-up language). Es un lenguaje de marcas (se utiliza insertando marcas en el interior del texto) que permite representar el contenido y otros recursos (imágenes, etc.), enlaces a otros documentos, mostrar formularios para posteriormente procesarlos, etc. Es un estándar a cargo del World Wide Web Consortium (W3C) o Consorcio WWW, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, su escritura e interpretación.

2.2.7.4. Servidor web

Es un programa que atiende y responde a peticiones de los navegadores, proporcionándoles los recursos que solicitan mediante el protocolo HTTP o el protocolo HTTPS (la versión segura, cifrada y autenticada de HTTP).

Uno de los aspectos más importantes del servidor web escogido es el nivel de soporte que nos ofrece para servir contenido dinámico. Dado que la mayor parte del contenido web que se sirve no proviene de páginas estáticas, sino que se genera dinámicamente.

2.2.7.5. Apache Tomcat

Es un contenedor web con soporte de servlets y JSPs desarrollado por Apache Software Foundation, e implementa las especificaciones de los servlets y de JavaServer Pages (JSP) de Oracle Corporation.

Tomcat es desarrollado y actualizado por miembros de la Apache Software Foundation y voluntarios independientes. Los usuarios

disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la Apache Software License. Tomcat es un contenedor web y no es un servidor de aplicaciones, como JBoss. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo y en la actualidad es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Para fines del presente trabajo, se usó Tomcat v7.0.107 de 64 bits como servidor web.

2.2.7.6. Ventajas de las aplicaciones web

Las aplicaciones web tienen ciertas ventajas respecto a su contraparte cliente/servidor:

- Su acceso es similar al de una página web, por lo que es necesario contar con acceso a Internet.
- Son independientes de la plataforma (sistema operativo) o dispositivo desde el cual se accede: PC, laptop, Tablet, smartphone, etc.
- No necesitan de espacio en el dispositivo de almacenamiento ya que no requieren instalación.

- Los virus no afectan a los datos, pues estos se guardan en servidores de bases de datos, a través del servidor de aplicaciones.
- Permite compartir información con múltiples usuarios al estar el servicio centralizado en una ubicación única.
- Se puede acceder desde dispositivos móviles, siempre que la aplicación tenga un diseño responsive.
- No requieren contar con computadoras muy potentes ni adquirir licencias de software.
- No consumen recursos de nuestro equipo ya que se realizan en el servidor web o de aplicaciones.
- Se tiene disponible siempre la última versión de la aplicación, pues esta se actualiza en el servidor web.

2.2.7.7. Terminología de las aplicaciones web

Dentro de las aplicaciones web, es común el uso de los siguientes términos:

- **Servidor Web:** Es el software que provee páginas Web cuando se hace una petición por medio de un navegadores Web; ya sea al hacer clic en algún vínculo de una página Web o a través del acceso de una URL en el navegador. Los servidores Web más conocidos son Microsoft Internet Information Server (IIS), NGINX, Apache HTTP Server, entre otros.
- **Servidor de aplicaciones:** Es el software que se encarga de procesar la lógica de negocio y el acceso a datos de las aplicaciones web. Sirve de apoyo al servidor web procesando las páginas solicitadas para luego ser enviadas al navegador.

- **Base de datos:** Son conjunto de datos que se almacenan en tablas donde cada fila constituye un registro de datos, y las columnas son los campos del registro (base de datos relacionales).
- **Base de datos relacional:** Base de datos formada por tablas que comparten datos por medio de relaciones.
- **Registros:** Son datos que se extraen de una o más tablas de una base de datos.
- **Controlador de base de datos:** Es el software cuya función es ser intérprete para que una aplicación Web pueda comunicarse con una base de datos, permitiéndole la lectura y manipulación datos que sin él sería indescifrable.
- **Sistema de administración de base de datos (DBMS):** Es el software que se usa para manipular y crear bases de datos. Entre las más conocidas tenemos a Oracle, MySQL, PostgreSQL y MS SQL Server (19).
- **Consulta de base de datos:** Es la acción de extraer registros de una base de datos. Las consultas constan de criterios de búsqueda codificados en un lenguaje de base de datos llamado SQL.
- **Página dinámica:** Es la página Web devuelta por el servidor de aplicaciones antes de ser enviada a un navegador.

2.2.7.8. Frameworks para el desarrollo Web

Los frameworks son herramientas de desarrollo web que permiten el desarrollo ágil de aplicaciones mediante la aportación de librerías y/o funcionalidades ya creadas. Los framework evitan reinventar la rueda permitiendo al programador centrarse en el problema a resolver y no en

la implementación de funcionalidades que son de uso común y que ya están resueltas por otros (20).

Los framework proporcionan soluciones a la mayoría de las problemáticas comunes del desarrollo de aplicaciones web, como por ejemplo:

- Arquitectura de Desarrollo MVC (Modelo, Vista, Controlador).
- Autenticación de usuarios, niveles control de acceso, sesiones, cookies…
- Estructura de Directorios y Archivos modulares.
- Manejo de Peticiones y Respuestas, (POST, GET, WebServices).
- Manejo de formularios y validación de datos.
- Manejo de localidades y multi-idioma.

Para el desarrollo del componente de software del presente trabajo se usó el Framework ZK, para la capa de presentación. La institución en estudio también usa los frameworks Spring para la inyección de dependencias y Hibernate para la persistencia de datos.

2.2.7.9. Modelo – Vista – controlador

MVC (Modelo – Vista - Controlador) es un patrón de arquitectura de software que separa el código en sus distintas responsabilidades a través de capas que se encargan de hacer una tarea muy concreta.

Los patrones arquitectónicos se abocan a un problema de aplicación específica dentro de un contexto dado y sujeto a limitaciones y restricciones. El patrón propone una solución arquitectónica que sirve como base para el diseño de la arquitectura (21).

- **Modelo:** Es la capa donde se trabaja con los datos. Contiene métodos para acceder a la información y también para actualizar su estado. Los datos se obtienen generalmente de una base de datos.
- **Vista:** Es la capa que contienen el código para la visualización de las interfaces de usuario. Permite la renderización de los estados de nuestra aplicación en HTML.
- **Controlador:** Es la capa Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc. Es la capa intermedia entre la vista y el modelo sirviendo de enlace entre ellas (22).

2.2.7.10. Diseño de interfaz para aplicación web

La interfaz de usuario es el medio por el cual el usuario interactúa con la aplicación web. Debe diseñarse de tal manera que responda a las siguientes preguntas:

- **¿Dónde estoy?:** Debe dar una indicación de la aplicación web a la que se ha accedido y su localización dentro de la misma.
- **¿Qué puedo hacer ahora?:** Debe ayudar al usuario a entender sus opciones actuales; cuáles funciones están disponibles, qué vínculos están activos, qué contenido es relevante, etc.

- **¿Dónde he estado, hacia dónde voy?:** Debe disponer un "mapa" de fácil entendimiento que indique dónde ha estado el usuario y las trayectorias que puede tomar para moverse a cualquier punto de la aplicación.

Algunos lineamientos prácticos para el diseño de las interfaces de aplicaciones web son las siguientes:

- La lectura en un monitor de computadora es aproximadamente 25 por ciento más lenta que la que se hace en un papel; por tanto, no se debe obligar al usuario a leer grandes cantidades de texto.
- Evitar los avisos "en construcción" cuando las páginas u opciones no están disponibles.
- La información importante debe situarse dentro de las dimensiones de una ventana normal de navegación.
- Los menús de navegación y los encabezados deben diseñarse de manera consistente y deben estar disponibles en todas las páginas a las que tenga acceso el usuario.

La estética nunca debe obstaculizar la funcionalidad. Las opciones de navegación deben de tener un objetivo claro y ser obvias para el usuario (21).

2.3. Hipótesis

2.3.1. Hipótesis general

La implementación de componentes de software mejorará el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020.

2.4. Variables

2.4.1. Variable independiente

Implementación de Componentes de Software para Mejorar el Desarrollo de los Aplicativos Web

III. Metodología

3.1. El tipo y el nivel de la investigación

3.1.1. Tipo

La presente investigación es de tipo cuantitativa.

Según Cabezas, Andrade y Torres (23), el método cuantitativo; utiliza la recolección de datos para probar la hipótesis, con base en la medición numérica y análisis estadístico, para establecer patrones de comportamiento y probar teorías.

3.1.2. Nivel

El nivel de la investigación realizado es aplicativo.

Según Murillo (24), este tipo de investigación también recibe el nombre de práctica o empírica. Se caracteriza porque busca la aplicación o utilización de los conocimientos que se adquieren.

3.2. Diseño de la investigación

El diseño que empleado es pre-experimental.

De acuerdo con Ramírez (25), en un diseño pre-experimental no hay una selección aleatoria de los elementos ni se incluye un grupo de control (diseño experimental); es cuando se toma un grupo y se evalúa (pre-test), en seguida se somete a un tratamiento (experimento), para finalmente repetir la

evaluación (post-test). Lo que se busca con este diseño es medir el cambio experimentado por el grupo de prueba a causa del tratamiento.

Tabla 2. Diseño de la investigación

Grupo	Asignación	Pre-test	Tratamiento	Post-test
A	no R	O	X	O

Donde:

R: Aleatorización (R del inglés random, “azar”)

O: Observación, medida registrada en el pre-test o en el post-test

X: Tratamiento (los subíndices 1 a n indican diferentes tratamientos) (26)

3.3. Población y muestra

La población está conformada por 6 desarrolladores del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima.

La muestra ocupa a toda la población, conformada por 6 desarrolladores, es por ello por lo que la denominaremos una población muestral, para obtener mayor precisión en los resultados a obtener.

3.4. Definición y operacionalización de las variables y los indicadores

Tabla 3. Definición y operacionalización de las variables

VARIABLE	DEFINICIÓN	DIMENSIONES	INDICADORES
Implementación de Componentes de Software para Mejorar el Desarrollo de los Aplicativos Web	**Definición conceptual:** • Componente de software: Bloque de construcción de software, parte modular, desplegable y sustituible de un sistema, que incluye implantación y un conjunto de interfaces. • Aplicativo web: Herramientas informática que se utiliza accediendo a un servidor web usando internet o una intranet por medio de un navegador. **Definición operacional:** esta variable se va a medir mediante un experimento	Se medirá el tiempo de desarrollo y complejidad antes del experimento. Se realizará el experimento (Implementación de componente de software). Se medirá el tiempo de desarrollo y complejidad después del experimento.	• Tiempo de desarrollo de un proceso en minutos antes del experimento. • Líneas de código del proceso antes del experimento. • Componente de software elaborado. • Tiempo de desarrollo de un proceso en minutos después del experimento. • Líneas de código del proceso después del experimento.

Fuente: Elaboración propia

3.5. Técnicas e instrumentos

Para la recolección de datos se utilizará como técnica la observación y como instrumento las notas de campo.

Para la implementación del componente de software se utilizó como instrumentos:

- Computadora personal
- Eclipse IDE como herramienta de desarrollo
- Framework ZK para la capa de presentación
- JAVA como lenguaje de programación
- StartUML 4.0.1 para los diagramas UML

3.6. Plan de análisis

Los datos obtenidos fueron procesados, codificados e ingresados en el Software de Ofimática Excel de Microsoft 365, el cual también se usó para el análisis de los datos y la obtención de tablas y gráficos de las variables en estudio.

3.7. Matriz de consistencia

Tabla 4. Matriz de consistencia

IMPLEMENTACIÓN DE COMPONENTES DE SOFTWARE PARA MEJORAR EL DESARROLLO DE LOS APLICATIVOS WEB EN EL MINISTERIO DE ECONOMÍA Y FINANZAS – LIMA; 2020.

PROBLEMA	OBJETIVOS	HIPOTESIS	VARIABLE	METODOLOGIA
¿De qué manera la implementación de componentes de software permite mejorar el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020?	**Objetivo General:** Implementar Componente de Software para Mejorar el Desarrollo de los Aplicativos Web en el Ministerio de Economía y Finanzas – Lima; 2020. **Objetivos Específicos:** 1. Identificar procesos de desarrollo de software a cambiar a modelo basado en componentes. 2. Medir el tiempo de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas –	La implementación de componentes de software mejorará el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020	**Variable independiente:** Implementación de Componentes de Software para Mejorar el Desarrollo de los Aplicativos Web.	Tipo: Cuantitativa Nivel: Aplicada Diseño: Pre-Experimental

	Lima; 2020 antes de implementar el componente de software. 3. Determinar la complejidad de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas – Lima; 2020 antes de implementar el componente de software. 4. Implementar componente de software para reducir el tiempo y complejidad del Desarrollo de los Aplicativos Web del Ministerio de Economía y Finanzas – Lima; 2020. 5. Comprobar y contrastar el tiempo y complejidad de desarrollo de un proceso, de un aplicativo web del Ministerio de Economía y Finanzas -			

	Lima; 2020 después de implementar el componente de software.			

Fuente: Elaboración propia

3.8. Consideraciones éticas y de rigor científico

Para el desarrollo del presente trabajo de investigación denominado "Implementación de Componentes de Software para Mejorar el Desarrollo de los Aplicativos Web en el Ministerio de Economía y Finanzas – Lima; 2020.", se ha considerado el cumplimiento estricto de los principios de la Deontología Informática con respecto a la protección del bien más preciado de las empresas e instituciones, que es la información.

Como Ingenieros de Sistemas estamos comprometidos a cumplir los siguientes ocho principios éticos (27):

1. **Responsabilidad:** actuando congruentemente con el interés social.
2. **Confidencialidad:** manejando en forma responsable la información que se dispone de las personas, empresa o institución producto de su trabajo.
3. **Calidad del producto:** el trabajo realizado debe estar acorde con los estándares profesionales adecuados, y que satisfagan las necesidades del usuario o cliente.
4. **Juicio:** mantener la objetividad profesional sobre el trabajo encomendado, moderando juicios técnicos y evitando prácticas que perjudiquen a la organización.
5. **Promover un enfoque ético en la gestión:** garantizando la buena gestión del proyecto encomendado, incluyendo procedimientos efectivos para promover calidad y reducción del riesgo.
6. **Promover el conocimiento:** incrementando la reputación e integridad de la profesión, acorde con el beneficio social.
7. **Apoyo Laboral:** apoyar y ser justos con sus colegas.
8. **Actualización Permanente:** participar permanentemente en el aprendizaje relacionado con la práctica de su profesión y con enfoque ético.

IV. Resultados

4.1. Resultados

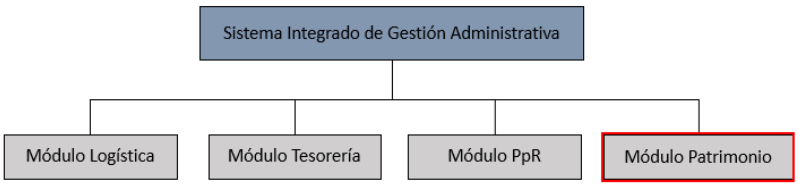
4.1.1. Recolección de datos

4.1.1.1. Identificación del proceso a componetizar

La aplicación web objeto de estudio es el Sistema Informático de Gestión Administrativa – SIGA en su plataforma Web, denominado SIGAWEB en su versión 20.05.00.

El SIGAWEB presenta varios módulos, de los cuales analizamos el Módulo de Patrimonio.

Gráfico 17. Módulos del SIGAWEB



Fuente: Ministerio de Economía y Finanzas

Se identificó un proceso que se repite en las diversas opciones del módulo estudiado, el cual consiste en la búsqueda de información a través de un listado. En la interfaz gráfica el nombre del componente que realiza dicha labor es el Bandbox el cual, como ya se indicó en el marco teórico, es personalizable y puede contener a otros componentes.

Gráfico 18. Búsqueda de información a través de un bandbox

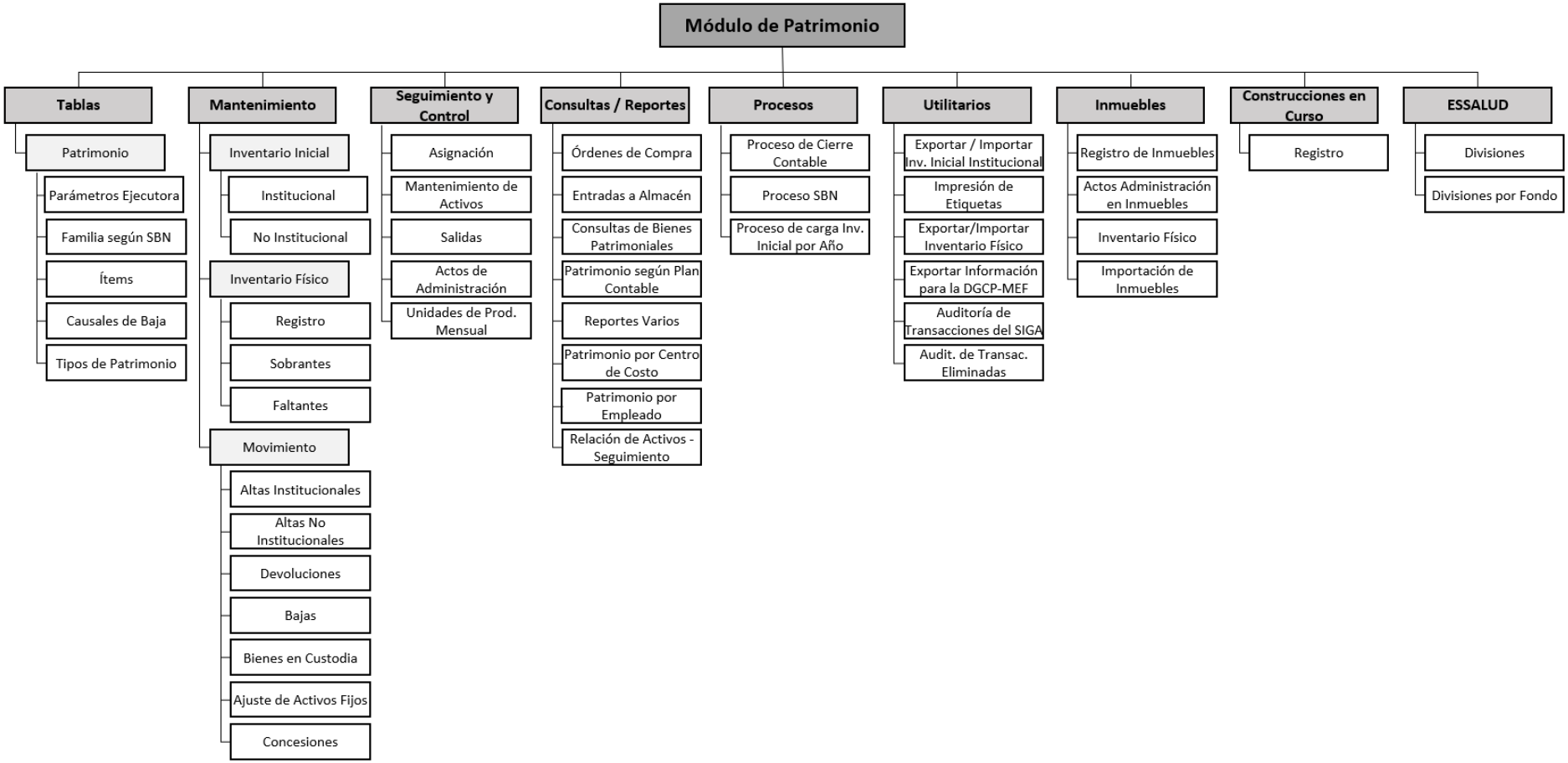
Fuente: SIGAWEB – Ministerio de Economía y Finanzas

La interfaz de búsqueda de información, que usa un bandbox, está formado internamente por dos componentes: un Listbox para presentar la información y Textbox para la búsqueda de información dentro de la lista. A este proceso identificado hemos llamado **Proceso buscar información**.

4.1.1.2. Identificación de opciones con procesos a componetizar

Se ha revisado las diferentes opciones de la que consta el módulo Patrimonio del SIGAWEB.

Gráfico 19. Opciones del Módulo de Patrimonio del SIGAWEB



Fuente: SIGAWEB – Ministerio de Economía y Finanzas

Se ha seleccionado las siguientes opciones:

Tabla 5. Opciones con procesos a componetizar

Cód.	Opción	Ventana	Campo
001	Mantenimiento / Inventario Inicial Institucional	Registro de Ingreso	Centro de Costo
002	Mantenimiento / Inventario Inicial Institucional	Registro de Ingreso	Empleado Responsable
003	Mantenimiento / Inventario Inicial Institucional	Datos del Activo Fijo	Ítem
004	Mantenimiento / Inventario Inicial Institucional	Datos del Activo Fijo	Ubic. Física
005	Mantenimiento / Inventario Inicial Institucional	Datos del Activo Fijo	Usuario Final
006	Mantenimiento / Inventario Inicial Institucional	Datos del Activo Fijo	Proveedor
007	Mantenimiento / Inventario Inicial Institucional	Datos del Activo Fijo	Almacén
008	Mantenimiento / Movimiento / Bajas	Baja de Activos	Familia
009	Seguimiento y Control / Salidas	Mantenimiento de Salida de Bienes	N° O/C
010	Seguimiento y Control / Salidas	Mantenimiento de Salida de Bienes	Sede
011	Seguimiento y Control / Salidas	Mantenimiento de Salida de Bienes	Centro Costo Solicitante
012	Seguimiento y Control / Salidas	Mantenimiento de Salida de Bienes	Responsable

Fuente: Elaboración propia

4.1.1.3. Recolección de datos previo al experimento

De acuerdo con los datos recogidos en las notas de campos del Anexo 4, se ha elaborado el siguiente cuadro resumen:

Tabla 6. Datos previos al experimento

Cód. Componente	Nº Col.	Filtros	Tiempo de Desarrollo (min.)	Nº Líneas Código Frontend	Nº Líneas Código Backend
001	2	No	25	25	18
002	3	No	27	31	20
003	5	Si	55	55	67
004	3	Si	75	34	106
005	5	No	25	43	26
006	6	No	32	49	12
007	3	No	27	31	22
008	2	Si	45	26	51
009	5	Si	65	58	97
010	2	Si	42	28	57
011	2	Si	30	28	56
012	4	Si	35	53	90

Fuente: Elaboración propia

Para todos los componentes evaluados, solo se ha tomado en cuenta el tiempo y las líneas de código para renderizar el componente en pantalla. Esto incluye cada uno de los componentes contenidos dentro del bandbox (listbox, textbox), el evento de selección de un ítem y el código de búsqueda y filtrado asociado a los eventos del textbox y listbox.

4.1.1.4. Recolección de datos posterior al experimento

De acuerdo con los datos recogidos en las notas de campos del Anexo 4, se ha elaborado el siguiente cuadro resumen:

Tabla 7. Datos posteriores al experimento

Cód. Componente	Nº Col.	Filtros	Tiempo de Desarrollo (min.)	Nº Líneas Código Frontend	Nº Líneas Código Backend
001	2	No	12	1	18
002	3	No	18	1	18
003	5	Si	20	1	19
004	3	Si	15	1	19
005	5	No	17	1	18
006	6	No	19	1	18
007	3	No	13	1	18
008	2	Si	11	1	19
009	5	Si	15	1	19
010	2	Si	13	1	19
011	2	Si	10	1	19
012	4	Si	12	1	19

Fuente: Elaboración propia

Para todos los componentes evaluados, se ha tomado en cuenta el tiempo y las líneas de código para renderizar el componente en pantalla y el evento de selección de un ítem, usando el componente desarrollado en la presente investigación.

4.1.2. Elección de las herramientas de desarrollo del componente web

4.1.2.1. Servidor de aplicaciones

Se escogió Apache Tomcat v7.0.107 de 64 bits como servidor web. La elección de dicho servidor se debió a los siguientes criterios:

- Soporta un alto nivel de tráfico.

- Al ser software libre, evita gastos adicionales a las entidades que hagan uso del sistema.
- Soporta aplicaciones web desarrolladas en JAVA
- Funciona en cualquier sistema operativo que disponga de la máquina virtual de JAVA, como Windows y Linux.
- Está acorde con las especificaciones técnicas solicitadas por la Oficina General de Tecnologías de la Información – OGTI del Ministerio de Economía y Finanzas para sus aplicaciones web.

4.1.2.2. Gestor de Base de Datos

Para el desarrollo del componente se usó como gestor de base datos SQL Server 2017, a fin de contar con datos de prueba mediante consulta y llenado de listas que sirve de input al componente.

Este gestor de base de datos está acorde con las especificaciones técnicas solicitadas por la Oficina General de Tecnologías de la Información – OGTI del Ministerio de Economía y Finanzas para sus aplicaciones web.

4.1.2.3. Lenguaje de Programación

El lenguaje de programación utilizado es JAVA, pues es robusto, orientado a objetos, multiplataforma y es el que usa el aplicativo SIGAWEB del Ministerio de Economía y Finanzas.

4.1.2.4. Entorno de desarrollo (IDE)

Se usó como entorno de desarrollo Eclipse Helios al ser la herramienta básica de desarrollo usada por el equipo de desarrollo SIGA de la

Oficina General de Tecnologías de la Información – OGTI del Ministerio de Economía y Finanzas.

4.1.2.5. Framework utilizado

Si bien el aplicativo SIGAWEB usa diversos frameworks en su código fuente, el componente desarrollado solo requiere el framework ZK.

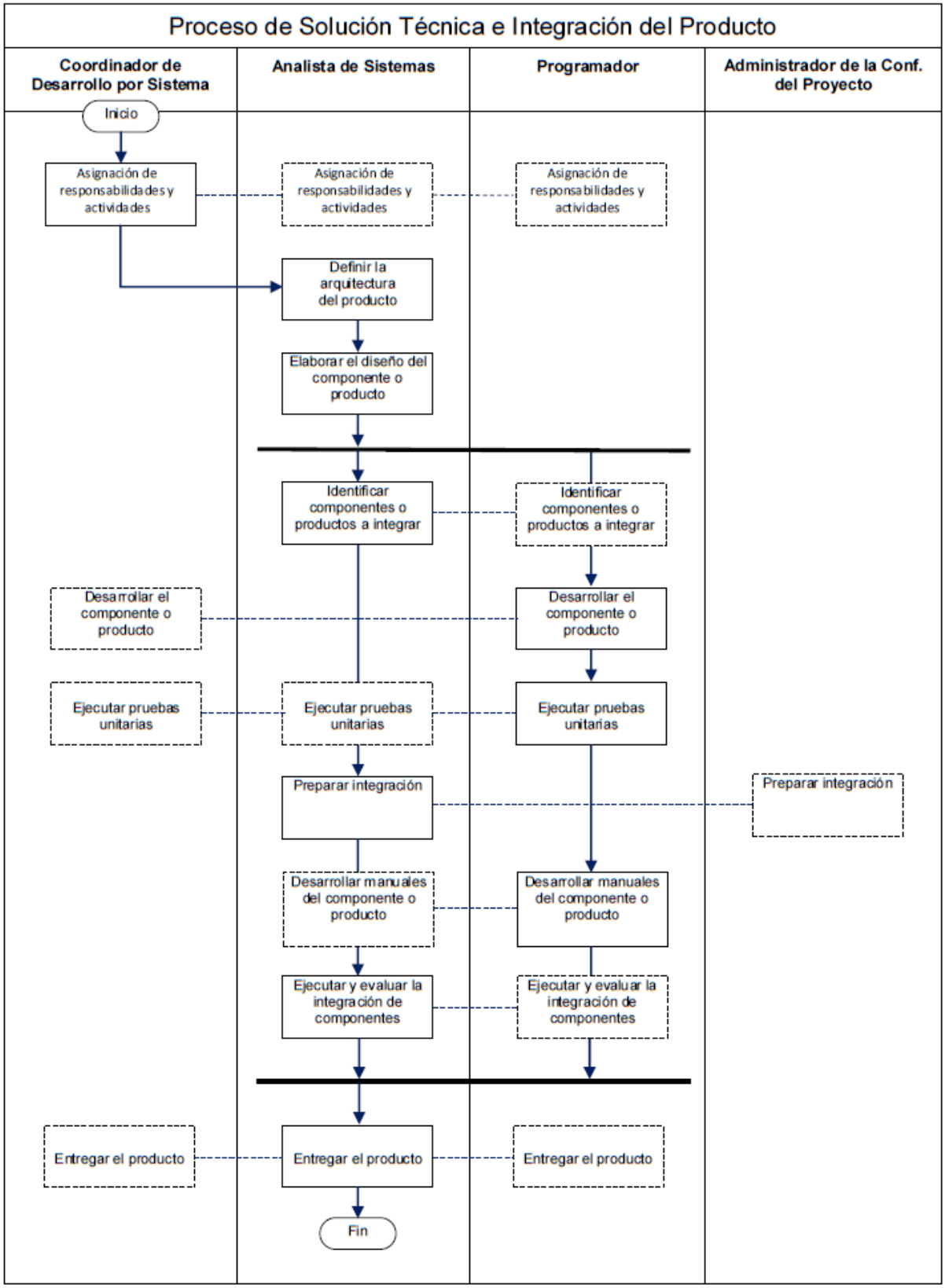
La versión de ZK utilizada es la 6.5.3, pues está acorde con las herramientas estándar autorizadas por la Oficina General de Tecnologías de la Información – OGTI del Ministerio de Economía y Finanzas para el desarrollo de sus aplicaciones web.

4.1.3. Metodología y desarrollo del componente

4.1.3.1. Metodología

Para la implementación del Componente de Búsqueda de Información se ha seguido el Proceso de Solución Técnica e Integración del Producto de la Oficina General de Tecnologías de la Información del Ministerio de Economía y Finanzas, el cual está enmarcado con el modelo Capability Maturity Model Integration (CMMI) v1.3.

Gráfico 20. Metodología de Desarrollo OGTI



Fuente: OGTI - Ministerio de Economía y Finanzas

Se ha asumido los roles Analista de Sistemas y Programador y realizado las siguientes actividades:

Tabla 8. Actividades para el desarrollo del componente

Analista de Sistemas	Programador
Elaborar el diseño del componente o producto	Desarrollar el componente o producto
	Ejecutar pruebas unitarias
	Desarrollar manuales del componente o producto

Fuente: Elaboración propia

4.1.3.2. Elaboración del componente

1. Diseño del componente

Requerimientos Funcionales

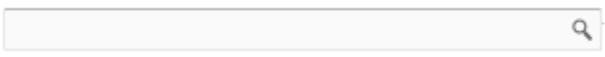
RF-01: Desarrollar un componente de software para la presentación y búsqueda de información.

Módulo: Patrimonio

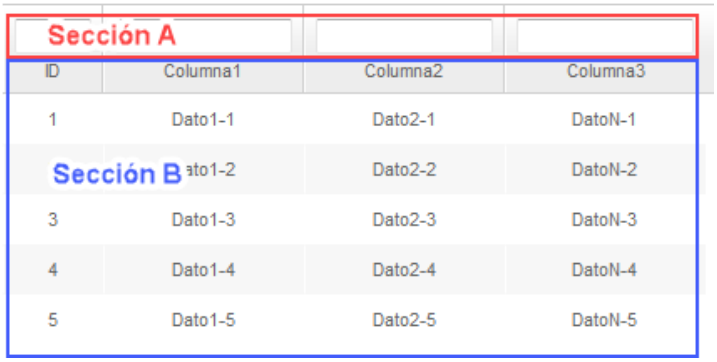
Especificaciones:

El componente a desarrollar debe cumplir las siguientes especificaciones:

1. Debe estar basado en el componente Bandbox del framework ZK.



2. Al hacer clic en el ícono desplegará una grilla con el siguiente diseño:



3. La Sección A (Filtros) tendrá una caja de texto por cada columna de la grilla. Cada caja de texto permitirá escribir un texto y al pulsar Enter filtrará los datos de la sección B que contengan la palabra escrita en dicha caja de texto para la columna respectiva. Se podrá filtrar datos por más de una columna.

4. La Sección B (Datos) mostrará información en forma de grilla de una lista de datos pasados al componente como dato de entrada.

5. El componente debe permitir ingresar el número de columnas a pintar.

6. El componente debe permitir ingresar el rótulo de la cabecera de cada columna.

7. El componente debe permitir indicar si se muestra o no la Sección A de filtros.

8. El componente debe permitir indicar la alineación que pueda tener los datos de cada columna (izquierda, centro, derecha).

9. El componente debe permitir indicar el ancho de cada columna.

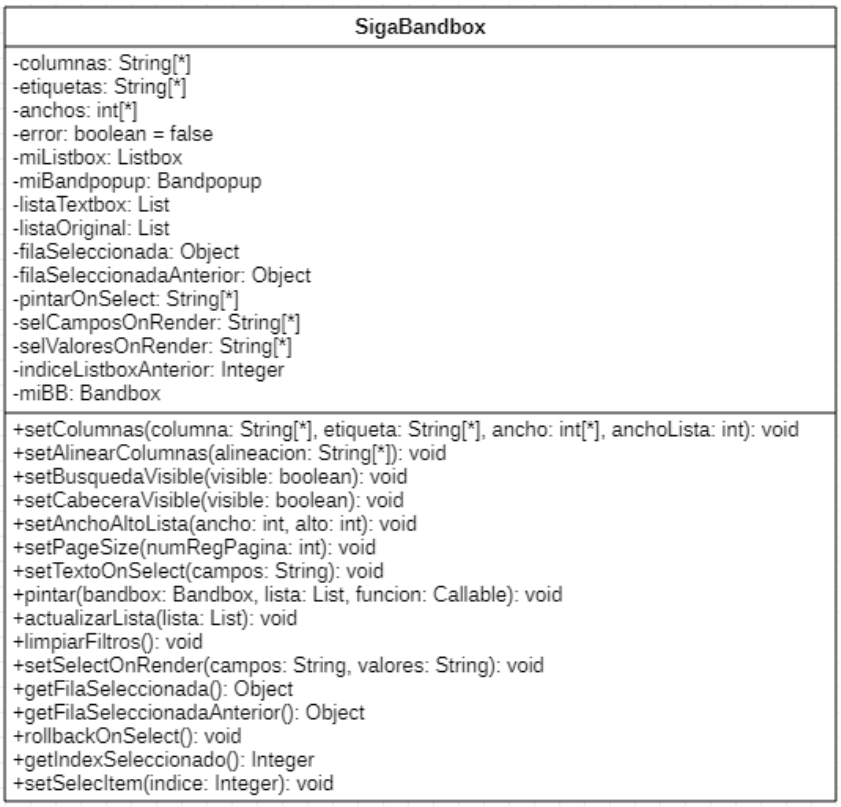
10. El componente debe permitir seleccionar un registro de la grilla y mostrarlo en el Bandbox. Debe permitir indicar que dato se muestra.

11. El componente debe crearse como objeto a partir de una clase y tener métodos para el manejo de la funcionalidad requerida.

2. Desarrollo del componente

Nombre del Componente: SigaBandox

Gráfico 21. Diagrama de Clases SigaBandbox



Fuente: Elaboración propia

Especificaciones de las Propiedades del SigaBandbox		
Propiedad	**Tipo**	**Descripción**
columnas	String[]	Arreglo de tipo cadena para almacenar los datos de las columnas de la grilla.
etiquetas	String[]	Arreglo de tipo cadena para almacenar las etiquetas de la cabecera de las columnas de la grilla.
anchos	int[]	Arreglo de tipo numérico para contener el ancho de cada columna.
error	boolean	Boleano para identificar si se ha producido un error en la funcionalidad del componente.
miListbox	Listbox	Componente Listbox para el renderizado de la lista de ítems.

miBandpopup	Bandpopup	Componente Bandpopup para poder desplegar y ocultar la grilla de datos.
listaTextbox	List<Textbox>	Lista de Textbox para los filtros de la grilla, por cada columna.
listaOriginal	List<Object>	Lista de objetos para almacenar la lista original de datos.
filaSeleccionada	Object	Objeto para almacenar la fila seleccionada
filaSeleccionadaAnterior;	Object	Objeto para almacenar la fila seleccionada anterior
pintarOnSelect	String[]	Almacena los datos a pintar al renderizar el componente
SelCamposOnRender	String[]	Almacena los nombres de los campos seleccionados al renderizar

selValoresOnRender	String[]	Almacena los valores de los campos de la fila seleccionada al renderizar
indiceListboxAnterior	Integer	Almacena el índice del registro anteriormente seleccionado
miBB	Bandbox	Almacenan el Componente Bandbox asociado definido en la página web.

Especificaciones de los Métodos del SigaBandbox
Método: **setColumnas**(String[] columna, String[] etiqueta, int[]ancho, int anchoLista) **Descripción:** Setea las columnas que tendrá el componente **Parámetros:** **columna**: Arreglo de tipo String **String[]** para indicar las columnas a visualizar. En caso se requiera que una columna este compuesta por más de un campo se deberá colocar dentro del arreglo dejando un espacio en blanco. **Ejemplo: {"campo1 campo2 campo3", "columna2"}** **etiqueta**: Arreglo de tipo String **String[]** para indicar las etiquetas (rótulos) de las columnas.

ancho: Arreglo de tipo int **int[]** para indicar en ancho de las columnas.

anchoLista: Indica el ancho de la lista. Para mostrar todas las columnas, se recomienda que este valor sea la suma del ancho de todas las columnas mostradas, más 80 píxeles.

Método:
setAlinearColumnas(String[] alineacion)
Descripción:
Setea la alineación de los datos en las columnas. El método debe ser llamado después de **setColumnas()**
Parámetros:
alineacion: Arreglo de cadenas String[]. Para indicar la alineación usar:
"left" = Alineado a la izquierda
"center" = centrado
"right" = Alineado a la derecha

Método:
setBusquedaVisible(boolean visible)
Descripción:
Establece si las cajas de búsquedas son visibles o no.
Parámetros:
visible : true=Visible false=Invisible.

Método:
setCabeceraVisible(boolean visible)
Descripción:
Establece si la cabecera de lista es visible o no.
Parámetros:
visible: true=Visible false=Invisible.

Método:
setAnchoAltoLista(int ancho, int alto)
Descripción:
Establece el Ancho y Alto de la lista.

Parámetros:

ancho: Ancho de la lista.
alto: Alto de la lista.

Método:

setPageSize(int numRegPagina)

Descripción:

Establece el número de registros por página.

Parámetros:

numRegPagina: Número de registros por página.

Método:

setTextoOnSelect(String campos)

Descripción:

Establece el dato a mostrar en el Bandbox al seleccionar un registro de la lista.

Parámetros:

campos: Cadena indicando los campos a mostrar separados por espacios en blanco. En el caso de trabajar con Map, se deberá colocar el identificador del elemento del mapa, en caso de ser un Domain, se deberá colocar el nombre del atributo de la clase con la primera letra en Mayúscula El nombre de los campos debe estar dentro del arreglo de campos especificados con el método setColumnas(). En caso de indicar un texto que no es columna, se insertará como parte del texto a mostrar.

Si se desea mostrar más de un campo concatenado se deberá colocar el símbolo +.

Ej.

```
String[] columnas = {"id", "nombre", "edad", "direccion", "sueldo"};
miBandbox.setTextoOnSelect("id - nombre");
miBandbox.setTextoOnSelect("id + nombre - sueldo");
```

Método:

pintar(Bandbox bandbox, List lista, Callable funcion)

Descripción:

Renderiza el Bandbox.

Parámetros:

bandbox: Bandbox a renderizar.

lista: Lista de datos con los que se alimenta el Bandbox. La lista es de la forma List<Map<String,Object>>

funcion: función que se ejecutará al seleccionar un registro. Usar la siguiente plantilla:

new Callable(){public Object call() {return onClickRow();}}

Ejem.

this.miBB.pintar(new Callable(){public Object call() {return onSelectBB();}});

onSelectBB() es el nombre de la función que se ejecutará y debe retornar un boolean.

Método:

actualizarLista(List lista)

Descripción:

Vuelve a renderizar el Bandbox con la nueva lista pasada como parámetro. En caso se dese limpiar el bandbox se deberá pasar null como parámetro.

Parámetros:

lista: Nueva lista para renderizar. Pasar null si se desea limpiar el bandbox

Método:

limpiarFiltros()

Descripción:

Limpia los filtros del Bandbox al refrescar la lista.

Parámetros:

Ninguno

Método:

setSelectOnRender(String campos, String valores)

Descripción:

Establece los campos y valores criterio a evaluar para seleccionar un registro al renderizar.

Parámetros:

campos: Cadena indicando los campos a evaluar separados por espacios en blanco. El nombre de los campos debe estar dentro del arreglo de campos especificados con el método setColumnas().

valores: Cadena indicando los valores de los campos a evaluar separados por espacios en blanco. El nombre de los campos debe estar dentro del arreglo de campos especificados con el método setColumnas().

Ej.

```
String[] columnas = {"id", "nombre", "edad", "direccion",
"sueldo"};
miBandbox.setSelectOnRender("id", "4");
```

Método:

getFilaSeleccionada()

Descripción:

Devuelve un mapa Map<String,Object> con el registro seleccionado.

Parámetros:

Ninguno

Método:
getFilaSeleccionadaAnterior()
Descripción:
Devuelve el registro seleccionado anterior.

Parámetros:
Ninguno

Método:
rollbackOnSelect()
Descripción:
Deshace la selección (Evento onSelect).

Parámetros:
Ninguno

Método:
getIndexSeleccionado()
Descripción:
Devuelve el índice del Registro actualmente seleccionado. Este dato puede usarse para regresar a esa posición, luego de haber seleccionado otro registro mediante el método setSelecItem(Integer indice).

Parámetros:
Ninguno

Retorna:
Integer

Método:
setSelecItem(Integer indice)
Descripción:
Selecciona el ítem correspondiente al índice pasado como parámetro.

Parámetros:
indice: Posición del ítem que se desea seleccionar. Este dato puede capturarse con el método getIndexSeleccionado().

3. **Pruebas unitarias**

   Ver el Anexo 5

4. **Manual de uso del componente**

   Ver el Anexo 6

4.1.4. Elaboración de tablas y cuadros

Tabla 9. Promedio de tiempo y líneas de código de desarrollo componente de búsqueda de información

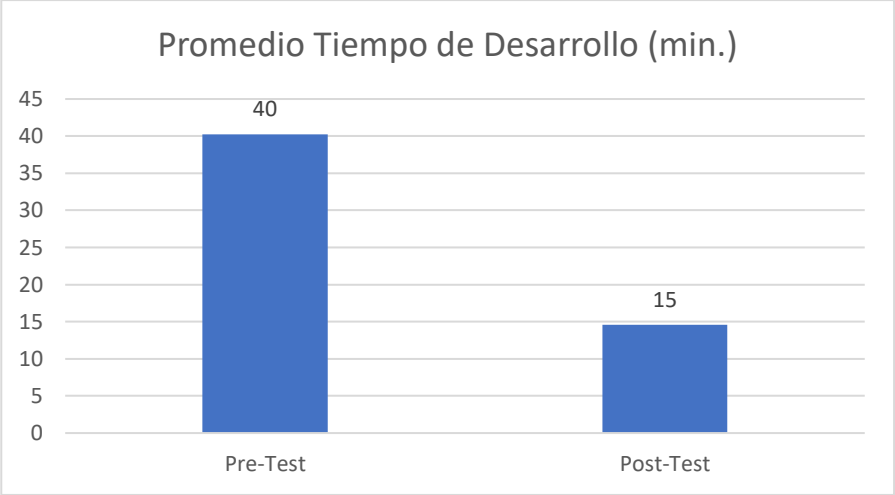
Experimento	Promedio Tiempo de Desarrollo (min.)	Promedio Nº Líneas Código Frontend	Promedio Nº Líneas Código Backend	Total Líneas Código
Pre-Test	40	38	52	90
Post-Test	15	1	19	20

Fuente: Elaboración propia

La Tabla 9 es un consolidado de las Tablas 6 y 7, mostrando los promedios de los tiempos y líneas de código de desarrollo del componente de búsqueda de información, antes y después de aplicar el experimento.

Se puede apreciar una disminución significativa en los tiempos de desarrollo, así como en las líneas de código.

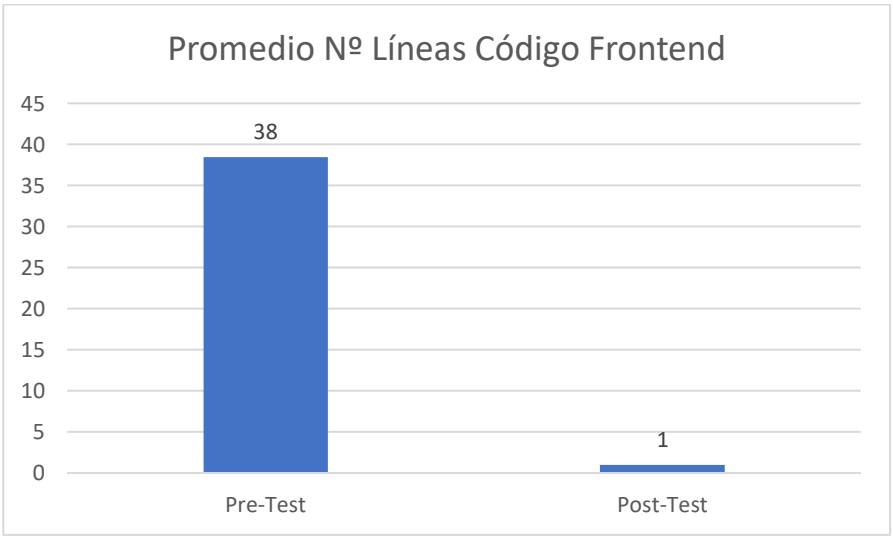
Gráfico 22. Promedio de tiempo de desarrollo del componente de búsqueda de información



Fuente: Elaboración propia

En el Gráfico 21 podemos apreciar que el tiempo de desarrollo del componente de búsqueda de información disminuye a menos de la mitad luego de aplicar el experimento. El componente de búsqueda de información SigaBandbox permite reducir al 36.2% del tiempo promedio que se demoraría elaborar el mismo componente de la forma original.

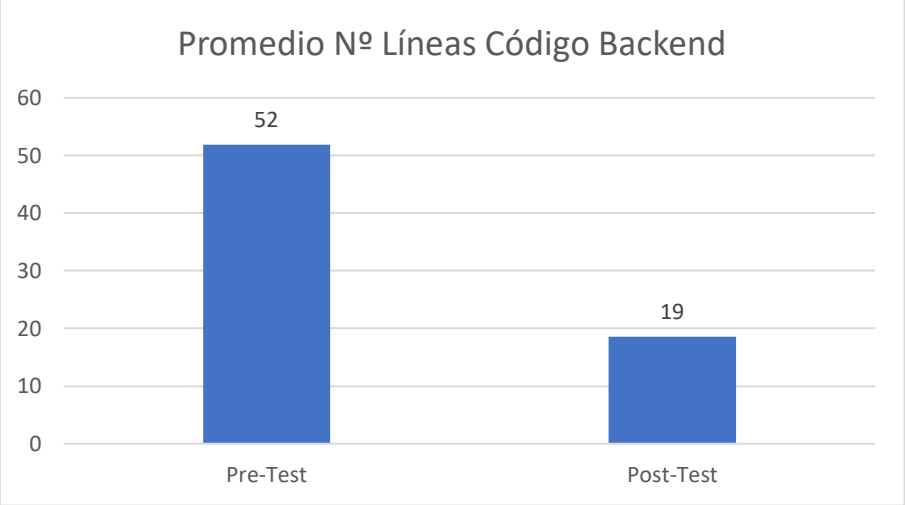
Gráfico 23. Promedio de líneas de código en el Frontend para el desarrollo del componente de búsqueda de información



Fuente: Elaboración propia

El Gráfico 22 contrasta las líneas de código promedio necesarias para desarrollar un componente de búsqueda de información con y sin el componente SigaBandbox. Es interesante resaltar que sin importar que tan complejo sea el requerimiento del componente de búsqueda de información, el uso del SigaBandbox siempre requiere una sola línea de código en el Frontend.

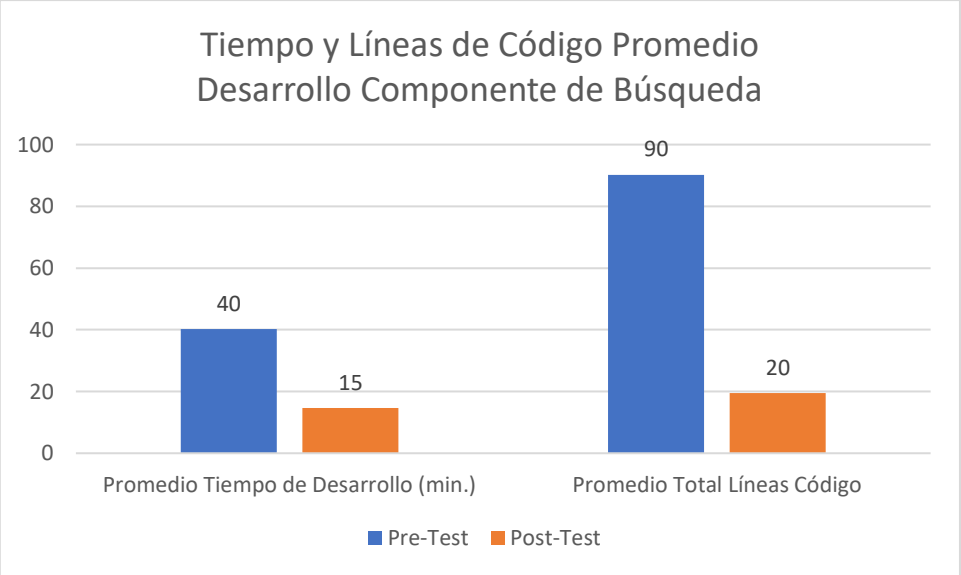
Gráfico 24. Promedio de líneas de código en el Backend para el desarrollo del componente de búsqueda de información



Fuente: Elaboración propia

El Gráfico 23, elaborado a partir de la Tabla 9, muestra las líneas de código necesarias para desarrollar el componente de búsqueda de información, antes y después de aplicar el componente SigaBandbox. Se puede apreciar que las líneas de código se reducen drásticamente, siendo necesarias entre 18 y 19 líneas en promedio para implementar cualquier componente de búsqueda de información, sin importar su complejidad.

Gráfico 25. Tiempo y líneas de código promedio en el desarrollo de un componente de búsqueda de información



Fuente: Elaboración propia

El Gráfico 24 muestra una comparativa entre las líneas de código y el tiempo de desarrollo de un componente de búsqueda de información, antes y después de aplicar el experimento (componente SigaBandbox). Se nota claramente la eficiencia del uso del componente SigaBandbox desarrollado en el presente trabajo de investigación. Un dato importante es que la complejidad del requerimiento puede aumentar, pero el uso del componente SigaBandbox siempre requerirá un promedio de 20 líneas de código entre el Frontend y Banckend. De igual manera, el tiempo de desarrollo se mantiene bajo y no aumenta significativamente.

4.2. Análisis de Resultados

Después de obtener los resultados de la investigación realizada, consistente en la implementación de componente de software para mejorar el desarrollo de los aplicativos web en el ministerio de economía y finanzas – lima; 2020, se realizó el presente análisis:

Con respecto al indicador Tiempo de Desarrollo, pre-test, se puede observar que aumenta en relación con la complejidad del requerimiento, bien por el número de columnas de datos o si muestra filtros o no por cada columna. Si lo comparamos con el tiempo de desarrollo post-test, vemos que este es considerablemente menor. Si tomamos en cuenta los tiempos promedios, el componente SigaBandbox desarrollado permite reducir a un 36.2% del tiempo de desarrollo original, variando ligeramente al aumentar la complejidad del requerimiento. Estos resultados nos demuestran la utilidad del uso del componente implementado. Cáceres (6) en su tesis "Componente Software para Mejorar el Acceso a las Bases de Datos Distribuida, software Middleware, 2016", en la ciudad de Ayacucho, demuestra que los tiempos de acceso a datos de los softwares que utilizan el componente software Middleware SQL es significativamente inferior a los que utilizan la técnica de Framework Hibernate. En ambos casos la implementación y uso de los componentes desarrollados ha permitido reducir el tiempo de desarrollo y acceso a datos respectivamente.

Igualmente, López (8) en su estudio "Diseño e Implementación de un Componente en la Plataforma .NET bajo la Metodología SCRUM para la Creación y Modificación de Planos Mediante la Teoría de Grafos", en la ciudad de Lima, concluye que el componente reduce significativamente el tiempo que toma la creación de planos eléctricos. Dichos tiempos fueron reducidos de aproximadamente de diez minutos a una hora que le toma a un operario humano de acuerdo con su pericia reducirlo a menos de veinte segundos. Por tanto, el estudio de López también está alineado a lo demostrado en el presente trabajo de investigación.

Respecto a la complejidad en líneas de código, el uso del componente SigaBandbox reduce a un 27.1% las líneas escritas con respecto al código original. Es interesante notar que para el caso del Frontend, el componente SigaBandbox siempre requiere una sola línea de código sin importar la

complejidad del requerimiento; en el caso del Backend solo requiere un promedio de 19 líneas de código. Palomino (5) en su tesis “Componente Software para Controlar las Versiones del Esquema de Base de Datos Relacional, AHREN Contratistas Generales SAC - Ayacucho, 2017” demuestra también que las líneas de código implementado para la creación y configuración del esquema de base de datos relacional de las aplicaciones que utiliza el componente software para controlar las versiones del esquema de base de datos relacional es significativamente inferior a los que utilizan los deltas scripts. Palomino también nos indica que los deltas scripts, que son scripts en lenguaje SQL, que resultan de la comparación de dos estados de una base de datos permitiendo resumir los cambios que ocurrieron entre los dos estados y de ser necesarios aplicarlos para igualar ambos esquemas de base de datos relacional. Es claro pues, que en ambos estudios la aplicación del componente desarrollado e implementado ha reducido la cantidad de líneas de código requeridas.

En cuanto a la variable Implementación de Componentes de Software para Mejorar el Desarrollo de los Aplicativos Web ha sido llevada a término, toda vez que el componente fue desarrollado, implementado y puesto a prueba en el proceso Buscar Información del aplicativo SIGAWEB; esto se demuestra con la creación del componente de software SigaBandox. Por otro lado, y en comparativa con los estudios antecedentes, se demuestra que el uso de compontes de software puede tener un uso muy variado, pero que en todos los casos busca solucionar un problema determinado.

V. Conclusiones y recomendaciones

5.1. Conclusiones

1. Se concluye que la Implementación de componentes de software mejora el desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima; 2020;

2. Se identificaron los procesos de desarrollo de software a cambiar a modelo basado en componentes mediante revisión de las diferentes opciones del Módulo de Patrimonio del SIGAWEB, desarrollado por la Oficina General de Tecnologías de la Información del ministerio de Economía y Finanzas, seleccionando el proceso Buscar Información.

3. Se midió el tiempo de desarrollo del proceso Buscar Información, del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima; 2020 antes de implementar el componente de software.

4. Se determinó la complejidad de desarrollo del proceso Buscar Información, del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima; 2020 antes de implementar el componente de software.

5. Se implementó el componente de software SigaBandbox para el proceso Buscar Información para reducir el tiempo y complejidad de desarrollo del aplicativo SIGAWEB del Ministerio de Economía y Finanzas – Lima; 2020.

6. Se comprobó y contrastó el tiempo y complejidad de desarrollo del proceso Buscar Información del aplicativo SIGAWEB del Ministerio

de Economía y Finanzas - Lima; 2020 después de implementar el componente de software SigaBandbox, obteniendo una reducción promedio del tiempo de desarrollo de 36.2% del tiempo original y disminución de la complejidad, expresada en líneas de código, del 21.7% con respecto al código original. Así mismo se determina que requerimientos más complejos en el proceso Buscar Información, más columnas de búsqueda y filtros, no incrementa las líneas de código y el tiempo de desarrollo varía ligeramente.

5.2. Recomendaciones

A fin de seguir mejorando los procesos de desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima, se recomienda lo siguiente:

1. Aplicar el componente SigaBandbox, para el proceso Buscar Información, en los nuevos desarrollos que lo requieran, de los diferentes módulos del SIGAWEB Ministerio de Economía y Finanzas – Lima; así mismo, capacitar al personal de desarrollo para su correcto uso.

2. Revisar e identificar procesos complejos y con tiempo de desarrollo elevado, que pueden ser convertidos en componentes a fin de seguir mejorando los procesos de desarrollo de los aplicativos web en el Ministerio de Economía y Finanzas – Lima.

3. Actualizar los frameworks de desarrollo a las versiones más recientes empleados en el SIGAWEB, en especial el Framework ZK, que es el que usa el componente SigaBandbox y evitar problemas de seguridad.

4. Adoptar el desarrollo basado en componentes, toda vez que sea posible, evitando así la redundancia de código y reduciendo los tiempos y complejidad de desarrollo.

Referencias bibliográficas

1. Finanzas MdEy. Ministerio de Economía y Finanzas - Perú. [Online]. [cited 2021 Enero 17. Available from: https://www.mef.gob.pe/es/?option=com_content&language=es-ES&Itemid=100558&lang=es-ES&view=article&id=2032.

2. Hincapié Ortiz BA, Pinto Ríos WD. Análisis y Prototipado de un Componente de Software de Exploración de Datos, Integrado a la Arquitectura de Visualización utilizando Dashboards. Tesis de Maestría. Pereira: Universidad Tecnólogica de Pereira, Facultad de Ingenierías: Eléctrica, Electrónica, Física y Ciencias de la Computación; 2019.

3. Cazalilla Morenas J. Diseño e implementación de un sistema de. Tesis Doctoral. Valencia: Universidad Politécnica de Valencia, España, Departamento de Ingeniería de Sistemas y Automática (ISA); 2017.

4. Silvera Charris AR, Sayas Arrieta I. Elaboración de un Componente de Software para el Cálculo de Métricas de Diseño a partir de XMI. Tesis de Grado. Cartagena: Universidad de Cartagena, Programa de Ingeniería de Sistemas; 2016.

5. Palomino Pariona A. Componente Software para Controlar las Versiones del Esquema de Base de Datos Relacional, AHREN Contratistas Generales SAC - Ayacucho, 2017. Tesis de Grado. Ayacucho: Universidad Nacional de San Cristóbal de Huamanga, Escuela Profesional de Ingeniería de Sistemas; 2018.

6. Cáceres Curo A. Componente Software para Mejorar el Acceso a las Bases de Datos Distribuida, Software Middleware, 2016. Tesis de Grado. Ayacucho: Universidad Nacional de San Cristóbal de Huamanga, Escuela de Formación Profesional de Ingeniería de Sistemas; 2016.

7. Aroni Urday RE. Desarrollo de un Componente de Software COM+ para Monitoreo y Control de un Equipo Informático bajo la Plataforma.NET. Tesis de Grado. Juliaca: Universidad Andina Néstor Cáceres Velásquez, Facultad de Ingeniería de Sistemas; 2015.

8. López Enciso DA. Diseño e Implementación de un Componente en la Plataforma.NET bajo la Metodología SCRUM para la Creación y Modificación de Planos Mediante la Teoría de Grafos. Tesis de Grado. Lima: Universidad Tecnológica del Perú, Facultad de Ingeniería de Sistemas; 2018.

9. Castro Vásquez JM, Exebio Fernández HA. Métodos y Herramientas de Trabajo para Arquitecturas Basado en Componentes de Desarrollo de Software: Una revisión Sistemática de la Literatura. Tesis de Grado. Lima: Universidad Peruana Unión, Facultad de Ingenieria y Arquitectura; 2019.

10. Rojas Herencia E. Póliza Electrónica usando Componente Backend para Reducir Costos de Despachos en Rimac Seguros y Reaseguros. Tesis de Grado. Lima: Universidad Nacional Mayor de San Marcos, Escuela Profesional de Ingeniería de Sistemas; 2018.

11. Carbajal Montesinos H. Construcción de un Componente de Software para la Búsqueda del Camino más Corto y el Control de Movimiento en un Videojuego de Estrategia en Tiempo Real. Tesis de Grado. Lima: Pontificia Universidad Católica del Perú, Facultad de Ciencias e Ingeniería; 2013.

12. Tomás Pascual GJ. Implementación del Sub Módulo de Pedidos de Transferencia en el Sistema Informático SIGA-Web del Ministerio de Economía y Finanzas usando el Framework ZK, 2016. Tesis. Chimbote: Universidad Católica Los Ángeles de Chimbote, Escuela Profesional de Ingeniería de Sistemas; 2016.

13. Sommerville I. Ingeniería del Software. Novena ed. Cruz Castillo LM, editor.: Pearson Educación de México, S.A. de C.V.; 2011.

14. Heineman T, Councill B. Component-Based Software Engineering: Putting the Pieces Together. Primera ed.: Addison-Wesley Professional; 2001.

15. Szyperski C. Component Software. Segunda ed.: Pearson Education Limited; 2002.

16. Deitel P, Deitel H. Java Cómo Programar. Novena ed. Deitel PDyH, editor.: Pearson Education, Inc.; 2012.

17. Org Z. ZK Framework Documentation. [Online].; 2021 [cited 2021 Enero 25. Available from: https://www.zkoss.org/documentation.

18. Mateu C. Desarrollo de Aplicaciones Web. Primera ed. Barcelona: Fundació per a la Universitat Oberta de Catalunya; 2004.

19. Emfasi Comunicació Digital, S.L. emfasi. [Online]. [cited 2021 Febrero 06. Available from: https://www.emfasi.com/desarrollo-de-aplicaciones-web.

20. Tébar E. We are Marketing. [Online].; 2020 [cited 2021 Febrero 08. Available from: https://www.wearemarketing.com/es/blog/frameworks-en-el-desarrollo-web-las-mejores-practicas-para-tu-negocio-online.html#.

21. Pressman R. Ingeniería del Software. Un Enfoque Práctico. Séptima ed. Vásquez PR, editor. México: McGraw-Hill Interamericana Editores, S.A.; 2010.

22. Alvarez MA. Desarrolloweb. [Online]. [cited 2021 Febrero 08. Available from: https://desarrolloweb.com/articulos/que-es-mvc.html.

23. Cabezas Mejía ED, Andrade Naranjo D, Torres Santamaría J. Introducción a la Metodología de la Investigación Científica. Primera ed. Aguirre DA, editor. Sangolquí: Universidad de las Fuerzas ESPE; 2018.

24. Murillo Hernandez WJ. monografias.com. [Online]. [cited 2021 Enero 4. Available from: https://www.monografias.com/trabajos15/invest-cientifica/invest-cientifica.shtml.

25. Ramírez González A. postgradoune. [Online].: Pontificia Universidad Javeriana [cited 2021 Enero 8. Available from: https://www.postgradoune.edu.pe/pdf/documentos-academicos/ciencias-de-la-educacion/1.pdf.

26. Murillo J. Universidad Nacional de Educación Enrique Guzmán y Valle. [Online]. [cited 2021 Enero 9. Available from: https://www.postgradoune.edu.pe/pdf/documentos-academicos/ciencias-de-la-educacion/10.pdf.

27. Escuela Profesional de Ingeniería de Sistemas. ULADECH. [Online].; 2008 [cited 2021 Enero 10. Available from: http://files.uladech.edu.pe/docente/32793925/DEONTOLOGIA/Contenidos%20Unidad%201/Caracterizacion_Ing_Sistemas.pdf.

ANEXOS

Anexo 1:

CRONOGRAMA DE ACTIVIDADES									
Nº	Actividades	Año 2020				Año 2021			
		Semestre II				Semestre I			
		Mes				Mes			
		1	2	3	4	1	2	3	4
1	Elaboración del Proyecto								
2	Revisión del proyecto por el Jurado de Investigación								
3	Aprobación del proyecto por el Jurado de Investigación								
4	Exposición del proyecto al Jurado de Investigación por el Docente Tutor								
5	Mejora del marco teórico								
6	Redacción de la revisión de la literatura								
7	Ejecución de la metodología								
8	Resultados de la investigación								
9	Conclusiones y recomendaciones								
10	Redacción del Pre-Informe de Investigación								
11	Redacción del Informe Final								
12	Aprobación del informe final por el Jurado de Investigación								
13	Presentación de ponencia en eventos científicos								
14	Redacción del artículo científico								

Anexo 2: Presupuesto

Presupuesto desembolsable (Estudiante)			
Categoría	**Base**	**% o Número**	**Total (S/)**
Bienes			
• Laptop	2500.00	1	2500.00
• Mobiliario	350.00	1	350.00
• Impresora	750.00	1	750.00
Sub total			**3,600.00**
Suministros			
• Impresiones	0.30	600	180.00
• Fotocopias	0.10	600	60.00
• Empastado	80.00	4	320.00
• Papel bond A-4 (500 hojas)	13.00	4	52.00
Sub total			**612.00**
Servicios			
• Uso de Turnitin	50.00	2	100.00
Sub total			**100.00**
Gastos de viaje			
• Pasajes para recolectar información	60	4	240.00
Sub total			**240.00**
Total de presupuesto desembolsable			**4,552.00**

Presupuesto desembolsable (Universidad)			
Categoría	**Base**	**% o Número**	**Total (S/)**
Servicios			
• Uso de Internet (Laboratorio de aprendizaje Digital - LAD)	30.00	4	120.00
• Búsqueda de información en base de datos	35.00	2	70.00
• Soporte informático (Módulo de Investigación del ERP University - MOIC)	40.00	4	160.00
• Publicación de artículo en repositorio institucional	50.00	1	50.00
Sub total			**400.00**
Recurso humano			
• Asesoría personalizada (5 horas por semana)	63.00	4	252.00
Sub total			**252.00**
Total de presupuesto no desembolsable			**652.00**

Total (S/)			**5,204.00**

Anexo 3: Instrumento de Recolección de Datos

NOTA DE CAMPO Nº_____			
Datos Generales			
Institución:			**Fecha:**
Observador:			
Aplicativo Informático:			
Módulo del Aplicativo:			
Opción del Aplicativo:			
Nombre de la Ventana:			
Campo:			
Nº Columnas:		**Con filtros:**	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)			
Con el componente (Post-test)			

Anexo 4: Validación del instrumento

UNIVERSIDAD CATOLICA LOS ANGELES DE CHIMBOTE
ESCUELA DE POSGRADO

FICHA DE EVALUACION DEL INSTRUMENTO

1.1. Nombres y apellidos del validador: Jorge Lennin Peña Chauca
1.2. Cargo e institución donde labora: coordinador soporte y Equip. FNF.
1.3. Nombre del instrumento evaluado: Nota de campo
1.4. Autor del instrumento: Bra. Guido Jhonny Tomás Pascual

II. ASPECTOS DE LA EVALUACIÓN

Revisar cada uno de los ítems del instrumento y marcar con un aspa dentro del recuadro (x) según la calificación que asigne a cada uno de los indicadores

1. Deficiente (Si menos del 30% de los ítems cumple con el indicador)
2. Regular (Si entre el 31% y 70% de los ítems cumplen con el indicador)
3. Buena (Si más del 70% de los ítems cumple con el indicador)

Aspectos de validación del Instrumento		1	2	3	Observaciones Sugerencias
Criterios	**Indicadores**	**D**	**R**	**B**	
Pertinencia	Los ítems miden lo previsto en los objetivos de investigación.			X	
Coherencia	Los ítems responden a lo que se debe medir en la variable y sus dimensiones.			X	
Congruencia	Los ítems son congruentes entre sí y con el concepto que mide.			X	
Suficiencia	Los ítems son suficientes en cantidad para medir la variable.			X	
Objetividad	Los ítems se expresan en comportamientos y acciones observables.		X		
Consistencia	Los ítems se han formulado en concordancia a los fundamentos teóricos de la variable.			X	
Organización	Los ítems están secuenciados y distribuidos de acuerdo a dimensiones e indicadores.			X	
Claridad	Los ítems están redactados en un lenguaje entendible para los sujetos a evaluar.			X	
Formato	Los ítems están escritos respetando aspectos técnicos (tamaño de letra, espaciado, interlineado, nitidez)			X	
Estructura	El instrumento cuenta con instrucciones, consignas, opciones de respuestas bien definidas.		X		
	CONTEO TOTAL		4	24	28
	(Realizar el conteo de acuerdo a las puntuaciones asignadas a cada indicador)	C	B	A	Total

Coeficiente De validez

$$\frac{A+B+C}{30} = 0.93$$

Intervalos	Resultados
0.00 – 0.49	Validez Nula
0.50 – 0.59	Validez muy baja
0.60 – 0.69	Validez baja
0.70 – 0.79	Validez aceptable
0.80 – 0.89	Validez buena
0.90 – 1.00	Validez muy buena

III. CALIFICACIÓN GLOBAL

Ubicar el coeficiente de validez obtenido en el intervalo respectivo y escriba sobre el espacio el resultado.

Validez muy buena

Chimbote, enero del 2021

PEÑA CHAUCA JORGE LENNIN
ING. DE SISTEMAS
Reg. Colegio de Ingenieros CIP N° 168119

UNIVERSIDAD CATOLICA LOS ANGELES DE CHIMBOTE
ESCUELA DE POSGRADO

FICHA DE EVALUACION DEL INSTRUMENTO

1.1. Nombres y apellidos del validador: Noé Gregorio Silva Zelada
1.2. Cargo e institución donde labora: Coord. TI - Docente - ULADECH
1.3. Nombre del instrumento evaluado: Nota de campo
1.4. Autor del instrumento: Ing. Guido Jhonny Tomás Pascual

II. **ASPECTOS DE LA EVALUACIÓN**

Revisar cada uno de los ítems del instrumento y marcar con un aspa dentro del recuadro (x) según la calificación que asigne a cada uno de los indicadores

1. Deficiente (Si menos del 30% de los ítems cumple con el indicador)
2. Regular (Si entre el 31% y 70% de los ítems cumplen con el indicador)
3. Buena (Si más del 70% de los ítems cumple con el indicador)

Aspectos de validación del Instrumento		1	2	3	Observaciones Sugerencias
Criterios	**Indicadores**	**D**	**R**	**B**	
Pertinencia	Los ítems miden lo previsto en los objetivos de investigación.			X	
Coherencia	Los ítems responden a lo que se debe medir en la variable y sus dimensiones.			X	
Congruencia	Los ítems son congruentes entre sí y con el concepto que mide.			X	
Suficiencia	Los ítems son suficientes en cantidad para medir la variable.			X	
Objetividad	Los ítems se expresan en comportamientos y acciones observables.		X		
Consistencia	Los ítems se han formulado en concordancia a los fundamentos teóricos de la variable.			X	
Organización	Los ítems están secuenciados y distribuidos de acuerdo a dimensiones e indicadores.			X	
Claridad	Los ítems están redactados en un lenguaje entendible para los sujetos a evaluar.		X		
Formato	Los ítems están escritos respetando aspectos técnicos (tamaño de letra, espaciado, interlineado, nitidez)		X		
Estructura	El instrumento cuenta con instrucciones, consignas, opciones de respuestas bien definidas.		X		
	CONTEO TOTAL		8	18	26
	(Realizar el conteo de acuerdo a las puntuaciones asignadas a cada indicador)	C	B	A	Total

Coeficiente De validez: $\frac{A+B+C}{30} = 0.86$

Intervalos	Resultados
0.00 – 0.49	Validez Nula
0.50 – 0.59	Validez muy baja
0.60 – 0.69	Validez baja
0.70 – 0.79	Validez aceptable
0.80 – 0.89	Validez buena
0.90 – 1.00	Validez muy buena

III. **CALIFICACIÓN GLOBAL**

Ubicar el coeficiente de validez obtenido en el intervalo respectivo y escriba sobre el espacio el resultado.

Validez buena

Chimbote, enero del 2021

Mg. Noé Gregorio Silva Zelada
Ingeniero Informático y de Sistemas
CIP: 83347

UNIVERSIDAD CATOLICA LOS ANGELES DE CHIMBOTE
ESCUELA DE POSGRADO

FICHA DE EVALUACION DEL INSTRUMENTO

1.1. Nombres y apellidos del validador: Oscar Arquimedes Ascón Valdivia
1.2. Cargo e institución donde labora: Docente
1.3. Nombre del instrumento evaluado: Nota de Campo
1.4. Autor del instrumento: Ing. Guido Jhonny Tomás Pascuas

II. **ASPECTOS DE LA EVALUACIÓN**

Revisar cada uno de los ítems del instrumento y marcar con un aspa dentro del recuadro (x) según la calificación que asigne a cada uno de los indicadores

1. Deficiente (Si menos del 30% de los ítems cumple con el indicador)
2. Regular (Si entre el 31% y 70% de los ítems cumplen con el indicador)
3. Buena (Si más del 70% de los ítems cumple con el indicador)

Aspectos de validación del Instrumento		**1**	**2**	**3**	**Observaciones Sugerencias**
Criterios	**Indicadores**	**D**	**R**	**B**	
Pertinencia	Los ítems miden lo previsto en los objetivos de investigación.			X	
Coherencia	Los ítems responden a lo que se debe medir en la variable y sus dimensiones.			X	
Congruencia	Los ítems son congruentes entre sí y con el concepto que mide.			X	
Suficiencia	Los ítems son suficientes en cantidad para medir la variable.		X		
Objetividad	Los ítems se expresan en comportamientos y acciones observables.		X		
Consistencia	Los ítems se han formulado en concordancia a los fundamentos teóricos de la variable.			X	
Organización	Los ítems están secuenciados y distribuidos de acuerdo a dimensiones e indicadores.			X	
Claridad	Los ítems están redactados en un lenguaje entendible para los sujetos a evaluar.			X	
Formato	Los ítems están escritos respetando aspectos técnicos (tamaño de letra, espaciado, interlineado, nitidez)			X	
Estructura	El instrumento cuenta con instrucciones, consignas, opciones de respuestas bien definidas.		X		
	CONTEO TOTAL		6	21	27
	(Realizar el conteo de acuerdo a las puntuaciones asignadas a cada indicador)	C	B	A	Total

Coeficiente De validez

$$\frac{A+B+C}{30} = 0.9$$

Intervalos	Resultados
0.00 – 0.49	Validez Nula
0.50 – 0.59	Validez muy baja
0.60 – 0.69	Validez baja
0.70 – 0.79	Validez aceptable
0.80 – 0.89	Validez buena
0.90 – 1.00	Validez muy buena

III. **CALIFICACIÓN GLOBAL**

Ubicar el coeficiente de validez obtenido en el intervalo respectivo y escriba sobre el espacio el resultado.

Validez muy buena

Ing. Oscar Ascón Valdivia
DNI: 32754949 CIP: 76142

Chimbote, enero del 2021

Anexo 4: Notas de Campo Recolectadas

NOTA DE CAMPO Nº 001			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/21
Observador: Liliana Mantilla Mostacero			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Inventario Inicial Institucional			
Nombre de la Ventana: Registro de Ingreso			
Campo: Centro de Costo			
Nº Columnas: 2		**Con filtros:** No	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	25	25	18
Con el componente (Post-test)	12	1	18

Liliana Mantilla Mostacero

NOTA DE CAMPO Nº 002			
Datos Generales			
Institución: Ministerio de Economía y Finanzas		**Fecha:** 12/01/21	
Observador: Liliana Mantilla Mostacero			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Inventario Inicial Institucional			
Nombre de la Ventana: Registro de Ingreso			
Campo: Empleado Responsable			
Nº Columnas: 2		**Con filtros:** No	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	27	31	20
Con el componente (Post-test)	18	1	18

Liliana Mantilla Mostacero

NOTA DE CAMPO Nº 003			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/21
Observador: Michael Palacios Curi			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Inventario Inicial Institucional			
Nombre de la Ventana: Datos del Activo Fijo			
Campo: Ítem			
Nº Columnas: 5		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	55	55	07
Con el componente (Post-test)	20	1	19

Michael Palacios Curi

NOTA DE CAMPO Nº 004			
Datos Generales			
Institución: Ministerio de Economía y Finanzas		**Fecha:** 12/01/21	
Observador: Michael Palacios Curi			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Inventario Inicial Institucional			
Nombre de la Ventana: Datos del Activo Fijo			
Campo: Ubic. Física			
Nº Columnas: 3		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	75	34	106
Con el componente (Post-test)	15	1	19

Michael Palacios Curi

NOTA DE CAMPO Nº 005			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/21
Observador: Félix Castillo Tafur			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Inventario Inicial Institucional			
Nombre de la Ventana: Datos del Activo Fijo			
Campo: Usuario Final			
Nº Columnas: 5		**Con filtros:** No	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	25	43	26
Con el componente (Post-test)	17	1	18

Félix Castillo Tafur

NOTA DE CAMPO Nº 006	
Datos Generales	
Institución: Ministerio de Economía y Finanzas	**Fecha:** 12/01/21
Observador: Félix Castillo Tafur	
Aplicativo Informático: SIGAWEB	
Módulo del Aplicativo: Patrimonio	
Opción del Aplicativo: Inventario Inicial Institucional	
Nombre de la Ventana: Datos del Activo Fijo	
Campo: Proveedor	
Nº Columnas: 6	**Con filtros:** No

Datos del Experimento

Etapa	Tiempo de Desarrollo (min.)	Nº Líneas Código Frontend	Nº Líneas Código Backend
Sin el Componente (Pre-test)	~~25~~ 32	49	12
Con el componente (Post-test)	19	1	18

Félix Castillo Tafur

NOTA DE CAMPO
N° 007

Datos Generales

Institución: Ministerio de Economía y Finanzas	**Fecha:** 12-01-2021
Observador: Hernando Montes Rosales	
Aplicativo Informático: SIGAWEB	
Módulo del Aplicativo: Patrimonio	
Opción del Aplicativo: Inventario Inicial Institucional	
Nombre de la Ventana: Datos del Activo Fijo	
Campo: Almacén	
N° Columnas: 3	**Con filtros:** No

Datos del Experimento

Etapa	Tiempo de Desarrollo (min.)	N° Líneas Código Frontend	N° Líneas Código Backend
Sin el Componente (Pre-test)	27	31	22
Con el componente (Post-test)	13	1	18

Hernando Montes Rosales

NOTA DE CAMPO
Nº 008

Datos Generales

Institución:	**Fecha:**
Ministerio de Economía y Finanzas	12-1-2021

Observador: Hernando Montes Rosales

Aplicativo Informático: SIGAWEB

Módulo del Aplicativo: Patrimonio

Opción del Aplicativo: Bajas

Nombre de la Ventana: Baja de Activos

Campo: Familia

Nº Columnas: 2	**Con filtros:** Si

Datos del Experimento

Etapa	Tiempo de Desarrollo (min.)	Nº Lineas Código Frontend	Nº Lineas Código Backend
Sin el Componente (Pre-test)	45	26	51
Con el componente (Post-test)	11	1	19

Hernando Montes Rosales

NOTA DE CAMPO Nº 009			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/21
Observador: David Yauris Huayta			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Salidas			
Nombre de la Ventana: Mantenimiento de Salida de Bienes			
Campo: N° O/C			
Nº Columnas: 5		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	65	58	97
Con el componente (Post-test)	15	1	19

David Yauris Huayta

NOTA DE CAMPO Nº 010			
Datos Generales			
Institución: Ministerio de Economía y Finanzas		**Fecha:** 12/01/21	
Observador: David Yauris Huayta			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Salidas			
Nombre de la Ventana: Mantenimiento de Salida de Bienes			
Campo: Sede			
Nº Columnas: 2		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	42	28	57
Con el componente (Post-test)	13	1	19

David Yauris Huayta

NOTA DE CAMPO Nº 011			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/21
Observador: Guido Tomás Pascual			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Salidas			
Nombre de la Ventana: Mantenimiento de Salida de Bienes			
Campo: Centro Costo Solicitante			
Nº Columnas: 2		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	30	28	56
Con el componente (Post-test)	10	1	19

Guido Tomás Pascual

NOTA DE CAMPO Nº 012			
Datos Generales			
Institución: Ministerio de Economía y Finanzas			**Fecha:** 12/01/24
Observador: Guido Tomás Pascual			
Aplicativo Informático: SIGAWEB			
Módulo del Aplicativo: Patrimonio			
Opción del Aplicativo: Salidas			
Nombre de la Ventana: Mantenimiento de Salida de Bienes			
Campo: Responsable			
Nº Columnas: 4		**Con filtros:** Si	
Datos del Experimento			
Etapa	**Tiempo de Desarrollo (min.)**	**Nº Líneas Código Frontend**	**Nº Líneas Código Backend**
Sin el Componente (Pre-test)	35	53	90
Con el componente (Post-test)	12	1	19

Guido Tomás Pascual

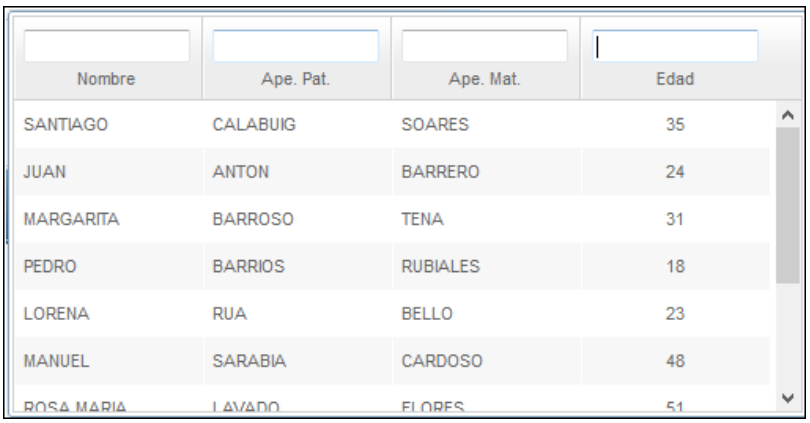
Anexo 5: Pruebas Unitarias

Se verificó la funcionalidad del componente visual **SigaBandbox** para un conjunto de datos de prueba.

Listado de datos de prueba

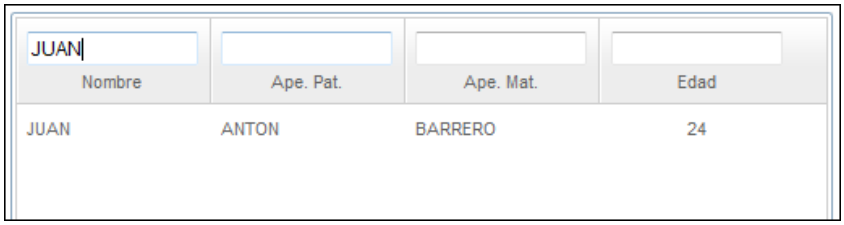
Nº	Nombres	Apellido Paterno	Apellido Materno	Edad
1	SANTIAGO	CALABUIG	SOARES	35
2	JUAN	ANTON	BARRERO	24
3	MARGARITA	BARROSO	TENA	31
4	PEDRO	BARRIOS	RUBIALES	18
5	LORENA	RUA	BELLO	23
6	MANUEL	SARABIA	CARDOSO	48
7	ROSA MARIA	LAVADO	FLORES	51
8	CRISTOBAL	ARRIAGA	COMESAÑA	27
9	JOSEFA	PEREIRA	PARDO	37
10	AITOR	PORTO	PORTILLO	42

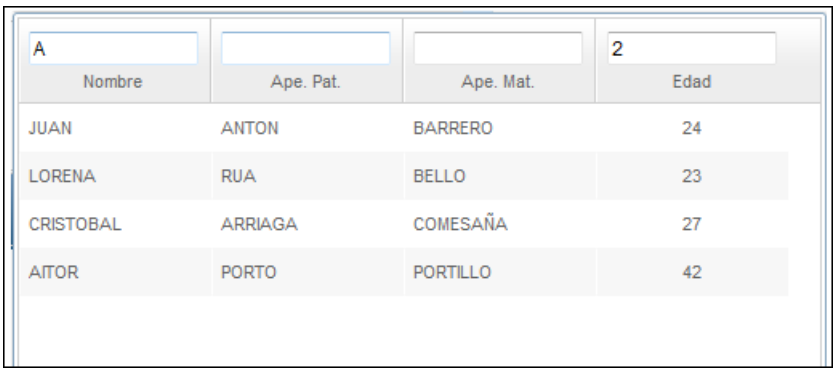
Prueba 01: Renderiza la lista de datos indicada



Se verifica los datos mostrados en el componente
Resultado: Correcto

Prueba 02: Filtra la información por una o más columnas





Se verifica el filtrado de información por uno o más campos
Resultado: Correcto

Prueba 03: Alinea los datos izquierda, centro, derecha, según requerimiento

Se verifica que los Nombres, Apellido Paterno y Materno se alinean a la izquierda según lo programado. La edad se centra.
Resultado: Correcto

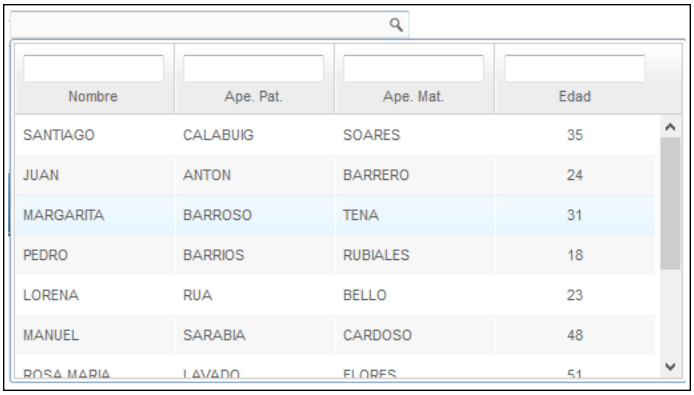
Prueba 04: Permite seleccionar un registro de la lista y lo muestra en la caja de texto

Se verifica que el registro seleccionado se muestra en la caja de texto del componente Bandbox.
Resultado: Correcto

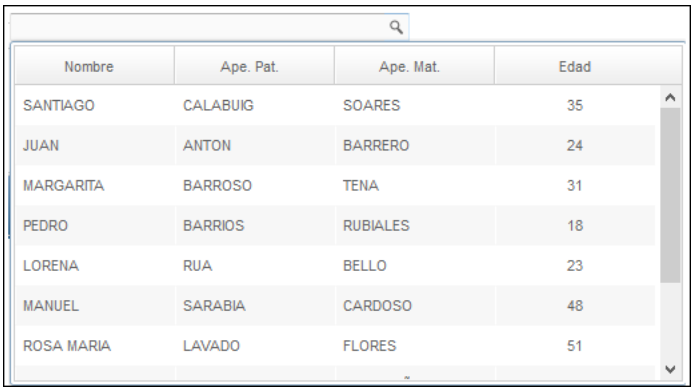
Prueba 05: Permite renderizar una o más columnas

Se verifica que el componente desarrollado permite renderizar una o más columnas de datos.
Resultado: Correcto

Prueba 06: Permite mostrar/ocultar la barra de búsqueda



Nombre	Ape. Pat.	Ape. Mat.	Edad
SANTIAGO	CALABUIG	SOARES	35
JUAN	ANTON	BARRERO	24
MARGARITA	BARROSO	TENA	31
PEDRO	BARRIOS	RUBIALES	18
LORENA	RUA	BELLO	23
MANUEL	SARABIA	CARDOSO	48
ROSA MARIA	LAVADO	FLORES	51



Nombre	Ape. Pat.	Ape. Mat.	Edad
SANTIAGO	CALABUIG	SOARES	35
JUAN	ANTON	BARRERO	24
MARGARITA	BARROSO	TENA	31
PEDRO	BARRIOS	RUBIALES	18
LORENA	RUA	BELLO	23
MANUEL	SARABIA	CARDOSO	48
ROSA MARIA	LAVADO	FLORES	51

Se verifica que la barra de búsqueda puede ocultarse o mostrarse.
Resultado: Correcto

Anexo 6: Manual de Uso del Componente

1. Consideraciones preliminares

Primeramente, debe existir un componente Bandbox definido en el archivo *.ZUL de la capa de presentación.

Ejemplo:

```
<hbox height="25px" align="center">
    <vbox width="420px">
        <bandbox id="bbxPrueba" width="300px" readonly="true"/>
    </vbox>
</hbox>
```

Importar la clase SigaBandox en el controlador que lo va a usar.

Ejemplo:

```
import pe.gob.mef.siga.web.util.SigaBandbox;
```

El ID del componente debe declararse como propiedad (variable) en la clase Controller asociada al *.ZUL

Ejemplo:

```
private Bandbox bbxPrueba;
```

Tener una lista de datos a renderizar. Puede ser lista de mapas o de una clase POJO.

Ejemplo:

```
List<Map<String, Object>> miLista = new ArrayList<Map<String, Object>>();
```

2. Declaración del componente SigaBandbox

Declarar e instanciar una variable global de tipo Sigabandox.

Ejemplo:

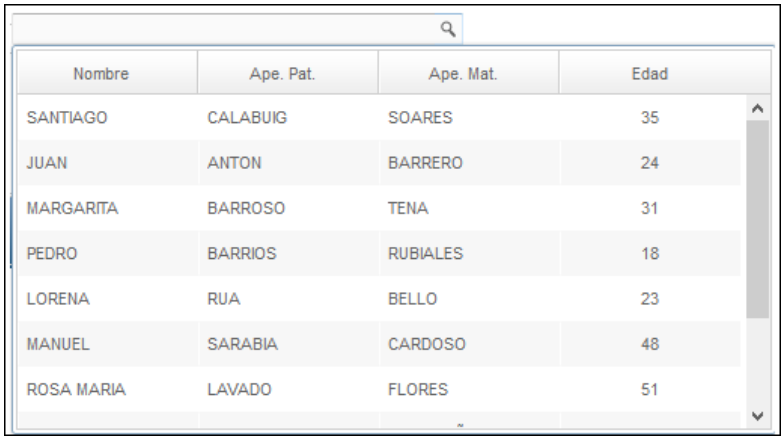
```
private SigaBandbox sbbPrueba = new SigaBandbox();
```

3. Creación de método para renderizar el componente

Es recomendable crear un método que encapsule los métodos que definen las características del componente a renderizar

Ejemplo:

Para renderizar el siguiente componente se debe hacer lo siguiente



Nombre	Ape. Pat.	Ape. Mat.	Edad
SANTIAGO	CALABUIG	SOARES	35
JUAN	ANTON	BARRERO	24
MARGARITA	BARROSO	TENA	31
PEDRO	BARRIOS	RUBIALES	18
LORENA	RUA	BELLO	23
MANUEL	SARABIA	CARDOSO	48
ROSA MARIA	LAVADO	FLORES	51

```
public void pintarSbbPrueba(){
    String[] columnas = {"dato1", "dato2", "dato3", "dato4"};
    String[] etiquetas = {"Nombre", "Ape. Pat.", "Ape. Mat.", "Edad"};
    String[] alineacion = {"left", "left", "left", "center",};
    int[] ancho ={100,100,100,100};
    this.sbbPrueba.setColumnas(columnas, etiquetas, ancho, 500);
    this.sbbPrueba.setAnchoAltoLista(500, 250);
    this.sbbPrueba.setAlinearColumnas(alineacion);
    this.sbbPrueba.setTextoOnSelect("dato1 dato2 dato3");
    this.sbbPrueba.pintar(this.bbxPrueba, miLista, new Callable()
        {public Object call(){return onSelectBandBox();}} );
}

public boolean onSelectBandBox(){
    // Código que se ejecuta al seleccionar un registro.
    Map<String,Object> miRegistro = (Map<String, Object>) this.sbbPrueba.getFilaSeleccionada();
    return true;
}
```

Donde:

columnas: Es un arreglo de cadenas que indica los campos a renderizar. Para el ejemplo hemos usado una lista de mapas Map<String, Object> donde dato1 (Nombres), dato2 (Ape. Pat.), dato3 (Ape. Mat.) y dato4 (Edad), son las llaves (campos) del mapa.

etiquetas: Es un arreglo de cadenas que indican los rótulos que se mostrarán como cabecera de columna.

alineación: Arreglo de cadenas que indica la alineación que tendrán los datos de cada columna: left (izquierda), center (centro), right (derecha).

ancho: Arreglo de enteros que indica el ancho que tendrá cada columna.

setColumnas: Método que indica al componente los datos de las columnas, alineación y ancho.

setAnchoAltoLista: Método que indica el ancho y alto en píxeles que tendrá la caja desplegable del componente.

setAlinearColumnas: Método que indica al componente la alineación que tendrán los datos de cada columna. Este método puede obviarse si no se desea una alineación determinada.

setTextoOnSelect: Este método indica que dato (columna) debe mostrarse al seleccionar un registro de la lista. Puede indicarse varias columnas concatenando los nombres de los campos dentro de una cadena, separados por espacios en blanco.

pintar: Este método deba llamarse al final, pues es el que renderiza el componente con todos los datos pasados con los métodos anteriores. En este método se ingresa como parámetros el nombre del Bandbox definido en el archivo ZUL, la lista de datos a renderizar y un método que servirá como evento al seleccionar un registro. Para el presente ejemplo se llama onSelectBandBox().

onSelectBandBox(): Es un método que es invocado al momento de seleccionar un registro. Dentro de él se debe capturar el registro seleccionado y asignado a una variable para su posterior uso.

4. Consideraciones finales

Los pasos anteriormente indicados son los necesarios para el renderizado de un componente típico de búsqueda de información. Existen otros métodos que pueden ser consultados en las especificaciones técnicas de la clase SigaBandbox

Anexo 7: Código Fuente del Componente SigaBandobox

```
/*************************************************************
 * Clase personalizada para la generación de Bandbox.<br><br>
 * <b>Ejem.</b>
 *
 * <pre>
 * private SigaBandbox miBandbox;
 * private List lista2; //Esta lista puede ser de cualquier clase.
 *
 * public void pintarBandbox(){
 *    this.miBandbox = new SigaBandbox();
 *    String[] columnas = {"id", "nombre", "edad", "direccion", "sueldo"};
 *    String[] etiquetas = {"ID", "Nombre", "Edad", "Dirección", "Sueldo"};
 *    int[] ancho ={40,300,50,450,120};
 *    this.miBandbox.setColumnas(columnas, etiquetas, ancho, 450);
 *    this.miBandbox.setTextoOnSelect("id - nombre");
 *        this.miBandbox.pintar(this.bbxDemo, this.lista2, new Callable(){public Object call(){return
onSelectBandBox();}} );
 * }
 *
 * public boolean onSelectBandBox(){
 *    // Código que se ejecuta al seleccionar un registro.
 *    return true;
 * }
 * </pre>
 *
 * @author gtomas
 * @since 04/01/2021
 *
 *  ************************************************************/
public class SigaBandbox {
    protected final Log logger = LogFactory.getLog(getClass());
    private String[] columnas;
    private String[] etiquetas;
    private int[] anchos;

    private boolean error = false;
    private Listbox miListbox = new Listbox();
    private Bandpopup miBandpopup = new Bandpopup();
    private List<Textbox> listaTextbox = new ArrayList<Textbox>();
    private List<Object> listaOriginal;
    private Object filaSeleccionada;
    private Object filaSeleccionadaAnterior;
    private String[] pintarOnSelect;
    private String[] selCamposOnRender;
    private String[] selValoresOnRender;
    private String tipoClase;
    private Integer indiceListboxAnterior;
    private Bandbox miBB;

    /*************************************************************
     * Especifica las columnas a visualizar.
```

```
*
* @author gtomas
* @since 04/01/2021
* @param columna : Arreglo de tipo String <b>String[]</b> para
* indicar las columnas a visualizar. En caso se requiera que una
* columna este compuesta por más de un campo se deberá colocar
* dentro del arreglo dejando un espacio en blanco. <b>Ejemplo:
* {"campo1 campo2 campo3", "columna2"}</b>
* @param etiqueta : Arreglo de tipo String <b>String[]</b> para
* indicar las etiquetas (rótulos) de las columnas.
* @param ancho : Arreglo de tipo int <b>int[]</b> para
* indicar en ancho de las columnas.
* @param anchoLista : Indica el ancho de la lista. Para mostrar
* todas las columnas, se recomienda que este valor sea la suma
* del ancho te toda las columnas mostradas, más 80 pixeles.
**************************************************************/
@SuppressWarnings({ "unchecked", "rawtypes" })
public void setColumnas(String[] columna, String[] etiqueta, int[]ancho, int anchoLista){
    /*--- Validamos que los arreglos tengan la misma longitud ---*/
    if(columna.length != etiqueta.length || columna.length != ancho.length){
        Messagebox.show("ERROR: Verifique el tamaño de los arreglos.");
        return;
    }
    this.columnas = columna;
    this.etiquetas = etiqueta;
    this.anchos = ancho;

    if(miListbox.getChildren().size() == 0){
        Listhead miListhead = new Listhead();

        for(int i=0; i<this.columnas.length; i++){
            Listheader miListheader = new Listheader();
            miListheader.setWidth(anchos[i]+20+"px");
            Vbox miVbox = new Vbox();
            miVbox.setWidth(anchos[i]+"px");
            miVbox. setAlign("center");
            final Textbox miTextbox = new Textbox();
            miTextbox.addEventListener("onOK",  new EventListener(){
                @Override
                public void onEvent(Event event) throws Exception {
                    filtrarLista((Integer)miTextbox.getAttribute("indice"));
                }
            });
            miTextbox.setWidth(anchos[i]+"px");
            miTextbox.setAttribute("indice", i);
            this.listaTextbox.add(miTextbox);
            Label miLabel = new Label(etiquetas[i]);
            miVbox.appendChild(miTextbox);
            miVbox.appendChild(miLabel);
            miListheader.appendChild(miVbox);
            miListhead.appendChild(miListheader);
        }
        this.miListbox.setWidth(anchoLista+"px");
        this.miListbox.appendChild(miListhead);
    }
```

```
    }

    /*****************************************************************
     * Setea la alineación de los datos en las columnas. El método
     * debe ser llamado después de <b>setColumnas()</b>
     * @author gtomas
     * @since 04/01/2021
     * @param alineacion : Arreglo de cadenas String[]. Para indicar
     * la alineación usar:<br>
     * "left"   = Alineado a la izquierda<br>
     * "center" = centrado<br>
     * "right"  = Alineado a la derecha
     *
     *****************************************************************/
    public void setAlinearColumnas(String[] alineacion){
        for(int i=0; i<this.miListbox.getChildren().get(0).getChildren().size(); i++){
            Listheader miListheader = (Listheader)
this.miListbox.getChildren().get(0).getChildren().get(i);
            miListheader.setAlign(alineacion[i]);
        }
    }

    /*****************************************************************
     * Establece si las cajas de búsquedas son visibles o no.
     * @author gtomas
     * @since 04/01/2021
     * @param visible : true=Visible   false=Invisible.
     *
     *****************************************************************/
    public void setBusquedaVisible(boolean visible){
        for(int i=0; i<this.miListbox.getChildren().get(0).getChildren().size(); i++){
            Listheader miListheader = (Listheader)
this.miListbox.getChildren().get(0).getChildren().get(i);
            Textbox miTbox = (Textbox) miListheader.getChildren().get(0).getChildren().get(0);
            miTbox.setVisible(visible);
        }
    }

    /*****************************************************************
     * Establece si la cabecera de lista es visible o no.
     * @author gtomas
     * @since 04/01/2021
     * @param visible : true=Visible   false=Invisible.
     *
     *****************************************************************/
    public void setCabeceraVisible(boolean visible){
        for(int i=0; i<this.miListbox.getChildren().get(0).getChildren().size(); i++){
            Listheader miListheader = (Listheader)
this.miListbox.getChildren().get(0).getChildren().get(i);
            Vbox miVbox = (Vbox) miListheader.getChildren().get(0);
            miVbox.setVisible(visible);

        }
    }
```

```
/*****************************************************************
 * Establece el Ancho y Alto de la lista.
 * @author gtomas
 * @since 04/01/2021
 * @param ancho : Ancho de la lista.
 * @param alto  : Alto de la lista.
 *
 *****************************************************************/
public void setAnchoAltoLista(int ancho, int alto){
    this.miListbox.setWidth(ancho+"px");
    this.miListbox.setHeight(alto+"px");
}

/*****************************************************************
 * Establece el número de registros por página.
 * @author gtomas
 * @since 04/01/2021
 * @param numRegPagina : Número de registros por página.
 *
 *****************************************************************/
public void setPageSize(int numRegPagina){
    this.miListbox.setMold("paging");
    this.miListbox.setPageSize(numRegPagina);
}

/*****************************************************************
 * Establece el dato a mostrar en el Bandbox al seleccionar un
 * registro de la lista.
 * @author gtomas
 * @since 04/01/2021
 * @param campos : Cadena indicando los campos a mostrar
 * separados por espacios en blanco. En el caso de trabajar con Map,
 * se deberá colocar el identificador del elemento del mapa,
 * en caso de ser un Domain, se deberá colocar el nombre del
 * atributo de la clase con la primera letra en <b>Mayúscula</b>
 * El nombre de los campos debe
 * estar dentro del arreglo de campos especificados con el método
 * <b>setColumnas()</b>. En caso de indicar un texto que no es
 * columna, se insertará como parte del texto a mostrar.<br>
 * Si se desea mostrar más de un campo concatenado se deberá colocar
 * el símbolo <b>+</b>.<br>
 * Ej.<br>
 * String[] columnas = {"id", "nombre", "edad", "direccion", "sueldo"};<br>
 * miBandbox.setTextoOnSelect("id - nombre");<br>
 * miBandbox.setTextoOnSelect("id + nombre - sueldo");
 *****************************************************************/
public void setTextoOnSelect(String campos){
    this.pintarOnSelect = campos.split(" ");
}

/*---------------------------------------------------------------*/
private void filtrarLista(int indice){
    List<Object> listaFiltrada = new ArrayList<Object>();
    boolean filtrar = false;
    boolean agregar;
```

```
        for(int i=0; i<this.listaTextbox.size(); i++){
            Textbox tb = this.listaTextbox.get(i);
            if(!tb.getText().trim().equals("")){
                filtrar = true;
                break;
            }
        }
        try{
            this.selCamposOnRender = null;
            this.selValoresOnRender = null;
            if(filtrar == false){// Pintamos la lista original
                this.miListbox.setModel(new ListModelList<Object>(this.listaOriginal));
            }else{
                for(Object m : this.listaOriginal){
                    String dato;
                    agregar = true;
                    for(int i=0; i<this.listaTextbox.size(); i++){
                        Textbox tb = this.listaTextbox.get(i);
                        if(!tb.getText().trim().equals("")){
                            dato = obtenerValorColumna(columnas[i], tipoClase, m);

    if(!StringUtils.stripAccents(dato.toUpperCase()).contains(StringUtils.stripAccents(tb.getText().tri
m().toUpperCase()))){
                                agregar = false;
                                break;
                            }
                        }
                    }
                    if(agregar)listaFiltrada.add(m);
                }
                this.miListbox.setModel(new ListModelList<Object>(listaFiltrada));
                if(listaFiltrada == null || listaFiltrada.size()<=0){
                    this.miListbox.setEmptyMessage("No se encontraron resultados.");
                }
            }
        }catch(Exception e){
            logger.error(e.getMessage());
        }
    }

    /**************************************************************
     * Pinta en el Bandbox el registro seleccionado.<br><br>
     *
     * @author gtomas
     * @since 04/01/2021
     * @param bandbox : Bandbox a renderizar.
     * @throws NoSuchMethodException
     * @throws InvocationTargetException
     * @throws IllegalAccessException
     * @throws SecurityException
     * @throws IllegalArgumentException
     **************************************************************/
    @SuppressWarnings("unchecked")
    private void pintarOnSelect(Bandbox bandbox) throws Exception{
```

```
        String cad;
        String texto = "";
        Object miObj;
        for(int i=0; i<pintarOnSelect.length; i++){
            cad = pintarOnSelect[i].trim();
            if(!cad.equals("")){
                if(cad.equals("+"))
                    texto = texto.substring(0, texto.length() - 1);
                else{
                    try{
                        miObj = tipoClase.equals("java.util.HashMap") ||
tipoClase.equals("java.util.Map") ?
                                ((Map<String,Object>)filaSeleccionada).get(cad) :
                                    obtieneValorMetodo(filaSeleccionada, cad);
                    }catch(Exception e){
                        miObj = null;
                    }
                    if(miObj != null)
                        texto = texto + (tipoClase.equals("java.util.HashMap") ||
tipoClase.equals("java.util.Map") ?
                                ((Map<String,Object>)filaSeleccionada).get(cad).toString() :
                                    obtieneValorMetodo(filaSeleccionada, cad).toString()) + " ";
                    else texto = texto + cad + " ";
                }
            }
        }
        if(texto.trim().equals("- TODOS")) texto = "TODOS";
        bandbox.setText(texto.trim());
    }

    /**************************************************************
     * Seleccionado un registro al renderizar.<br><br>
     *
     * @author gtomas
     * @since 04/01/2021
     * @param bandbox : Bandbox a renderizar.
     **************************************************************/
    @SuppressWarnings("unchecked")
    private boolean selectOnRender(Object miData){
        String lsValor;
        boolean coincide = true;
        try{
            for(int i=0; i<this.selCamposOnRender.length; i++){
                lsValor = tipoClase.equals("java.util.HashMap") || tipoClase.equals("java.util.Map") ?
                        ((Map<String,Object>)miData).get(this.selCamposOnRender[i]).toString() :
                            obtieneValorMetodo(miData, this.selCamposOnRender[i]).toString();
                if(!lsValor.equals(this.selValoresOnRender[i].trim())){
                    coincide = false;
                    break;
                }
            }
        }catch(Exception e){
            logger.error(e.getMessage());
            return false;
        }
```

```
    if(coincide) return true;
    else return false;
}

/*****************************************************************
 * Renderiza el Bandbox.<br><br>
 *
 * @author gtomas
 * @since 04/01/2021
 * @param bandbox : Bandbox a renderizar.
 * @param lista : Lista de datos con los que se alimenta el
 * bandbox. La lista es de la forma <b>List&#60;Map&#60;String,Object>></b>
 * @param funcion : función que se ejecutará al seleccionar un
 * registro. Usar la siguiente plantilla:
 * <br><br>
 *
 * <b>new Callable(){public Object call() {return <font color="red">
 * onClickRow()</font>;}}</b><br><br>
 * Ejem.<br>
 * <b>this.miBB.pintar(new Callable(){public Object call()
 * {return <font color="red">onSelectBB()</font>;}});</b><br><br>
 * El texto en rojo es el nombre de la función que se ejecutará
 * y debe retornar un boolean.
 *****************************************************************/
@SuppressWarnings({ "unchecked", "rawtypes" })
public void pintar(final Bandbox, List lista, final Callable funcion){
    if(error) return;
    this.listaOriginal = lista;
    this.miListbox.setModel(new ListModelList<Object>(lista));
    this.miBB = bandbox;

    if(bandbox.getChildren().size() == 0){
        this.miBandpopup.appendChild(miListbox);
        bandbox.appendChild(this.miBandpopup);
    }
    limpiarFiltros();
    miListbox.addEventListener(Events.ON_SELECT, new EventListener(){
        @Override
        public void onEvent(Event event) throws Exception {
            filaSeleccionadaAnterior = filaSeleccionada;
            filaSeleccionada = miListbox.getSelectedItem().getValue();
            Integer indiceTmp = miListbox.getSelectedIndex();
            /*-- Pinta el registro seleccionado en la caja del Bandbox ---*/
            if(pintarOnSelect != null) pintarOnSelect(bandbox);

            bandbox.close();
            funcion.call();

            //funcion.call puede hacer rollback.
            //Si no se hizo rollback entonces se actualiza el indice
            if(indiceTmp != null && indiceTmp.equals(miListbox.getSelectedIndex())){
                indiceListboxAnterior = miListbox.getSelectedIndex();
            }
        }
    });
```

```
    this.miListbox.setItemRenderer(new ListitemRenderer() {
        @Override
        public void render(Listitem item, Object data, int index) throws Exception {
            tipoClase = data.getClass().getName();
            item.setValue(data);
            for(int i=0; i<columnas.length; i++){
                Listcell miListCell = new Listcell(obtenerValorColumna(columnas[i], tipoClase, data));
                item.appendChild(miListCell);

                /*--- Seleccionar un registro al renderizar ---*/
                if(selCamposOnRender != null && selValoresOnRender != null){
                    if(selectOnRender(data)){
                        indiceListboxAnterior = index;
                        filaSeleccionada = data;
                        filaSeleccionadaAnterior = data;
                        miListbox.setSelectedIndex(index);
                        pintarOnSelect(bandbox);
                    }
                }
            }
        }
    });
    correctivoSelectOnRender(bandbox);
}

/**************************************************************
 * Corrige el método SelectonRender. Este problema se presenta
 * cuando la lista del Bandbox tiene más de 50 registros ya que
 * no capturaba ni pintaba el registro seleccionado.<br><br>
 *
 * @author gtomas
 * @since 04/01/2021
 *
 **************************************************************/
private void correctivoSelectOnRender(Bandbox bandbox){
    for(int i=0; i<this.listaOriginal.size(); i++){
        Object data = this.listaOriginal.get(i);
        this.tipoClase = data.getClass().getName();
        /*--- Seleccionar un registro al renderizar ---*/
        if(selCamposOnRender != null && selValoresOnRender != null){
            if(selectOnRender(data)){
                this.filaSeleccionada = data;
                this.indiceListboxAnterior = i;
                try {
                    pintarOnSelect(bandbox);
                    break;
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    }
}
```

```
    @SuppressWarnings("unchecked")
    private String obtenerValorColumna(String col, String tipoClase, Object data){
        String valorCeleda = "";
        String[] lsCampos  = col.split(" ");
        for(String miCampo : lsCampos){
            valorCeleda = valorCeleda + (tipoClase.equals("java.util.HashMap") ||
tipoClase.equals("java.util.Map") ?
                    (((Map<String,Object>)data).get(miCampo) == null ? "":
((Map<String,Object>)data).get(miCampo).toString()) :
                    (obtieneValorMetodo(data, miCampo) == null ? "" :
                        obtieneValorMetodo(data, miCampo).toString()));
        }
        return valorCeleda;
    }

    /**************************************************************
     * Vuelve a renderizar el Bandbox con la nueva lista pasada
     * como parámetro. En caso se dese limpiar el bandbox se deberá
     * pasar <b>null</b> como parámetro.
     *
     * @param lista : Nueva lista para renderizar. Pasar <b>null</b>
     * si se desea limpiar el bandbox
     * @author gtomas
     * @since 04/01/2021
     *
     **************************************************************/
    @SuppressWarnings({ "rawtypes", "unchecked" })
    public void actualizarLista(List lista){
        this.listaOriginal = lista;
        if(lista == null) this.miListbox.setModel(new ListModelList());
        else this.miListbox.setModel(new ListModelList<Object>(lista));
        limpiarFiltros();
        if(this.miBB != null) this.miBB.setText("");
    }

    /**************************************************************
     * Limpia los filtros del Bandbox al refrescar la lista.
     * @author gtomas
     * @since 04/01/2021
     *
     **************************************************************/
    public void limpiarFiltros(){
        for(Textbox miTB : this.listaTextbox){
            miTB.setValue("");
        }
    }

    /**************************************************************
     * Establece los campos y valores criterio a evaluar para
     * seleccionar un registro al renderizar.
     * @author gtomas
     * @since 04/01/2021
     * @param campos : Cadena indicando los campos a evaluar
     * separados por espacios en blanco. El nombre de los campos debe
     * estar dentro del arreglo de campos especificados con el método
```

```
     * <b>setColumnas()</b>.<br><br>
     *
     * @param valores : Cadena indicando los valores de los campos a
     * evaluar separados por espacios en blanco. El nombre de los
     * campos debe estar dentro del arreglo de campos especificados
     * con el método <b>setColumnas()</b>.<br><br>
     *
     * Ej.<br>
     * String[] columnas = {"id", "nombre", "edad", "direccion", "sueldo"};<br>
     * miBandbox.setSelectOnRender("id", "4");
     ************************************************************/
    public void setSelectOnRender(String campos, String valores){
        String[] lsCampos  = campos.split(" ");
        String[] lsValores = valores.split(" ");
        String cadena = "";

        /*--- Limpiamos de espacios en blanco ---*/
        for(String cad : lsCampos){
            if(!cad.trim().equals("")) cadena = cadena + cad + " ";
        }
        cadena = cadena.trim();
        this.selCamposOnRender = cadena.split(" ");

        cadena = "";
        for(String cad : lsValores){
            if(!cad.trim().equals("")) cadena = cadena + cad + " ";
        }
        cadena = cadena.trim();
        this.selValoresOnRender = cadena.split(" ");

        if(this.selCamposOnRender.length != this.selValoresOnRender.length){
            error = true;
            Messagebox.show("setSelectOnRender()\n\nERROR: El número de campos no coincide con
el número de valores. Verificar !!!");
        }
    }

    /************************************************************
     * Establece los campos y valores criterio a evaluar para
     * seleccionar un registro al renderizar.
     * @author gtomas
     * @since 04/01/2021
     * @param campos : Cadena indicando los campos a evaluar
     * separados por espacios en blanco. El nombre de los campos debe
     * estar dentro del arreglo de campos especificados con el método
     * <b>setColumnas()</b>.<br><br>
     *
     * @param valores : Cadena indicando los valores de los campos a
     * evaluar separados por espacios en blanco. El nombre de los
     * campos debe estar dentro del arreglo de campos especificados
     * con el método <b>setColumnas()</b>.<br><br>
     *
     * @separador : Caracteres usado como separador de campos y
     * valores.
     * Ej.<br>
```

```
     * String[] columnas = {"id", "nombre", "edad", "direccion", "sueldo"};<br>
     * miBandbox.setSelectOnRender("id", "4");
     ****************************************************************/
    public void setSelectOnRender(String campos, String valores, String separado){
        String[] lsCampos  = campos.split(separado);
        String[] lsValores = valores.split(separado);
        String cadena = "";

        /*--- Limpiamos de espacios en blanco ---*/
        for(String cad : lsCampos){
            if(!cad.trim().equals("")) cadena = cadena + cad + " ";
        }
        cadena = cadena.trim();
        this.selCamposOnRender = cadena.split(separado);

        cadena = "";
        for(String cad : lsValores){
            if(!cad.trim().equals("")) cadena = cadena + cad + " ";
        }
        cadena = cadena.trim();
        this.selValoresOnRender = cadena.split(separado);

        if(this.selCamposOnRender.length != this.selValoresOnRender.length){
            error = true;
            Messagebox.show("setSelectOnRender()\n\nERROR: El número de campos no coincide con
el número de valores. Verificar !!!");
        }
    }

    /****************************************************************
     * Devuelve un mapa <b>Map&#60;String,Object></b> con el registro
     * seleccionado.<br><br>
     *
     * @author gtomas
     * @since 04/01/2021
     *
     ****************************************************************/
    public Object getFilaSeleccionada() {
        return filaSeleccionada;
    }

    private Object obtieneValorMetodo(Object miObjeto, String nombreAtributo){
        try {
            String[] metodos = nombreAtributo.split("\\.");
            nombreAtributo = "get" + Util.UpperCaseFirstChar(metodos != null && metodos.length > 1
?
                    metodos[0] : nombreAtributo);
            Object resultadoId = miObjeto.getClass().getMethod(nombreAtributo).invoke(miObjeto);
            if(metodos.length > 1){
                nombreAtributo = "get" + Util.UpperCaseFirstChar(metodos[1]);
                if(resultadoId == null)
                    return null;
                Object                                  resultadoFinal                                  =
resultadoId.getClass().getMethod(nombreAtributo).invoke(resultadoId);
                return resultadoFinal == null ? null : resultadoFinal.toString();
```

```
        }else
            return resultadoId == null ? null : resultadoId.toString();
    } catch (Exception e) {
        logger.info(e.getMessage());
    }
    return null;
}

public String getTipoClase() {
    return tipoClase;
}

public void setTipoClase(String tipoClase) {
    this.tipoClase = tipoClase;
}

/**************************************************************
 * Devuelve el registro seleccionado anterior.<br><br>
 *
 * @author gtomas
 * @since 04/01/2021
 *
 **************************************************************/
public Object getFilaSeleccionadaAnterior() {
    return this.filaSeleccionadaAnterior;
}

/**************************************************************
 * Deshace la selección (Evento onSelect).<br><br>
 *
 * @author gtomas
 * @since 04/01/2021
 *
 **************************************************************/
public void rollbackOnSelect(){
    this.filaSeleccionada = this.filaSeleccionadaAnterior;
    this.miListbox.setSelectedIndex(this.indiceListboxAnterior);
    try {
        pintarOnSelect(this.miBB);
    } catch (Exception e) {
        e.printStackTrace();
    }
}

/**************************************************************
 * Devuelve el índice del Registro actualmente seleccionado.
 * Este dato puede usarse para regresar a esa posición, luego
 * de haber seleccionado otro registro mediante el método
 * <b>setSelecItem(Integer indice)</b>.
 *
 * @return Integer
 * @author gtomas
 * @since 04/01/2021
 *
 **************************************************************/
```

```
public Integer getIndexSeleccionado(){
    if(this.miListbox != null){
        return this.indiceListboxAnterior;
    }else{
        return -1;
    }
}

/*****************************************************************
 * Selecciona el ítem correspondiente al índice pasado como
 * parámetro.
 *
 * @param indice : Posición del ítem que se desea seleccionar.
 * Este dato puede capturarse con el método
 * <b>getIndexSeleccionado()</b>.
 * @author gtomas
 * @since 04/01/2021
 *
 *****************************************************************/
public void setSelecItem(Integer indice){
    if(this.miListbox != null){
        this.miListbox.setSelectedIndex(indice);
        if(indice.intValue()>=0){
            this.filaSeleccionada = this.miListbox.getSelectedItem().getValue();
            try {
                pintarOnSelect(this.miBB);
            } catch (Exception e) {
                e.printStackTrace();
            }
        }else{
            this.miBB.setText("");
        }
    }
}

/*****************************************************************
 * Especifica las columnas a visualizar. Sobrecarga el método
 * setcolumnas detectando automáticamente el ancho del listbox
 * que es la suma de los anchos de cada columna más 80 pixeles.
 * El alto del listbox se setea en 250px por defecto. En caso se
 * desee modificar estos valores, usar el método
 * <b>setAnchoAltoLista()</b>.
 *
 * @author gtomas
 * @since 04/01/2021
 * @param columna : Arreglo de tipo String <b>String[]</b> para
 * indicar las columnas a visualizar. En caso se requiera que una
 * columna este compuesta por más de un campo se deberá colocar
 * dentro del arreglo dejando un espacio en blanco. <b>Ejemplo:
 * {"campo1 campo2 campo3", "columna2"}</b>
 * @param etiqueta : Arreglo de tipo String <b>String[]</b> para
 * indicar las etiquetas (rótulos) de las columnas.
 * @param ancho : Arreglo de tipo int <b>int[]</b> para
 * indicar en ancho de las columnas.
 *****************************************************************/
```

```
    public void setColumnas(String[] columna, String[] etiqueta, int[]ancho){
        int anchoLista = 0;
        if(ancho != null && ancho.length>0){
            for(int i=0; i<ancho.length; i++){
                anchoLista = anchoLista + ancho[i];
            }
            anchoLista = anchoLista + 80;
            setColumnas(columna, etiqueta, ancho, anchoLista);
            this.miListbox.setHeight("250px");
            this.miListbox.setMold("paging");
            this.miListbox.setPageSize(50);
        }
    }
}
```