Módulo Partes tecnológicos para el Sistema de Gestión Universitaria.

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor:

José Carlos Rodríguez Texidor

Tutor(es):

Ing. Yamila Mateu Romero

Ing. Dianly Santiler Álvarez

Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: Módulo Partes Tecnológicos para el Sistema de Gestión Universitaria y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente tesis a los __ días del mes de Junio del año 2019.

José Carlos Rodríguez Texidor

_____

Firma del autor

Ing. Yamila Mateu Romero

_____

Firma de la tutora

Ing. Dianly Santiler Álvarez

_____

Firma del tutor

Agradecimientos

Gracias, a todas aquellas personas que contribuyeron de alguna forma a mi formación y crecimiento personal durante estos 5 años, sobre todo a mi familia, amigos y profesores, quienes siempre me apoyaron y nunca me dejaron caer. Gracias por su compañía, por perdonar mis errores y ayudarme a corregirlos, por las segundas, terceras y hasta cuartas oportunidades.

Dedicatoria

A mi familia, amigos y profesores, por su valiosa ayuda y apoyo incondicional en los momentos más difíciles, gracias por ser un pedacito de cielo en este mundo y hacer que la vida haya valido la pena.

Resumen

La Universidad de las Ciencias Informáticas es un centro de altos estudios que hace uso de las Tecnologías de la Información y las Comunicaciones en el proceso de informatización de la sociedad y la propia institución, siendo el Sistema de Gestión Universitaria una de las herramientas de apoyo a varios de sus procesos. Uno de ellos es la creación de informes para conocer el estado de la tecnología en la universidad, realizado mediante hojas de cálculo, sin verificación de los locales o la cantidad de dispositivos en ellos.

La presente investigación tiene como objetivo el desarrollo de un módulo que permita agilizar y centralizar la creación de partes tecnológicos e informes consolidados. El componente desarrollado permite la selección de los dispositivos que deben aparecer en los partes, así como la definición de las características asociadas a cada uno de ellos. Además, permite verificar la cantidad de medios tecnológicos en los locales que tiene estructura. Admite publicación de convocatorias para generar estos partes y la especificación de las personas responsables de llenarlos. El módulo acepta la consolidación de informes teniendo en cuenta el área de responsabilidad, el tipo de local, los dispositivos y sus características. Para su desarrollo se utilizó la metodología *Scrum* y el entorno tecnológico definido por el Departamento de Desarrollo de la Dirección de Informatización.

Palabras clave: Balance de la tecnología, dispositivos, informes consolidados, partes tecnológicos.

Índice

Índice de tablas

Índice de Figuras

Introducción

La nueva política económica de Cuba plantea el avance gradual en el proceso de informatización de la sociedad con el objetivo de aprovechar las ventajas de las tecnologías de la información y las comunicaciones (TIC), como herramientas para el desarrollo del conocimiento, la economía y la actividad política e ideológica. Como parte de esta estrategia, en el artículo 115 se propone ampliar la vinculación de los sectores empresarial y académico con el propósito de incentivar la aplicación de los resultados científicos y tecnológicos en la producción y los servicios.

La Universidad de Ciencias Informáticas (UCI) es una de las instituciones que contribuye a la migración del país al uso de tecnologías de software libre y código abierto, con un alto impacto en la informatización de la sociedad, a través de los servicios informáticos que ofrece. Para garantizar la realización de todos sus procesos sustantivos, mantiene una infraestructura dotada de amplias y modernas instalaciones al servicio de la comunidad universitaria. Dicha infraestructura está conformada por más de 150 edificaciones distribuidas en 268 hectáreas. De ellas 6 edificios que acogen 150 aulas, 30 salones de conferencia y más de 200 laboratorios destinados a las actividades docente-productivas de las diferentes facultades. Esto significa que existe gran cantidad y diversidad de equipos tecnológicos distribuidos en todas sus áreas, por lo que es necesario mantener un inventario de estos activos.

El inventario[1] es una parte fundamental de cualquier negocio. Saber cuáles son los activos, en qué condiciones se encuentran o dónde están ubicados, es fundamental para mantener el control. El uso de software especializado en la gestión de inventarios facilita además, otras operaciones, como la creación de archivos históricos y la generación de informes que consolidan la información a diferentes niveles.

Para gestionar la información relacionada a los activos de la Universidad se utiliza el módulo Activos Fijos del sistema contable-financiero *Assets NS*. Este es un software propietario basado en la filosofía cliente-servidor que se compone de la aplicación escritorio *Assets NS* y una base de datos distribuida. Este módulo permite controlar los medios básicos por centros de costos y entre sus funcionalidades se encuentran las siguientes: “definir la ubicación física de los mismos, controlar los procesos de compras, alquiler, altas, bajas, préstamos, traspasos hacia otras áreas dentro y fuera de la entidad, realizar reparaciones, controlar ociosos, realizar ajustes, controlar la depreciación acumulada de cada activo” [1]

[1] Inventario: Existencias de materias primas, producción en proceso y bienes terminados que mantiene una organización para satisfacer sus necesidades operativas.

entre otras. Sin embargo, la actualización periódica del estado de cada activo no está contemplada entre sus funciones.

Para conocer el estado técnico de los equipos tecnológicos cada cierto tiempo, el Director de Gestión Tecnológica de la Universidad debe solicitar a cada responsable de área un informe. En este debe quedar reflejada la cantidad de medios por local, teniendo en consideración el tipo de dispositivo (computadoras, monitores, baterías, impresoras, *laptops*, entre otros), su estado técnico (rotos, buen estado, desuso, entre otros), y en algunos casos sus características (si un monitor es de tecnología LCD o CRT, si una PC es de tecnología iCore o no). Una vez que los responsables de área crean los informes, el Director de Gestión Tecnológica debe consolidar la información según los tipos de local (laboratorios docentes, laboratorios de producción, aulas, oficinas, entre otros) para conocer como está distribuida la tecnología en la universidad.

Actualmente todo el proceso es realizado de forma manual, pues no existe una herramienta informática que centralice todos los pasos necesarios para la creación de un informe tecnológico consolidado. Como primera actividad el Director de Gestión Tecnológica define una plantilla con los datos que debe reflejar el parte, usando generalmente una hoja de cálculo. Luego, hace llegar este modelo a cada responsables de área notifica la fecha límite para la entrega del parte.

Para la elaboración de estos partes se usan herramientas ofimáticas y por tanto no existe ningún tipo de sincronización automática con otros sistemas para validar que la cantidad de medios de cada tipo sea realmente la registrada en el sistema económico, ni existe verificación sobre los locales que tiene cada estructura. Ello trae consigo que los informes estén expuestos a errores humanos que pueden afectar los datos y generalizarse a los diferentes niveles de autoridad.

Para crear un informe consolidado el Director de Gestión Tecnológica comprueba manualmente que en cada parte la cantidad de dispositivos y los locales coinciden con los datos del sistema económico. Una vez realizada esta operación, consolida la información en una hoja de cálculo según los tipos de local, procedimiento que no es automático y debe ser realizado manualmente analizando cada uno de los partes. Para los especialistas de esta dirección realizar informes donde se conozca el estado de la tecnología en los locales conlleva cada vez más esfuerzo. En algunas ocasiones es necesario realizar esta labor en horas extras para poder cumplir con las solicitudes que el Ministerio de Educación Superior hace a la universidad.

Con el decursar del tiempo se ha generado un cúmulo de información que ha sido archivado en diferentes formatos y con diferentes criterios sin estar centralizada dificultando la búsqueda y análisis de la información.

Del análisis anterior se desprende el siguiente **problema de investigación**: ¿Cómo mejorar la eficiencia del proceso de gestión de los partes tecnológicos y reportes consolidados en la Universidad de las Ciencias Informáticas? Que tiene como **objeto de estudio**: Sistemas de gestión técnica de activos y sistemas de generación de informes estadísticos y cuyo **campo de acción** es el proceso de generación de informes estadísticos sobre el estado de activos tecnológicos en el Sistema de Gestión Universitaria (SGU) de la Universidad de las Ciencias Informáticas.

Para dar solución al problema de investigación se plantea como **objetivo general:** Desarrollar un módulo en el Sistema de Gestión Universitaria de la Universidad de Ciencias Informáticas que automatice el proceso de generación de partes tecnológicos e informes consolidados, de manera que contribuya a la accesibilidad, estandarización y persistencia de la información. De este objetivo general se derivan los siguientes **objetivos específicos:**

- Realizar un análisis crítico de las herramientas informáticas de gestión de activos.
- Describir los métodos, técnicas y herramientas de desarrollo adecuadas para la implementación del módulo para el Sistema de Gestión Universitaria que automatice el proceso de gestión de informes tecnológicos consolidados.
- Realizar el análisis y diseño para la implementación del módulo para el Sistema de Gestión Universitaria que automatice el proceso de gestión de informes tecnológicos.
- Implementar el módulo para automatizar el proceso de generación de informes sobre el estado de la tecnología en la Universidad de Ciencias Informáticas.

  Realizar las pruebas para verificar y validar la solución propuesta.

Para sustentar la investigación es planteada la siguiente **hipótesis:** La implementación de un módulo informático integrado al sistema de gestión universitaria, mejora la eficiencia en los procesos relacionados con la gestión de partes tecnológicos y reportes consolidados en la Universidad de las Ciencias Informáticas.

Identificándose como **variable independiente** el módulo, Partes Tecnológicos del Sistema de Gestión Universitaria y como **variable dependiente** la eficiencia en el proceso de generación de partes tecnológicos e informes consolidados.

Tabla 1: Operacionalización de las variables (Elaboración propia).

Variable conceptual	**Dimensiones**	**Indicadores**	**Unidad de medida**
Módulo Partes Tecnológicos.	Adecuación funcional	Completitud funcional	95-100 (%)
	Usabilidad	Reconocibilidad	85-95 (%)
		Estética de interfaz de usuario	Menos de 85 (%)
Eficiencia en el proceso de generación de partes tecnológicos e informes consolidados.	Tiempo medio necesario para crear un parte tecnológico.	Alta	1-5 (min)
		Media	5-15 (min)
		Baja	Más de 15 (min)
	Tiempo medio necesario para generar un informe consolidado.	Alta	0-5(seg)
		Media	5-7 (seg)
		Baja	Más de 7 (seg)
	Fiabilidad e integridad de los datos registrados.	Alta	95-100 (%)
		Media	85-95 (%)
		Baja	Menos de 85 (%)

Para dar cumplimiento a los objetivos específicos se plantean las siguientes **tareas de investigación:**

- Realización de un estudio sobre el uso de las tecnologías en los procesos de gestión de activos tecnológicos.
- Caracterización de los sistemas de gestión de activos tecnológicos.
- Definición de los conceptos relacionados con el marco teórico de la investigación y las tecnologías necesarias para el desarrollo de la solución.
- Descripción de la metodología de desarrollo, tecnologías y estándares usados para desarrollar el módulo de gestión de partes tecnológicos.
- Selección de las herramientas que se necesitan para implementar la propuesta de solución.
- Definición de los requisitos funcionales y no funcionales de la propuesta de solución.
- Implementación de la propuesta de solución.
- Desarrollo y documentación de las pruebas carga, estrés y de aceptación del software diseñadas para medir la calidad de la solución.

Para dar solución a la presente investigación serán utilizados **métodos de investigación** teóricos que se basan en el trabajo con conceptos, categorías, leyes, teorías, entre otros y empíricos que se obtienen de la interacción directa entre el sujeto y el objeto de la investigación.

Entre los métodos **teóricos** de investigación a ser usados se encuentran:

- **Histórico-Lógico**: Posibilita realizar un estudio de los principales conceptos relacionados con inventarios tecnológicos desde su surgimiento hasta la actualidad, así como de los diferentes sistemas de gestión de medios tecnológicos existentes hasta la fecha. La utilización de este método permite que el estudio no se limite a una simple descripción de los hechos, sino que facilite el descubrimiento de la lógica objetiva del desarrollo histórico.
- **Analítico-Sintético**: el análisis permite el estudio de cada uno de los elementos necesarios en la gestión de activos para las diferentes aristas en las que debe presentarse la información para contribuir a la generación de partes e informes. Por su parte la síntesis permite descubrir las relaciones existentes entre un elemento y otro, así como la interacción dialéctica que se establece entre ellos.
- **Modelación**: Facilita la comprensión de la solución propuesta mediante representación de sus características, procesos, componentes y la relación entre ellos a través de diagramas y esquemas.
- **Hipotético-deductivo**: El método hipotético-deductivo permite llegar a nuevas conclusiones y predicciones empíricas o sea posibilita pronosticar y verificar nuevas hipótesis, así como inferir otras a partir del conocimiento que se posee. Su utilización permite la realización de propuestas o recomendaciones a partir de los resultados parciales.

Y como métodos **empíricos** de investigación se utilizarán:

- **Entrevista:** Se realiza a través de encuentros con el cliente, usuarios finales y con especialistas para conocer las características del negocio, la necesidad del desarrollo de la propuesta de solución, y definir sus requisitos funcionales y no funcionales.
- **Revisión documental:** se pone de manifiesto en las revisiones a la bibliografía existente y a la hora de consultar la información en sitios de interés nacional e internacional, para apoyar la realización de las tareas definidas en la investigación.

El contenido de la presente investigación se estructura en 3 capítulos distribuidos de la siguiente manera:

Capítulo 1: Fundamentación Teórica.

En este capítulo se realiza un estudio de los referentes teórico-metodológicos asociados a la investigación. Se incluye un estudio del estado del arte abordando temas como las últimas tendencias y técnicas de software para la gestión de activos. Se analizan los diferentes sistemas que manejan la gestión de inventarios tecnológicos y generación de reportes estadísticos a nivel internacional y nacional, enfocándose en las propias necesidades del negocio. Se describen la metodología, herramientas y tecnologías a utilizar en la propuesta de solución.

Capítulo 2: Análisis y diseño del sistema.

En este capítulo se describe el sistema que se propone para dar solución a la problemática. Se argumentan los procesos que dan origen a esta situación y los que serán automatizados. Se recogen las características básicas del sistema expresadas en requisitos no funcionales y funcionalidades que se describen en las historias de usuario. Además, se incluye la definición del modelo del análisis, el modelo de datos, así como los patrones de diseño utilizados y se describe la organización del módulo en un diagrama de despliegue.

Capítulo 3: Implementación y validación del sistema.

Se especifican los estándares de codificación a utilizar. Se realizan y describen las pruebas definidas para el sistema garantizando su correcto funcionamiento y culmina con la validación de la hipótesis de investigación.

Posibles resultados:

Con la realización de la presente investigación, se pretende obtener un módulo para el Sistema de Gestión Universitaria de la Universidad de las Ciencias Informáticas que permita realizar el proceso de gestión de los partes tecnológicos e informes consolidados de una forma más rápida, sencilla e intuitiva.

Capítulo 1: Fundamentación teórica

1.1 Introducción

Con el objetivo de lograr una mayor comprensión del alcance de la investigación y esclarecer su objeto de estudio, se exponen en el presente capítulo, los conceptos asociados al dominio de la investigación y se realiza un análisis del estado del arte de sistemas homólogos. Se presenta la metodología y el entorno de desarrollo a utilizar para dar solución al problema planteado.

1.2 Marco conceptual

Con el propósito de una mejor comprensión de los temas que serán tratados en este capítulo, directamente vinculados con el objeto de estudio de la investigación, se describen a continuación un grupo de conceptos relacionados con el dominio del problema.

Parte tecnológico: Informe estadístico que recoge la información técnica de los activos tecnológicos en un área de responsabilidad. Agrupa los datos por locales y en algunos casos tiene en cuenta además, del tipo de medio tecnológico, las características y estado técnico del dispositivo.

Informe consolidado: Se crea a partir de la centralización de los datos de cada uno de los partes, dependiendo de la necesidad de información estos son agrupados según diversos criterios, por ejemplo, según las áreas de responsabilidad, las características del dispositivo o su estado técnico.

Área de responsabilidad: Se refiere según términos financieros a cada uno de los centros de costo de la universidad donde existan dispositivos tecnológicos.

Director de Gestión Tecnológica: Persona que ocupa el cargo administrativo principal en la Dirección de Gestión Tecnológica de la universidad, es un rol único que solo desempeña una persona en un momento determinado.

Responsable de área: Es un rol definido de manera autónoma por el Director de Gestión Tecnológica, y no está relacionado a ningún cargo, autoridad, o estructura de la institución. Los requisitos para desempeñar esta función son determinados por la autoridad competente, en este caso, el Director de Gestión Tecnológica.

1.3 Análisis de los sistemas homólogos

Como convenio en esta investigación, se tomará al término ‘sistema homólogo’ como la relación análoga o de semejanza entre varios sistemas informáticos que tienen funcionalidades parecidas a pesar de haber sido desarrollados usando tecnología distinta o una arquitectura diferente.

A continuación se realiza un análisis de algunos de los sistemas disponibles cuyas características y funcionalidades están relacionadas con las necesidades del negocio. En la investigación se tendrán en cuenta soluciones de alcance global, nacional e institucional.

1.3.1 ManageEngine AssetExplorer

ManageEngine AssetExplorer es un software ITAM[2] basado en la web que permite monitorizar y administrar los activos en una red desde la fase de planificación hasta la fase de eliminación. Proporciona una serie de formas para garantizar el descubrimiento de todos los activos en una red. Se pueden administrar activos de software y hardware, garantizar el cumplimiento de las licencias de software y rastrear pedidos de compra y contratos. Es fácil de instalar y funciona de inmediato.

Este software permite generar informes predefinidos / personalizados / detallados de consulta, brindando la posibilidad de programar estos informes y entregarlos en la bandeja de entrada. Monitoriza el ciclo de vida completo de los activos desde la adquisición hasta la disposición. Permite el descubrimiento automático, la administración de todo el inventario de hardware y software implementado en la red y el desglose de activos para ver la información crítica (historial, información, entre otras) [2].

Es privativo, tiene asociada una licencia libre que provee las funcionalidades básicas, una de prueba por 30 días y una profesional que expira con el tiempo.

A pesar de identificar los dispositivos en la red este software contiene la funcionalidad no deseada de validación de licencias de software y no tiene una manera de determinar el estado de los dispositivos que no están conectados a la red, por lo que su dominio de acción es bastante limitado para los requisitos del negocio. Además, es un software privativo sujeto a licencias, la versión de licencia libre y permanente limita a 25 la cantidad de dispositivos a administrar, lo cual no satisface las necesidades de la universidad que cuenta con gran cantidad de activos tecnológicos.

[2] ITAM (*Information Technology Asset Management,* gestión de activos tecnológicos) es el conjunto de prácticas comerciales que se unen a funciones financieras, contractuales y de inventario para respaldar la gestión del ciclo de vida y la toma de decisiones estratégicas del entorno IT.

1.3.2 SysAid IT Asset Management

SysAid> Help Desk Software es un software de IT *Help Desk*[3] que proporciona las funciones esenciales para administrar las tareas de IT (*Information Technology*, Tecnologías de la Información) en un *Service Desk*[4] entre ellas: La administración del inventario de hardware y software, integración *Help Desk* (Servicio de Asistencia Técnica), administración de parches, monitoreo, MDM[5], mensajes *broadcast* (multidifusión), proveedor de catálogos, control remoto, entre otras.

Facilita el descubrimiento de los activos en una red mostrando la imagen completa de sus componentes de hardware y productos de software. Puede ver el software instalado en cada activo y los detalles del hardware dentro de ellos, por ejemplo, CPU (*Central Process Unit*, Unidad Central de Procesamiento), RAM (*Random Access Memory*, Memoria de Acceso aleatorio) y HDD (*Hard Disk Drive*, Disco Duro).

Cualquier cambio de hardware o software realizado en el inventario de los elementos de red es automáticamente añadido a los registros de actividad de activos. Esto crea un registro que permite mantener listado de los cambios y ayuda al personal de soporte en el proceso de administración de incidentes. Compila automáticamente una lista de todos los modelos de hardware encontrados en la red. Permitiendo saber cuántos dispositivos de cada tipo se tienen y permitiendo añadir proveedor, garantía, mantenimiento u otra información relevante de cada modelo de catálogo [3].

Esta es una aplicación de software libre, puede ser instalado en Linux y Windows, tiene soporte en una amplia variedad de idiomas entre los que se incluye el español, y cumple parcialmente con los requerimientos y funcionalidades que se desean obtener. A pesar de que permite conocer de manera automática la cantidad de medios de cada tipo no es posible agruparlos por local, y solo registra aquellos

[3] Un servicio soporte técnico o servicio de asistencia técnica (*help desk*) es un recurso destinado a proporcionar al cliente o usuario final información y asistencia relacionada con los productos y servicios de una empresa o institución.

[4] Un centro de servicio al cliente (*Service desk*) es un centro de comunicaciones que proporciona un SPOC (*single point of contact*, único punto de contacto) entre una empresa y sus clientes, empleados y socios comerciales. El propósito de una mesa de servicio es asegurar que los usuarios reciban la ayuda adecuada de manera oportuna.

[5] En los negocios, MDM (*Master Data Management*, Gestión de Datos Maestros) es un método utilizado para definir y administrar los datos críticos de una organización para proporcionar, con integración de datos, un único punto de referencia. Los datos que se controlan pueden incluir datos de referencia: el conjunto de valores permitidos y los datos analíticos que respaldan la toma de decisiones.

que se encuentran conectados a la red por lo que no podría almacenar información de aquellos que están desconectados o rotos.

1.3.3 Asset Track

Asset Track es una solución de seguimiento de activos innovadora, robusta y simple, diseñada y desarrollada por *Jolly Technologies Inc*., una compañía de software que ayuda a varias empresas en la administración de todos sus activos fijos e IT. El software de seguimiento de activos es ideal tanto para pequeñas empresas que tienen necesidades limitadas de gestión de activos como para grandes empresas que se ocupan de múltiples sitios, productos y servicios, y requisitos complejos.

La solución permite a los usuarios realizar una auditoría de todos sus activos fijos, lo que les permite monitorear su condición y ubicación. *Asset Track*, tiene la capacidad de proporcionar los valores de depreciación de los activos fijos y de IT. Permite a los usuarios rastrear a los empleados y al personal que tomaron prestados sus activos y equipos, y notifica automáticamente a los prestatarios cuando necesitan devolver los activos.

Asset Track se asegura de que las empresas puedan mantener registros detallados de los activos lo que permite validar los datos capturados y la gestión de la seguridad a nivel de funcionalidades y recursos. Posibilita la consulta y procesamiento de los datos registrados a través de una herramienta Generación Dinámica de Reportes (GDR). Estos datos son resultado del análisis de las soluciones existentes, servicios de mantenimiento e información relacionada con la garantía de todos los activos fijos críticos. Estos registros contienen información detallada, como el tipo de activo, el número de serie, el número de pieza, el proveedor y el costo.

Es un software foráneo que no se comunica de forma natural con el sistema de gestión universitaria de manera que no reconoce los locales, áreas o tipos de local que están registrados. Los reportes que genera no consolidan la información del estado de los dispositivos por local sino que están orientados a dar información específica de un dispositivo.

1.3.4 Sistema Integrado de Gestión Estadística (SIGE) 3.0

El Sistema Integrado de Gestión Estadística (SIGE) es una aplicación web que tiene como objetivo principal la automatización de los procesos de gestión estadística de una institución. SIGE es una innovación que está basada en tecnologías libres: ExtJS 3.4 y el marco de trabajo UCI para la capa de presentación; Symfony 1.1 en la capa de negocio y PostgreSQL 9.3 como gestor de bases de datos.

Posee una arquitectura modular que permiten la captura y recolección de datos, presentación de la información e inteligencia organizacional. Dicha arquitectura está compuesta de 7 módulos: Gestión de la Configuración, Diseñador de Plantillas, Entrada de Datos, Herramientas, Administración, Georeferencias y Seguridad. Además, tiene integrado el Generador Dinámico de Reportes (GDR) que le brinda un conjunto de herramientas para el diseño, generación y visualización de reportes [4].

El sistema de gestión estadística SIGE está orientado a la recepción de datos a través de formularios, por lo que no responde a la sincronización automática que se desea con el Sistema de Gestión Universitaria quien toma los datos de *Assets*, módulo del sistema contable financiero de la universidad, donde los datos de medios tecnológicos no son estáticos y un medio puede cambiar de ubicación, darse de baja o repararse.

1.3.5 Sistema de Gestión Administrativa: Módulo Activos

El Sistema de Gestión Administrativa (Kainos) es un software diseñado para mejorar la planeación, organización y dirección de los procesos en la Universidad de la Ciencias Informáticas. Es una aplicación web desarrollada usando el marco de trabajo GUUD, el cual es una combinación de la biblioteca jQuery y el marco de trabajo CodeIgniter. El equipo de desarrollo utiliza *Scrum* como metodología para guiar desarrollo y los lenguajes HTML (*HyperText Markup Language*, Lenguaje de Marcado de Hipertexto), CSS (*Cascading Style Sheets*, Hojas de Estilo en Cascada), JavaScript, PHP (*Hypertext Preprocessor*, Procesador de Hipertexto) y UML (*Unified Modeling Language*, Lenguaje Unificado de Modelado) para la construcción del software. La aplicación se ejecuta de manera distribuida en un servidor web Nginx, un servidor de base de datos PostgreSQL y dispositivos clientes con navegadores web.

Kainos facilita el trabajo con la información asociada a los procesos dentro de la institución, así como la consulta rápida y segura de la información asociada a los mismos. Garantiza la salvaguarda de la información generada por los procesos que gestiona. Facilita la revisión de los hechos y acciones registradas a partir de informes estadísticos y de listados realizados.

Este sistema se compone de varios módulos y submódulos entre los cuales se encuentra Activos. El mismo pertenece a la agrupación funcional Sistema y permite el control de los activos auxiliándose de los datos que el Sistema Contable Financiero de la universidad proporciona. Su objetivo principal es realizar operaciones locales y generar informes con lo cual disminuye en número de operaciones innecesarias realizadas al sistema *Assets NS*.

Entre las operaciones que pueden realizarse con este módulo se encuentra la generación de reportes que consolidan la información según diferentes criterios. Por ejemplo, pueden generarse reportes según la clasificación (edificios, máquinas y equipos productivos, aparatos y equipos técnicos especiales, entre otros), subclasificación (edificio, equipo médico, impresora, computadora, entre otros), centro de costo (Vicerrectoría de Producción, Facultad 1, Centro de Informatización de Entidades, entre otros) y el área de responsabilidad al que pertenece el activo.

Sin embargo estos reportes son creados automáticamente con la información que se obtiene de *Assets NS*, sin que pueda incluirse información adicional de los activos, como es requerido por los informes tecnológicos. El flujo de información requerido para crear un informe consolidado requiere:

1. La asignación a cada dispositivo de atributos y características.
2. La creación partes donde se registre la cantidad de medios en cada local de un área teniendo en cuenta dichas características y atributos.

Sin embargo, dicho módulo carece de las funcionalidades necesarias para asignar características a los dispositivos y crear partes tecnológicos.

Por otra parte Kainos tiene un conjunto de características que hacen factible la creación de un módulo para dar solución a la situación problemática. Estas características son las siguientes:

- Constituye una herramienta de apoyo personalizada para el perfeccionamiento de la gestión de los procesos en la Universidad de Ciencias Informáticas.
- Su uso permite el desarrollo coherente de una estrategia organizacional a todos los niveles de decisión presentes en el proceso de gestión universitaria.
- Logra la sincronización con el sistema contable financiero de la universidad y cuenta con un equipo de desarrollo en la propia institución que tiene conocimientos de la arquitectura del sistema y experiencia en el marco de trabajo.
- Es un sistema modular en el que sus componentes pueden comunicarse entre sí y compartir información.

1.3.6 Resultado del análisis de las soluciones existentes

El análisis realizado sobre las aplicaciones para gestionar los activos IT y generar reportes estadísticos, permitió realizar un resumen (ver tabla 2) teniendo en cuenta los siguientes parámetros:

Adquisición: se refiere al estado jurídico de la aplicación en cuanto a su uso, modificación y distribución, esta puede ser libre, en aquellas que no necesitan un pago para ser utilizadas, o pago de licencia, en aquellas que sí lo requieren.

Entorno: se refiere a si la herramienta es una aplicación web o de escritorio.

Funcionalidad: nivel con que la aplicación satisface las necesidades del negocio.

Tabla 2: Resumen de las características de los sistemas homólogos (Elaboración propia).

Herramienta	**Adquisición**	**Entorno**	**Funcionalidad**
ManageEngine AssetExplorer	Pago de licencia	Escritorio	Parcial
SysAid IT Asset Management	Libre	Escritorio	Parcial
Asset Track	Pago de licencia	Escritorio	Parcial
Sistema Integrado de Gestión Estadística	Libre	Web	Parcial
Sistema de Gestión Administrativa, módulo Activos	Libre	Web	Parcial

Los sistemas analizados sólo cubren parcialmente las necesidades del negocio, ya que para facilitar la generación de reportes a los distintos niveles es necesaria la gestión de los estados técnicos y clasificaciones de los activos tecnológicos. Esto requiere una integración con el sistema contable-financiero de la Universidad en la que se comparta la información relacionada con los inmuebles y activos.

A pesar de que el módulo Activos del Sistema de Gestión Administrativa genera informes consolidados con los datos que obtiene del sistema contable financiero de la universidad, estos informes no pueden ser personalizados y solo reflejan la información registrada en *Assets NS*. Lo que significa que no está diseñado para la generación de partes tecnológicos, donde se ingrese información sobre los activos de un local.

El análisis y estudio de estas herramientas contribuyó a comprender en profundidad necesidades del sistema, además, proporcionaron una visión más general del negocio contribuyendo al levantamiento de nuevos requisitos. Algunas de las funcionalidades que son tomadas en consideración son la generación dinámica de reportes suscitada durante el análisis de los sistemas SIGE y *AssetTrack*, y el envío de notificaciones e informes vía *e-mail* sugerida durante el estudio de *SysAid* y *ManageEngine*. La investigación del módulo “Activos” del Sistema de Gestión Administrativa Kainos permitió comprender su estructura y la manera en que presenta la información que obtiene de *Asset NS*. Proporcionó además, una

primera aproximación a los esquemas y bases de datos del SGU que suministran la información necesaria para la realización de las tareas del módulo a desarrollar.

1.4 Metodología a utilizar para la implementación de la solución

Una **metodología** es un sistema de principios o reglas generales de las cuales procedimientos y métodos específicos pueden ser derivados para interpretar o resolver diferentes problemas en el ámbito de una disciplina en particular. A diferencia de un algoritmo una metodología no es una fórmula sino un conjunto de prácticas [5].

Una **metodología de desarrollo de software** o una metodología de desarrollo de sistemas en ingeniería de software es un marco de trabajo que se utiliza para estructurar, planificar y controlar el proceso de desarrollo de un sistema de información. La metodología puede incluir la definición previa de entregables específicos y artefactos creados por un equipo de proyecto para desarrollar o mantener una aplicación.

Para la selección de la metodología se ha elegido un enfoque ágil dado que el equipo de desarrollo es pequeño, el software es en general un sistema de gestión por lo que la implementación de un módulo conlleva poca complejidad. Además, los clientes y usuarios están disponibles, y los requisitos y el ambiente son inestable.

Se ha decidido usar *Scrum* para el desarrollo de la solución propuesta dado que la aplicación SGU es un software de gestión y los proyectos de este tipo se caracterizan por ser de mediana o baja complejidad. Los usuarios finales se encuentran en la propia institución, por lo que se pueden realizar entrevistas y pruebas de aceptación frecuentemente. Además, se prevé que en estos encuentros con el cliente se definan nuevos requisitos o evolucionen los que ya habían sido especificados. Por último, el equipo de desarrollo del Sistema de Gestión Universitaria utiliza esta metodología para estructurar, planificar y controlar el proceso de desarrollo de la aplicación por tanto es la mejor alternativa para mantener la uniformidad. A continuación se realiza una caracterización de la misma.

1.4.1 Scrum

Scrum es una metodología de desarrollo ágil para gestionar proyectos de software. Dicha gestión no se basa en el seguimiento de un plan, sino en la adaptación continua a las circunstancias de la evolución del proyecto. Su uso es recomendado en proyectos con entornos complejos, donde se necesita obtener

resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum propone prácticas de refactorización en las tareas de diseño y codificación para evitar los problemas de degradación del sistema o de la arquitectura por la evolución continua del producto.

1.5 Marco de trabajo a utilizar en el desarrollo de la propuesta solución

El módulo a desarrollar formará parte del Sistema de Gestión Universitaria responsable de la informatización de diferentes procesos de la UCI. Por tanto este componente debe implementarse sobre una arquitectura que ya ha sido definida por los arquitectos de la aplicación. El equipo de desarrollo exige el uso del marco de trabajo GUUD, en la implementación de cualquier módulo o subsistema que se integre a esta arquitectura.

El marco de trabajo constituye un elemento importante a la hora de desarrollar una solución, pues provee una arquitectura sólida y más extensible para la implementación de los diferentes componentes en un sistema informático. Con la utilización del marco de trabajo se obtendrá un rápido desarrollo y una mayor calidad del producto, pues permite a los programadores, analistas y todo el equipo de desarrollo compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación que se desea construir.

1.5.1 GUUD 2.0

El Departamento de Desarrollo de la Dirección de Informatización (DIN) define para el desarrollo de la propuesta de solución como marco de trabajo GUUD. Este constituye un híbrido entre el marco de trabajo PHP, CodeIgniter y la biblioteca JavaScript, jQuery. El equipo de desarrollo propone el uso de CodeIgniter teniendo en consideración algunas de sus principales características como son: versatilidad, facilidad de instalación, flexibilidad, ligereza y documentación amplia y específica. Por otra parte la biblioteca jQuery ofrece un conjunto de funcionalidades, para el fácil manejo de JavaScript, que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio. En ambos casos se tuvo en cuenta que al ser productos de software libre su uso es ilimitado, y libre de cualquier costo.

1.5.1.1 jQuery 1.9

jQuery es una biblioteca JavaScript ligera y rápida de código abierto publicada bajo las licencias MIT (*Massachusetts Institute of Technology*, Instituto Tecnológico de Massachusetts) y *GNU* GPL (*GNU*

General Public License, Licencia Pública General de *GNU*). Dispone de un mecanismo de selección robusto y eficiente, lo que facilita a los desarrolladores la recuperación de la pieza exacta del documento que necesita ser inspeccionada o manipulada. Maneja ciertas cuestiones de compatibilidad entre navegadores asegurando que el código escrito se comporte exactamente igual en la mayoría de los navegadores.

Además, esta biblioteca puede cambiar las clases o las propiedades de estilo individuales aplicadas a una parte del documento incluso después de que la página haya sido renderizada. No se limita solo a cambios estéticos. Puede modificar el contenido de un documento, por ejemplo, el texto se puede cambiar, las imágenes se pueden insertar o intercambiar, las listas se pueden reordenar o la estructura completa del HTML se puede reescribir y ampliar. Todo ello es realizado a través de una única Interfaz de programación de aplicaciones (API) fácil de usar.

1.5.1.2 CodeIgniter 1.7.2

CodeIgniter es un marco de desarrollo, un kit de herramientas para las personas que crean aplicaciones web utilizando PHP. Su objetivo es permitir un desarrollo de proyectos más rápido de lo que se podría si se escribiera código desde cero, al proporcionar un amplio conjunto de bibliotecas para las tareas más comunes, así como una interfaz simple y una estructura lógica para acceder a estas bibliotecas. Permite al desarrollador concentrarse creativamente en el proyecto al minimizar la cantidad de código necesario para una tarea determinada. Se publica bajo la licencia de código abierto MIT, por lo que puede ser usado libremente.

El sistema central solo requiere un conjunto de pequeñas bibliotecas y las adicionales se cargan dinámicamente a pedido, según las necesidades de un proceso dado, por lo que el sistema base es ágil y rápido. CodeIgniter utiliza el enfoque Modelo-Vista-Controlador, que permite una gran separación entre la lógica y la presentación. Esto es particularmente bueno para los proyectos en los que los diseñadores están trabajando con sus archivos de plantilla, ya que el código que contienen estos archivos se minimizará. Las URL generadas por CodeIgniter son limpias y amigables para los motores de búsqueda. En lugar de utilizar el enfoque estándar de "cadena de consulta" para las URL que es sinónimo de sistemas dinámicos, este marco de trabajo utiliza un enfoque basado en segmentos.

Incluye una amplia gama de bibliotecas que permiten las tareas de desarrollo web más comunes, como acceder a una base de datos, enviar correos electrónicos, validar datos de formularios, mantener

sesiones, manipular imágenes, trabajar con datos XML-RPC[6], entre otros. El sistema se puede extender fácilmente mediante el uso de bibliotecas propias, *helpers*[7] o mediante extensiones de clase o enlaces del sistema. No requiere de un analizador de plantillas, aunque tiene uno integrado que puede usarse opcionalmente. CodeIgniter es un sistema dinámicamente instanciado, ligeramente acoplado con alta singularidad de componentes diseñados para conservar la simplicidad, la flexibilidad y el alto rendimiento en un paquete de tamaño reducido [6].

1.6 Lenguajes a utilizar en la implementación de la propuesta de solución.

Las políticas de desarrollo de la DIN definen GUUD 2.0 como marco de trabajo para el desarrollo de la solución web 'Sistema de Gestión Universitaria' lo que implica el uso de los lenguajes de programación HTML, CSS, Javascript, PHP para la implementación del mismo. Adicionalmente se utilizará UML para el modelado del sistema por ser un estándar mundial que facilita esta tarea, y cuyo uso constituye un aspecto importante en el desarrollo de software orientado a objetos.

1.6.1 UML 2.5

El Lenguaje Unificado de Modelado (UML) es el lenguaje de modelado estándar para el desarrollo de software y sistemas. Utiliza la notación gráfica para crear modelos visuales de sistemas de software y admite conceptos de desarrollo de alto nivel, como marcos, patrones y colaboraciones.

1.6.2 XHTML

HTML es un lenguaje descriptivo que especifica la estructura de una página web. Un documento HTML es un archivo de texto plano organizado con elementos que se encuentran rodeados por etiquetas de apertura y cierre coincidentes. Cada etiqueta comienza y termina con corchetes angulares (<>). Hay algunas etiquetas vacías que no pueden encerrar ningún texto, por ejemplo <img>. Puede extender las etiquetas HTML con atributos, lo que proporciona información adicional que afecta la forma en que el navegador interpreta el elemento. Un archivo HTML normalmente se guarda con una extensión .htm o .html, servida por un servidor web, y puede ser procesada por cualquier navegador web.

[6] XML-RPC es un protocolo de llamada a procedimiento remoto que usa XML para codificar los datos y HTTP como protocolo de transmisión de mensajes.

[7] *Helper*: es un archivo que contiene un conjunto de funciones, que apuntan a un rol en particular. Por ejemplo, un file *helper* contiene funciones para manejar archivos y un *text helper* tiene funciones para manipular texto.

XHTML (*EXtensible HyperText Markup Language*, Lenguaje Extensible de Marcado de Hipertexto) fue desarrollado combinando las fortalezas de HTML y XML para hacer HTML más extensible y aumentar la interoperabilidad con otros formatos de datos. Hay dos razones principales detrás de la creación de XHTML: crear un estándar más estricto para hacer páginas web, reduciendo las incompatibilidades entre los navegadores.

Las principales diferencias entre HTML y XHTML en lo que se refiere a la estructura del documento, es que las etiquetas DOCTYPE, html, body, *head*, *title* y *body* son requeridas.

1.6.3 JavaScript 1.8.5

La aplicación del marco de trabajo GUUD y en este de la biblioteca jQuery implica necesariamente el uso de JavaScript como lenguaje de programación para agregar interactividad y otras características dinámicas a la aplicación web. Es un lenguaje dinámico basado en prototipos, de múltiples paradigmas, que admite estilos orientados a objetos, imperativos y declarativos (por ejemplo, programación funcional). JavaScript (JS) es un lenguaje de programación compilado interpretado ligero o Just In Time (justo a tiempo) por ello se utiliza generalmente en el lado del cliente, pero puede usarse en el lado del servidor, por ejemplo, utilizando paquetes como *Node*.js.

Es interpretado principalmente en el navegador permitiendo a los desarrolladores manipular el contenido de la página web a través del DOM[8], manipular los datos con *Ajax* e *IndexedDB*, dibujar gráficos con lienzo, interactuar con el dispositivo que ejecuta el navegador, entre otros. JavaScript es uno de los lenguajes más utilizados en el mundo, debido al reciente crecimiento y mejora del rendimiento de las API disponibles y compatibilidad con la mayoría de los navegadores.

1.6.4 CSS

CSS es un lenguaje declarativo que controla el aspecto de las páginas web en el navegador. El navegador aplica declaraciones de estilo a los elementos seleccionados para mostrarlos correctamente. Una declaración de estilo contiene las propiedades y sus valores, que determinan el aspecto de una página

[8] El DOM (*Document Object Model*, Modelo de Objetos del Documento) es una API que representa e interactúa con cualquier documento HTML o XML. El DOM es un modelo de documento cargado en el navegador y representa el documento como un árbol de nodos, donde cada nodo representa parte del documento (por ejemplo, un elemento, una cadena de texto o un comentario).

web. CSS es una de las tres tecnologías web principales, junto con HTML y JavaScript. Usualmente diseña elementos HTML, pero se puede usar con otros lenguajes de marcado como SVG o XML.

1.6.5 PHP 7

PHP es un lenguaje de *scripts* de servidor de código abierto que se puede incrustar en HTML para crear aplicaciones web y sitios web dinámicos.

PHP resulta familiar para programadores de C y *Perl* lo que favorece la curva de aprendizaje del lenguaje. Es sencillo, no se tienen que incluir bibliotecas ni directivas de compilación especiales, el motor de PHP lee el código secuencialmente y si es correcto sistemáticamente se ejecuta. Es seguro, ya que proporciona a los desarrolladores y administradores un conjunto flexible y eficiente de salvaguardas de seguridad. Estas salvaguardas se pueden dividir en dos marcos de referencia: nivel de sistema y nivel de aplicación. Los *scripts* PHP se compilan completamente en el lado del servidor y la respuesta es enviada al usuario esto significa que es independiente del navegador.

1.7 Herramientas a utilizar para el desarrollo de la solución

Dado el proceso de migración a software libre al cual se somete la universidad y por ende la Dirección de Informatización, el equipo de desarrollo exige el uso de herramientas libres para la implementación del Sistema de Gestión Universitaria. Siendo de estricto uso el servidor web Nginx y el servidor de base de datos PostgreSQL pues constituyen parte del entorno de despliegue de este sistema.

1.7.1 Visual Paradigm 15.0

Visual Paradigm es una herramienta CASE (Computer Aid-Software Engineering, Ingeniería de Software Asistida por Computación) que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción y despliegue. Permite la gestión de proyectos con estándares y plantillas abiertas y bien establecidas y ampliamente utilizadas como TOGAF o PMBOK que aseguran que las iniciativas de IT estén alineadas con la visión empresarial, objetivos y estrategias.

Contribuye de una manera rápida a la construcción de aplicaciones de mayor calidad a un menor costo pues permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. Admite la generación y reversión de bases de datos para las bases de datos populares en el mercado, incluyendo MySQL, MS SQL Server, Oracle, Sybase, PostgreSQL, HSQL, MariaDB, Derby, Informix, Firebird, SQLite, IBM DB2, FrontBase, H2, Amazon Redshift y entre

otras. Soporta la revisión ortográfica, brindando sugerencias para los idiomas: inglés, español, francés, alemán y portugués.

1.7.2 Git 2.17

Git es un sistema de gestión de código fuente (*Source Code Management*, SCM) distribuido, de código abierto y gratuito. Facilita el manejo de bases de código con equipos de desarrollo distribuidos. Lo que lo diferencia de los sistemas SCM anteriores es la capacidad de realizar operaciones comunes (bifurcación, confirmación, entre otros.) en su máquina de desarrollo local, sin tener que cambiar el repositorio principal o incluso tener acceso de escritura [7].

1.7.3 NGINX 1.14

NGINX es un software de código abierto para servidores web, *proxy* inverso, almacenamiento en caché, balanceo de carga, transmisión de medios y entre otros. Además de sus capacidades de servidor HTTP, puede funcionar como un servidor *proxy* para correo electrónico (IMAP, POP3 y SMTP) y un *proxy* inverso y un equilibrador de carga para servidores HTTP, TCP y UDP [8].

Es conocido por su alto rendimiento, estabilidad, amplio conjunto de características, configuración simple y bajo consumo de recursos. Es uno de los pocos servidores escritos para solucionar el problema C10K[9]. A diferencia de los servidores tradicionales, no se basa en subprocesos para manejar las solicitudes. En su lugar, utiliza una arquitectura subyacente escalable basada en eventos (asíncrona) que ha demostrado ser ideal para muchas tareas web más allá de servir contenido. Esta arquitectura utiliza cantidades pequeñas, pero más importantes, predecibles de memoria bajo carga. NGINX se escala en todas las direcciones: desde el VPS[10] más pequeño hasta grandes grupos de servidores [9].

1.7.4 NetBeans 8.2

NetBeans IDE[11] permite desarrollar aplicaciones de escritorio, móviles y web de Java, así como aplicaciones HTML5 con HTML, JavaScript y CSS. Proporciona un gran conjunto de herramientas para

[9] C10K, término que describe la dificultad que algunos servidores web experimentan al manejar grandes números (los 10K) de conexiones concurrentes.

[10] VPS (*Virtual Private Server*, Servidor Virtual Privado) es un servidor creado mediante la virtualización de software. Funciona como un servidor físico, pero es una instancia virtualizada creada dentro de un servidor.

[11] IDE (Intregrate Development Environment, Ambiente de Desarrollo Integrado).

desarrolladores de PHP y C / C ++. Es gratuito y de código abierto y tiene una gran comunidad de usuarios y desarrolladores en todo el mundo.

Un IDE es mucho más que un editor de texto. El editor de NetBeans agrega sangría a las líneas, combina palabras y corchetes, y resalta el código fuente de manera sintáctica y semántica. Le permite refactorizar fácilmente el código, con una gama de herramientas útiles y poderosas, mientras que proporciona plantillas de código, sugerencias de codificación y generadores de código.

El editor soporta muchos lenguajes desde Java, C / C ++, XML y HTML, hasta PHP, Groovy, Javadoc, JavaScript y JSP. Debido a que el editor es extensible, puede conectar soporte para muchos otros idiomas.

1.7.5 PostgreSQL 9.5

PostgreSQL es un sistema de código abierto para la gestión de bases de datos objeto relacionales compatible con una gran parte del estándar SQL. Ofrece muchas características integridad transaccional (ACID[12]), llaves foráneas, vistas, vistas actualizables, uniones externas, secuencias, lanzadores (triggers), consultas complejas (subconsultas), control de concurrencia y multiversión. En algunos aspectos, está diseñado para que sea extensible por los usuarios, por ejemplo, agregando nuevos tipos de datos, funciones, operadores, funciones agregadas, métodos de índice, lenguajes procedurales.

Tiene interfaces de programación nativas para C/C++, Java, .Net, *Perl*, *Python*, *Ruby*, Tcl, ODBC[13], entre otros. Y debido a la licencia liberal, PostgreSQL puede ser utilizado, modificado y distribuido por cualquier persona de forma gratuita para cualquier propósito, ya sea privado, comercial o académico.

1.7.6 pgAdmin III 1.22

pgAdmin III es un sistema completo de gestión y diseño de bases de datos PostgreSQL para sistemas Unix y Windows. Está disponible de forma gratuita según los términos de la licencia de PostgreSQL y puede redistribuirse siempre que se cumplan los términos de la licencia. El proyecto es administrado por el equipo de desarrollo pgAdmin.

[12] ACID (*Atomicity, Consistency, Integrity and Durability*, Atomicidad, Consistencia, Integridad y Durabilidad).

[13] *Open DataBase Connectivity* (ODBC) es un estándar de acceso a las bases de datos permite acceder a cualquier dato desde una aplicación, sin importar qué sistema de gestión de bases de datos (DBMS) almacene los datos.

Está diseñado para responder a las necesidades de todos los usuarios, desde la escritura de simples consultas SQL hasta el desarrollo de complejas base de datos. La interfaz gráfica soporta todas las características de PostgreSQL y facilita la administración. La aplicación incluye un editor SQL con énfasis de sintaxis, un editor de código del lado del servidor, un SQL/batch/shell agente de planeación de tareas, soporte para el motor de réplicas Slony-I, entre otros. La conexión puede hacerse utilizando TCP/IP o Unix Domain *Sockets* (en plataformas *nix), y puede ser encriptada con SSL por seguridad. No son necesarios drivers adicionales para comunicarse con el servidor de base de datos.

1.7.7 Firefox 67 Developer Tools

Inspector de página: Permite ver y editar el contenido y diseño de una página web. Visualiza muchos aspectos de la página, incluidos el modelo de cuadro, las animaciones y los diseños de cuadrícula.

Consola web: Registra información asociada con una página web: solicitudes de red, JavaScript, CSS, errores y advertencias de seguridad, así como mensajes de error, advertencia e informativos registrados explícitamente por el código JavaScript que se ejecuta en el contexto de la página. Permite interactuar con una página web mediante la ejecución de expresiones de JavaScript en el contexto de la página [10].

Monitor de red: El Monitor de red le muestra todas las solicitudes de red que realiza Firefox (por ejemplo, cuando carga una página, o debido a XMLHttpRequests), el tiempo que tarda cada solicitud y los detalles de cada solicitud. El Monitor de red registra las solicitudes de red siempre que la caja de herramientas esté abierta, incluso si el monitor de red en sí no está seleccionado. Esto significa que puede comenzar a depurar una página en, por ejemplo, la consola web, y luego cambiar al monitor de red para ver la actividad de la red sin tener que volver a cargar la página [11].

Herramientas de rendimiento: La herramienta de rendimiento le brinda información sobre la capacidad de respuesta general, el rendimiento de JavaScript y el diseño de su sitio. Con la herramienta de rendimiento, crea una grabación o perfil de su sitio durante un período de tiempo. Luego, la herramienta le muestra una descripción general de las cosas que hacía el navegador para representar su sitio en el perfil y un gráfico de la velocidad de fotogramas en el perfil. Obtendrá cuatro herramientas secundarias para examinar aspectos del perfil con más detalle:

- *Waterfall* (Cascada) muestra las diferentes operaciones que estaba realizando el navegador, como ejecutar el diseño, JavaScript, las repeticiones y la recolección de basura.

- *Call Tree* (Árbol de Llamadas) muestra las funciones de JavaScript en las que el navegador pasó la mayor parte de su tiempo.
- *Flame Chart* (Gráfico de Llamas) muestra la pila de llamadas de JavaScript en el transcurso de la grabación.
- *Allocations View* (Vista de Asignaciones) muestra las asignaciones a la pila hechas por el código en el transcurso de la grabación. Esta vista solo aparece se marca "Grabar asignaciones" en la configuración de la herramienta *Rendimiento* [12].

1.7.8 *Pencil* 3.0.4

Pencil está diseñado con el propósito de proporcionar una herramienta de creación de prototipos GUI gratuita y de código abierto que las personas pueden instalar y usar fácilmente para crear maquetas en las plataformas de escritorio populares [13].

Ofrece varias colecciones de formas integradas para dibujar diferentes tipos de interfaz de usuario, desde plataformas de escritorio a móviles. También, se implementan funciones de dibujo populares para simplificar las operaciones de dibujo. Admite conectores que se pueden usar para "cablear" formas en un diagrama. También, está disponible una colección de formas de diagrama de flujo para dibujar diagramas. Soporta la salida del documento de dibujo en diferentes tipos de formatos. Puede exportar su dibujo como un conjunto de archivos PNG rasterizados o como una página web que se puede entregar a los espectadores. *Pencil*, admite la exportación de documentos a formatos populares, incluidos los documentos de texto OpenOffice / LibreOffice, Inkscape SVG y Adobe PDF [14].

1.8 Conclusiones del capítulo

En el presente capítulo se abordaron los elementos teóricos que dan sustento a la propuesta de solución. La definición de los principales conceptos asociados al dominio de la presente investigación y las relaciones entre estos, permitió alcanzar una mayor comprensión de la propuesta de solución.

El análisis de los sistemas homólogos, permitió identificar las tendencias en cuanto al desarrollo de herramientas informáticas para la gestión de inventarios tecnológicos. Las deficiencias que impiden que sean utilizadas con el objetivo de registrar el estado de la tecnología en la universidad, siendo determinante la sincronización con los datos reales del sistema contable financiero para garantizar la integridad de los mismos.

El análisis de la metodología de desarrollo, así como las herramientas, tecnologías y lenguajes de programación utilizados en su implementación, permitió especificar el ambiente de desarrollo para la propuesta de solución. Siendo capaz de cumplir con las necesidades de integración con el Sistema de Gestión Universitaria como uno de sus módulos.

Capítulo 2: Análisis y diseño del sistema.

Una de las prioridades cuando se desea desarrollar un software, es establecer un entendimiento entre el cliente y el equipo de trabajo en relación con los objetivos a lograr, realizando un correcto análisis y diseño de dicho sistema. El objetivo de este capítulo es presentar los resultados que se obtuvieron una vez cumplidas las fases de análisis y diseño que propone la metodología *Scrum*. Para ello se detallan las características del módulo y se presenta el conjunto de artefactos generados durante el proceso, entre ellos, las reglas de negocio, el listado de requisitos funcionales y no funcionales, y el modelo de base de datos.

2.1 Modelado del negocio

El modelo de negocio es una descripción o diseño de las diferentes partes de una empresa u organización que muestra la forma en que trabajarán juntas para la operación exitosa de un negocio. En esta sección se muestra como está conformado el negocio en cuestión, especificando la relación entre los conceptos y su funcionamiento a través de procesos que manejan los especialistas al realizar el control de los activos tecnológicos.

2.1.1 Modelo conceptual

El modelo conceptual es un modelo descriptivo de un sistema basado en supuestos cualitativos sobre sus elementos, sus interrelaciones y los límites del sistema [15].

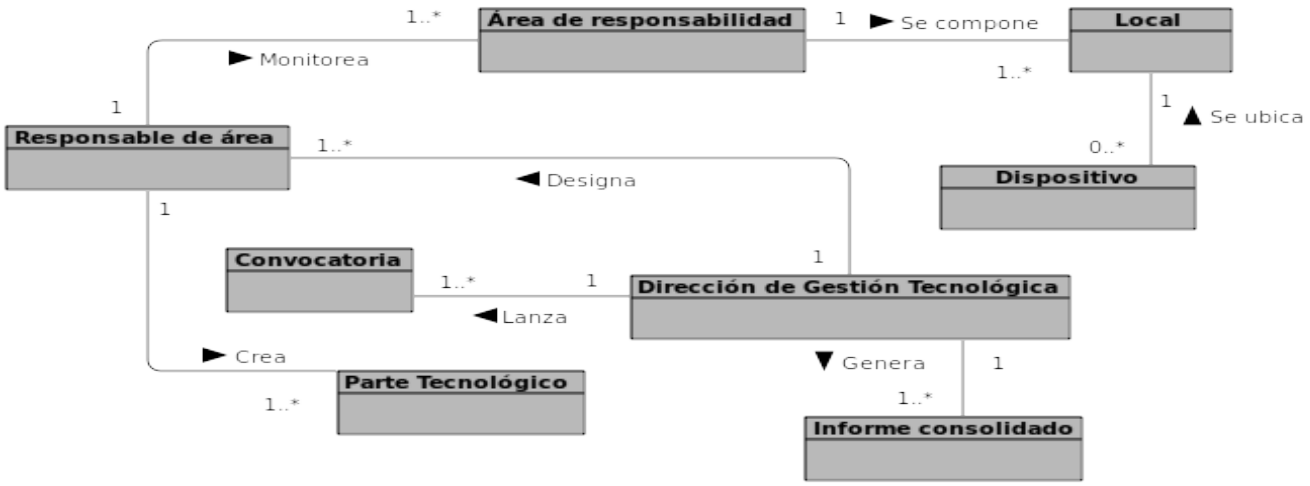


Figura 1: Modelo conceptual

- A partir del análisis realizado a los procesos que requieren ser informatizados, se realizó un modelo conceptual donde se aprecian los diferentes conceptos asociados a la gestión de los partes tecnológicos e informes consolidados y la relación estructural de los mismos. Para una mejor comprensión, a continuación, se describen cada uno de los conceptos que intervienen en el modelo conceptual mostrado.
- Área de responsabilidad: Se refiere a cada una de los centros de costo, entre los que se encuentran: facultades, centros de producción, vicerectorías y las distintas direcciones.
- Responsable de área: Persona encargada de llevar el control de los partes tecnológicos de un área de responsabilidad.
- Dirección de Gestión Tecnológica: Debe definir quiénes serán los responsables de área, además, tiene la responsabilidad de lanzar las convocatorias para la creación de los partes, generar los informes consolidados una vez que los partes han sido creados.
- Dispositivos: Hace referencia al tipo de medio tecnológico (computadoras, monitores, baterías, impresoras, *laptops*, entre otros). Los dispositivos pueden tener asociado un estado técnico (buen estado, mal estado, desuso, entre otros), y en algunos casos características (si un monitor posee pantalla LCD o CRT, si una PC tiene microprocesador i3, i5 ó i7).
- Local: Ubicación física donde se encuentran los dispositivos, por ejemplo, laboratorio docente, laboratorio de producción, aula, oficina, entre otros.
- Parte tecnológico: Informe donde queda plasmado el estado técnico de los medios tecnológicos por local, es elaborado por los responsables de área y entregado a la Dirección de Gestión Tecnológica en el tiempo establecido.

2.1.2 Descripción de las reglas del negocio

Las reglas del negocio son un eslabón fundamental en la descripción de las políticas, normas, operaciones, definiciones y restricciones de una organización, permitiendo alcanzar los objetivos plasmados en la misión de la empresa.

A continuación se definen un conjunto de reglas que orientan el desarrollo, el funcionamiento y organización del módulo para la gestión de los estados técnicos de los activos tecnológicos:

- El módulo define solo 2 roles, Director de Tecnología y Responsable de Área. El Director de Tecnología tienes los privilegios necesarios para gestionar los responsables de área, las

convocatorias, definir los estados de un parte o convocatoria, definir sobre cuáles dispositivos se deber crear los partes y cuáles son las características o estados de cada uno de ellos que deben quedar reflejadas en el parte. Este rol, también, tiene la responsabilidad de generar informes a partir de la información consolidada de los partes. El Responsable de Área debe crear los partes según los metadatos definidos por el Director de Tecnología.

- Solo se crearán partes de los elementos que han sido activados por la dirección de tecnología, permitiendo crear partes personalizados. Por ejemplo, podrán elegirse qué dispositivos, atributos o características serán necesarias para generar el parte.
- El Director de Gestión Tecnológica puede definir atributos asociados a un dispositivo y características, asociadas a estos.
- El Director de Gestión Tecnológica puede precisar qué características de un atributo se relacionan con otras de algún otro atributo del dispositivo, por ejemplo: Asociar las características Pantalla DDR y Estado Técnico, Roto.
- Una característica puede asociarse a otra que pertenezca a otro atributo del mismo dispositivo.
- Las PC rotas son aquellas que se encuentran rotas total o parcialmente, pero que el área desea conservar o arreglar.
- Las PC en desuso son PC que no se usan y que el área no desea tener en su inventario. Pueden ser PC de bajas prestaciones en buen estado, rotas que no quieran reparar o listas para baja técnica.
- Los monitores CRT rotos, UPS rotas, *laptops* rotas e Impresoras rotas son aquellos que el área desea conservar o arreglar.
- Los monitores, UPS, *laptops* e impresoras en desuso son aquellos que no se usan y que el área no desea tener en su inventario. Pueden ser en buen estado, rotos o listos para baja técnica.
- El total de cada tipo de medio tecnológico es una cifra predefinida que se extrae del Sistema Contable Financiero de la Universidad *ASSETS* y se valida en el caso de:
  - Monitores = LCD en BET + LCD rotos + CRT en BET + CRT rotos + Desuso en BET + Desuso rotos
  - PC = PC en BET + PC rotos + Desuso en BET + Desuso rotos (además, se debe validad que los Core i <= total)
  - UPS *Laptop* Impresora = BET + Rotos + Desuso en BET + Desuso rotos

BET: Buen estado Técnico

- Al parte se le almacena la fecha en la que fue creado, que debe estar entre la fecha de inicio y fecha de fin de la convocatoria.
- La opción de crear un nuevo parte solo se visualiza si hay una convocatoria abierta y esta área no lo ha creado y solo puede ser modificado el parte en el caso que la convocatoria aún esté abierta. En el caso de modificación se deben seguir validando las cantidades respecto al total como en el caso de creación.
- Los tipos de locales se deben gestionar porque en el esquema existente un salón puede estar clasificado como aula, o sea, realmente cuando se refiere a tipo de local se debe crear una subclasificación para las clasificaciones, por ejemplo:
  - clasificación: aula, subclasificación: salón.
  - (en las facultades) clasificación: laboratorio, subclasificación: laboratorio docente.
  - (en los centros de desarrollo) clasificación: laboratorio, subclasificación: laboratorio de producción.
- Solo se visualizan aquellos locales que tengan medios tecnológicos.
- Los locales se extraen desde el esquema ‘activos’ organizados por estructura y se almacenan en esta base de datos cuando se le asigna la subclasificación.
- Los tipos de medios tecnológicos se extraen del esquema ‘activos’ del Sistema de Gestión Universitaria para que puedan ser activados o no. Actualmente solo es requerida la información de PC’s, Impresoras, UPS’s, *laptops*, Monitores.
- El reporte consolidado solo se presenta al Director de Gestión Tecnológica. Este informe debe mostrar un vista integral de la UCI con todas las áreas subordinadas a la rectoría y permitiendo escoger un área específico para mostrar los datos de la misma. Los datos de cada tipo de medio son agrupados según la clasificación del tipo de local que tiene cada área subordinada, por ejemplo: las facultades tienen laboratorios docentes, departamentos docentes, oficinas y aulas.
- La convocatoria se crea con una fecha de inicio y fin. Cuando la convocatoria cierra por llegar a la fecha de fin cambia por defecto el estado a todos los partes que fueron creados a estado archivado.
- Con el objetivo de mantener la integridad en las bases de datos los nomencladores no son eliminados en vez de ello son desactivados.

2.2 Descripción de la propuesta de solución

El objetivo de la propuesta de solución es facilitar la creación de reportes sobre el estado técnico de los activos tecnológicos en las distintas áreas de la universidad y la consolidación de estos datos en informes, de manera que el proceso sea realizado de una forma más eficiente. El módulo aprovechará la información registrada en el Sistema Contable Financiero de la universidad accesible desde el Sistema de Gestión Universitaria lo que significa que en todo momento los datos serán homogéneos y consistentes. Este módulo explotará las ventajas que ofrece la integración al Sistema de Gestión Universitaria, por ejemplo, la autenticación, seguridad, gestión de sesiones, manejo de excepciones, librerías, funciones para generar reportes en formato pdf o xsl, entre otras.

El módulo cuenta con 2 agrupaciones funcionales, Partes Tecnológicos y Configuración. Cada una de las funcionalidades dentro de estas agrupaciones tendrá un nivel de acceso restringido a un rol específico. Por ejemplo, la generación de informes que se encuentra dentro de Partes Tecnológicos solo será accesible para el Director de Gestión Tecnológica, mientras que *Reportes* diseñado para permitir la creación de reportes sobre el estado técnico de los activos en un área, agrupados según el local será visible para ambos roles.

La agrupación funcional Configuración solo será visible para el Director de Gestión Tecnológica, que debe definir quiénes serán los responsables para asociar a ellos una o varias áreas. Otra funcionalidad es la de permitir definir sobre cuáles activos tecnológicos se va realizar el reporte. Por último, permite la creación de convocatorias que notifican a todos los responsable de área cuándo comenzar a crear los reportes y antes de qué fecha terminarlos.

Para que los responsables de área puedan crear partes el Director de Gestión Tecnológica debe seleccionar los dispositivos que aparecerán en los partes, a cada dispositivo debe asignarle las características y estados técnicos a tener en cuenta en el parte. Para lograr esto se ha definido en el negocio 2 abstracciones (atributos y estados de explotación) que permiten agrupar las características y estados técnicos respectivamente. Cada característica debe tener asociado un estado técnico aunque estos podrían no tener asociada ninguna característica. Adicionalmente un estado técnico con una característica asociada puede ser marcado como no ‘Exclusivo’ si se debe tener en cuenta en el parte (sin la característica asociada).

2.3 Definición de las técnicas de obtención de requisitos

Los requisitos son representaciones documentadas de la condición, capacidad, propiedad o característica de un sistema o parte de este, son necesarios para solucionar un problema o alcanzar un objetivo en un contexto real y portan valor y utilidad al cliente y a los usuarios finales. Las técnicas utilizadas para la obtención de información fueron las entrevistas a los especialistas y el análisis de la documentación, las cual permitieron una mejor comprensión de las necesidades del cliente y las especificaciones del sistema, facilitando una mejor perspectiva de los requisitos funcionales y no funcionales.

La entrevista es un método para el levantamiento de la información que se encuentra basado en preguntas. Generalmente están destinadas a individuos que se encuentran dentro de la organización, aunque en algunas ocasiones pueden ser aplicadas a personas externas que interactúan con el sistema. Esta herramienta permitió analizar en detalle las necesidades del cliente y posibilitó el estudio de las expectativas del mismo respecto a la propuesta de solución, descubriendo sus motivaciones y actitudes.

La técnica de obtención de requisitos mediante el análisis de la documentación consiste en obtener la información sobre los requerimientos funcionales y no funcionales de software a partir de documentos que ya están elaborados. Dicha técnica es útil cuando los expertos en la materia no están disponibles para ser entrevistados o ya no forman parte de la organización. Permite además, introducir al analista al dominio de operación y el vocabulario que se utiliza en el negocio.

Documentación consultada: Modelo de estado de la tecnología (equipos de cómputo) de la UCI.

Los prototipos suelen consistir en versiones reducidas, demos o conjuntos de pantallas (que no son totalmente operativos) de la aplicación. La elaboración de los prototipos permitió mostrar las funcionalidades de la propuesta de solución para su validación por parte de los interesados. Este tipo de técnica sirve para eliminar dudas sobre lo que desea el cliente y además, para desarrollar la interfaz más amigable.

2.4 Requisitos funcionales

Requerimientos funcionales son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos casos, los requerimientos funcionales también explican lo que no debe hacer el sistema.

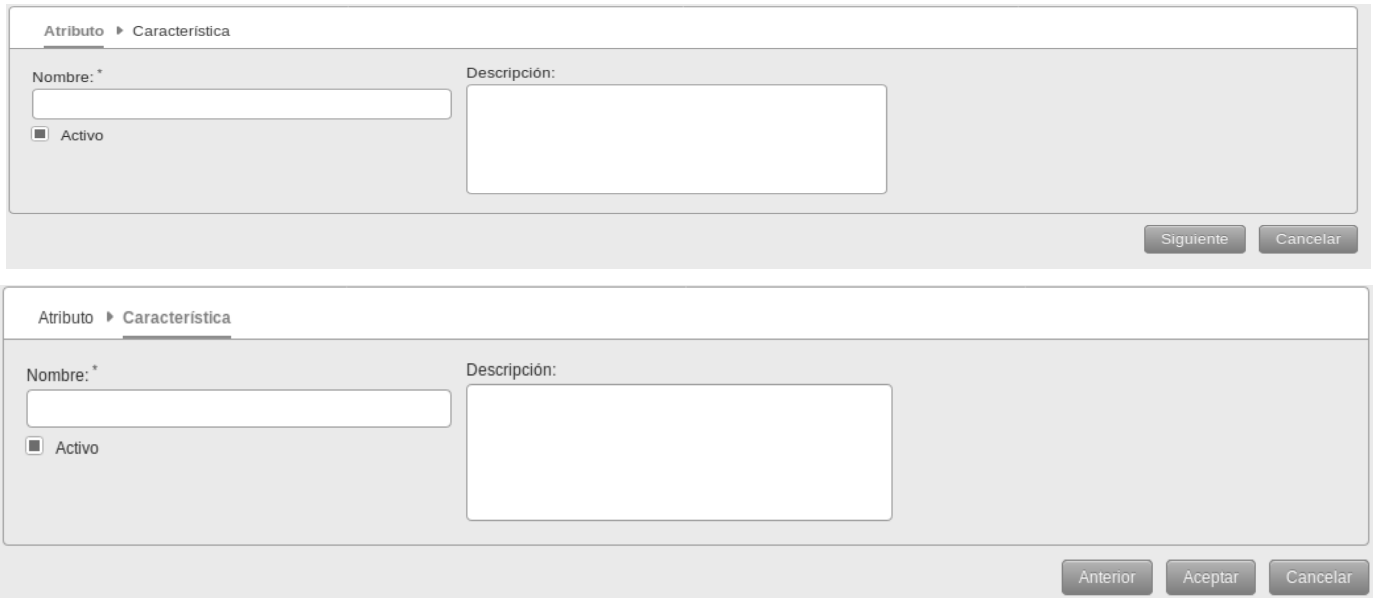
Tabla 3: Requisitos funcionales del módulo de gestión de partes tecnológicos (Elaboración propia).

Agrupación funcional	No.	Nombre	Complejidad	Prioridad
Partes tecnológicos	RF1	Listar partes tecnológicos.	Alta	Alta
	RF2	Crear parte tecnológico.	Alta	Alta
	RF3	Modificar parte tecnológico.	Alta	Alta
	RF4	Ver detalles del parte tecnológico.	Alta	Alta
	RF5	Eliminar parte tecnológico.	Alta	Alta
	RF6	Exportar parte tecnológico.	Alta	Alta
	RF7	Imprimir parte tecnológico.	Alta	Alta
Informes tecnológicos	RF8	Listar informes tecnológicos.	Alta	Alta
	RF9	Ver detalles del informe tecnológico.	Alta	Alta
	RF10	Exportar informe tecnológico.	Alta	Alta
	RF11	Imprimir informe tecnológico.	Alta	Alta
Convocatorias	RF12	Listar convocatorias.	Media	Media
	RF13	Crear convocatoria.	Media	Media
	RF14	Modificar convocatoria.	Media	Media
	RF15	Ver detalles de las convocatorias.	Media	Media
	RF16	Eliminar convocatoria.	Media	Media
Responsables de áreas	RF17	Listar responsables de áreas.	Media	Media
	RF18	Crear responsable de área.	Media	Media
	RF19	Modificar responsable de área.	Media	Media
	RF20	Ver detalles del responsable de área.	Media	Media
	RF21	Eliminar responsable de área.	Media	Media
Atributos	RF22	Listar atributos de un dispositivo.	Baja	Baja
	RF23	Crear atributo de un dispositivo.	Baja	Baja
	RF24	Modificar atributo de un dispositivo.	Baja	Baja
	RF25	Ver detalles del atributo de un dispositivo.	Baja	Baja
Características	RF26	Listar características de un dispositivo.	Baja	Baja
	RF27	Crear característica de un dispositivo.	Baja	Baja
	RF28	Modificar característica de un dispositivo.	Baja	Baja
	RF29	Ver detalles de la característica de un dispositivo.	Baja	Baja
Dispositivo	RF30	Listar dispositivo.	Baja	Baja
	RF31	Activar/Desactivar los dispositivo.	Baja	Baja
	RF32	Ver detalles de un dispositivo.	Baja	Baja

2.5 Descripción de requisitos

Con el objetivo de evitar errores y representar los requisitos funcionales de una manera consistente y detallada para el desarrollo del componente, se hace una descripción de requisitos. Para ello se establecen la prioridad, la complejidad, una breve descripción de las acciones principales y el prototipo de la interfaz para el desarrollo de la funcionalidad para su posterior implementación.

Tabla 4: Descripción del requisito Crear atributo de un dispositivo (Elaboración propia).

Número: RF53 Crear atributo de un dispositivo	**Requisito:** Crear atributo
Programador: José Carlos Rodríguez Texidor	**Iteración asignada:** Segunda
Prioridad: Baja	**Tiempo estimado:** 2 horas
Riesgo de desarrollo: Ninguno	**Tiempo real:** 2 horas
Descripción: El requisito permite crear un nuevo atributo. Inicialmente se muestra un árbol cuyos nodos raíz son los dispositivos, cada uno de ellos tiene la opción de crear atributo. Al dar *click* en este botón, se abre un formulario compuesto, en el primer componente se introducen los datos para crear un nuevo atributo y en la segunda los datos para crear una nueva característica asociada a este.	
Observaciones: El nombre del atributo y la característica son campos requeridos.	
Prototipo elemental de interfaz de usuario: 	

2.6 Requisitos no funcionales

En ingeniería de sistemas e ingeniería de requisitos, un requisito no funcional (RNF) a menudo se denomina "atributos de calidad" de un sistema es un requisito que especifica criterios que se pueden usar para juzgar el funcionamiento de un sistema, en lugar de comportamientos específicos. Estos definen atributos del sistema como seguridad, confiabilidad, rendimiento, mantenibilidad, escalabilidad y facilidad de uso. Actúan como restricciones en el diseño del sistema en los diferentes *backlogs* asegurando la usabilidad y efectividad del mismo.

Tabla 5: Descripción de los requisitos no funcionales (Elaboración propia).

Usabilidad:	
RNF1.	El componente debe presentar un menú lateral y una barra de íconos flotantes que permitan el acceso rápido a la información por parte de los usuarios.
RNF2.	Las vistas del componente deben indicar en cada momento la acción que se está realizando, así como los íconos deben estar representados por una imagen acorde a la acción que se realiza.
RNF3.	Sólo se mostrarán a los usuarios aquellas acciones o informaciones del menú lateral a las que por su responsabilidad o rol dentro del negocio necesiten acceder.
Confiabilidad:	
RNF4.	El tratamiento de las excepciones permite un seguimiento hasta guardar información, acerca del lugar donde se produjo el error y de los parámetros utilizados en el sistema que lo provocaron. Cuando ocurre una excepción el sistema mostrará un mensaje explicativo del error ocurrido.
Eficiencia:	
RNF5.	El módulo deberá tener por cada transacción un tiempo de respuesta promedio de 1,5 segundos y un máximo de 5 segundos.
RNF6.	El módulo soportará la conexión simultánea de todos los posibles usuarios, con un promedio de 1000 y un máximo de 3000 según los requisitos de hardware definidos en este documento.
Soporte	
RNF7.	El componente cumplirá con las normas de codificación, conversiones para nomenclatura, bibliotecas de clase definidas para el Sistema de Gestión Universitaria.
RNF8.	El componente contará con toda la documentación definida en el expediente de proyecto asociada a su proceso de desarrollo para las actividades de soporte.
Restricciones de diseño	
RNF9.	El componente deberá ser desarrollado en su totalidad con tecnologías y componentes de código abierto.
RNF10.	El componente estará desarrollado con las herramientas definidas para el Sistema de Gestión Universitaria en el documento, Arquitectura de Software Vista de tecnología e infraestructural, que se encuentra en el expediente de

	proyecto del Núcleo.
RNF11.	El módulo cumplirá con la arquitectura de información definida para el Sistema de Gestión Universitaria en el documento, Arquitectura de Software Vista de Presentación, que se encuentra en el expediente de proyecto del Núcleo.
Interfaz:	
RNF12.	La interfaz de usuario se guiará por la vista de arquitectura de presentación definidas en el documento —Pautas de diseño XAUCE-SGUǁ.
Software:	
RNF13.	Para el despliegue del módulo se usará en el servidor de aplicaciones web Nginx 1.14, PHP en su versión 7 y las bibliotecas php7-ldap, php7-gd, php7-pgsql, php7-xsl.
RNF14.	Para el despliegue del componente se recomienda usar el servidor de bases de datos PostgreSQL 9.5 en el sistema operativo Linux 16.4 o una versión superior de cualquiera de los dos sistemas.
RNF15.	Para el uso del módulo se requiere un dispositivo cliente donde pueda ser instalado el navegador web Mozilla Firefox 65 o superior para el uso de la aplicación web.
Hardware	
RNF16.	La PC cliente deberá tener 1 GB disco duro disponible como mínimo y 512 MB de memoria RAM. La PC servidor de BD y la PC servidor web deberá tener los siguientes componentes de hardware: microprocesador de 8 núcleos, 4 GB de memoria RAM, 250 GB de disco duro y una fuente de 800 W como mínimo.
RNF17.	La comunicación entre el cliente y el servidor de aplicaciones se realiza a través del protocolo HTTPS.
Confidencialidad:	
RNF18.	La información manejada por el módulo está protegida de acceso no autorizado y divulgación.

2.7 Validación de requisitos

La validación de requisitos examina las descripciones para asegurar que todos los requisitos del sistema han sido establecidos sin ambigüedad, sin inconsistencias, sin omisiones, que los errores detectados hayan sido corregidos y que el resultado del trabajo se ajusta a los estándares establecidos para el proceso, el proyecto y el producto [16].

El objetivo es evitar los cambios innecesarios en un sistema ya implementado, pues ello implica generalmente cambiar el diseño y la implementación. Lo que conduce al aumento del costo del proyecto, el uso de fuerza de trabajo innecesaria y mayor consumo de tiempo. A continuación se presenta una descripción de la estrategia utilizada para la validación de los requisitos.

2.7.1 Estrategia de validación de requisitos

Para la validación de los requisitos se utilizaron las siguientes técnicas:

Revisión técnica formal de requisitos: se realizan reuniones donde se examinan las descripciones del sistema buscando errores en el contenido o en la interpretación, ideas donde se necesitan aclaraciones, información incompleta, inconsistencias, requisitos contradictorios o requisitos imposibles o inalcanzables.

Construcción de prototipos: la confección de los prototipos permite que el usuario pueda conocer cómo es la propuesta que el equipo de desarrollo implementa y puede aprobar la idea o corregir los elementos ajenos a los requisitos funcionales descritos.

Generación de casos de prueba: la elaboración de casos de prueba permite comprobar el cumplimiento de los requisitos funcionales y que estos presentan la calidad requerida. Resultado de aplicar los criterios de validación.

Interrogantes para validar los requisitos del cliente.

- ¿El proveedor del requisito es un proveedor válido?
- ¿El requisito está identificado como único?
- ¿El requisito es modificable?
- ¿El requisito no es ambiguo?
- ¿El requisito está completo?
- ¿El requisito es congruente con otros requisitos relacionados?
- ¿El requisito puede ser implementado?
- ¿El requisito puede ser probado?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El requisito está correcto?

Al concluir el proceso de revisión de requisitos se detectaron algunas inconsistencias que fueron erradicadas de inmediato. Entre las más comunes se pueden citar:

- Se interpretaron de forma incorrecta algunas de las funcionalidades y características solicitadas por el cliente.
- Errores ortográficos o tipográficos.

2.8 Diseño arquitectónico

La arquitectura de software consiste en un conjunto de patrones y abstracciones coherentes que describen la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre

ellos y el ambiente proporcionando el marco de referencia necesario para guiar el diseño, construcción y evolución del software. Su definición afecta el desempeño, la estabilidad, así como la capacidad de mantenimiento del sistema, es decir, la forma en que los componentes se organizan y comunican influye en los requerimientos no funcionales.

El diseño arquitectónico es la primera etapa en el proceso de diseño del software y es la que continúa a la ingeniería de requisitos, estando estrechamente relacionada con esta. Su propósito es determinar la organización del sistema definiendo la relación entre sus principales componentes estructurales, los estilos arquitectónicos y los patrones de diseño.

De manera general, puede decirse que un patrón es una descripción abstracta de la solución a un problema recurrente, que se ensayó y puso a prueba en diferentes sistemas y entornos. Un patrón incluye al problema y la solución así como las relaciones que los unen, debe describir una organización de sistema que ha tenido éxito en sistemas previos y que puede considerarse una guía para solucionar problemas del mismo tipo. El uso de patrones en la solución propuesta contribuye a la reusabilidad, extensión y mantenimiento.

Dado que actualmente no existe un consenso en cuanto a los términos exactos para referirse a patrones de la arquitectura de un sistema en sus distintos niveles de abstracción, se usarán las expresiones: estilo arquitectónico, patrón arquitectónico y patrón de diseño para representar estos niveles. Por tanto, el término estilo arquitectónico se usará para aludir a los patrones enfocados en la relación del sistema con otros sistemas, programas y componentes de programa. Patrón arquitectónico cuando se trate de la relación entre los componentes de la misma aplicación y los patrones de diseño cuando la relación sea entre clases u objetos.

2.8.1 Estilo arquitectónico

La propuesta de solución ha sido concebida como un módulo del Sistema de Gestión Universitaria, el cuál emplea el modelo cliente-servidor para definir su arquitectura en el más alto nivel de abstracción. Por ello resulta conveniente explotar los beneficios que ofrece dicha arquitectura. A continuación se introduce una descripción de dicho modelo.

La arquitectura cliente-servidor es un modelo de sistemas distribuidos en el que las tareas se reparten entre clientes y servidores. Los servidores son aplicaciones que proveen algún recurso o servicio y tienen un rol pasivo en la comunicación pues se mantienen a la espera o escucha peticiones. Al recibir una

nueva solicitud se procesa y se envía una respuesta al cliente, cuya identidad no necesitan conocer por adelantado. El cliente tiene un rol activo pues siempre inicia la comunicación enviando una solicitud al servidor, para ello debe conocer de antemano la identidad del servicio que invoca o necesita.

Una de las ventajas de esta arquitectura es que la capacidad de proceso está dividida entre clientes y servidores, lo que además, posibilita la centralización del control pues el acceso a los recursos y datos está determinado por los servidores.

El sistema se hospedará en el servidor web de aplicaciones Nginx instalado en uno de los servidores del nodo central de la universidad y se podrá acceder mediante la dirección URL https://sgu.uci.cu/. El uso de esta arquitectura permite que los usuarios desde cualquier dispositivo conectado a la red universitaria puedan acceder al sistema a través de un navegador web, logrando centralizar la información y facilitando el acceso a esta.

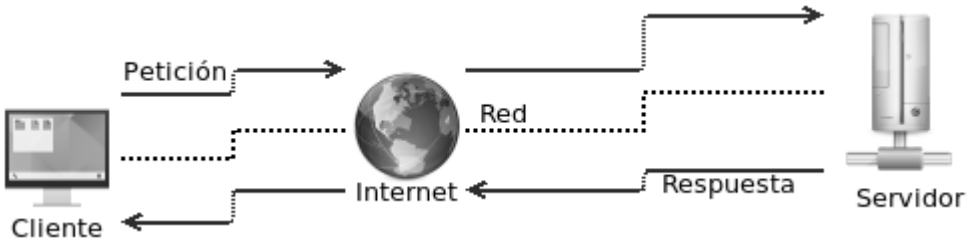


Figura 2: Modelo cliente-servidor

2.8.2 Patrón arquitectónico Modelo-Vista-Controlador

El marco de trabajo GUUD cuenta con un controlador frontal que inicializa los recursos básicos necesarios para usar Codelgniter. Cuando un cliente realiza una petición HTTP, el controlador frontal analiza la URI y a partir de ella determina qué clase controladora debe ser cargada para atender la solicitud.

Cada clase controladora implementa la lógica del negocio inherente a las acciones relacionadas con una funcionalidad y tiene asociada una o varias librerías responsables de procesar los datos que utiliza. De manera similar cada librería tiene asociado uno o varios modelos encargados del acceso a los datos. Luego de ser cargado el controlador, este examina la petición para determinar si solo debe cargar una vista determinada o si es necesario interactuar con la base de datos. En este último caso el controlador

envía los datos recibidos a una o varias librerías que invocan a las clase modelo necesarias para gestionar la información en la base de datos.

Cuando los datos son obtenidos, se retornan al controlador de aplicación en un proceso inverso al anterior. Posteriormente, el controlador carga estos datos a archivos escritos en HTML los cuales pueden incluir llamadas a archivos escritos en JavaScript para manejar dinámicamente su contenido. Finalmente, el resultado obtenido de todo este proceso es enviado al navegador web como respuesta a la petición inicial.

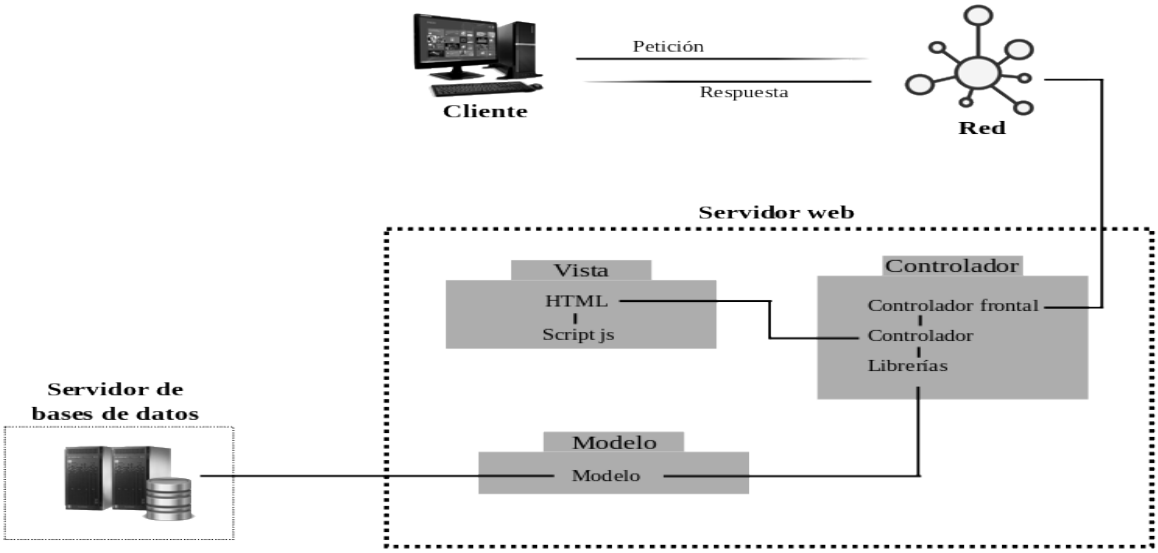


Figura 3: Arquitectura Modelo-Vista-Controlador en el marco de trabajo GUUD

2.8.3 Patrones de diseño GRASP

Los patrones GRASP (*Object-Oriented Design General Responsibility ASsignment Principles Patterns*, Patrones de Principios Generales para Asignar Responsabilidades en el Diseño Orientado a Objetos) describen los principios fundamentales de la asignación de responsabilidades a objetos, constituyen buenas prácticas y un apoyo para la enseñanza que permite comprender lo esencial del diseño de objetos aplicando un razonamiento sistemático, racional y explicable.

Patrón experto: Este patrón se aplica en ocasiones donde se necesita definir qué clase debería tener cierta responsabilidad. La solución es asignar la responsabilidad de la tarea al experto en la información, o

sea, a la clase que cuenta con la información necesaria para cumplir la responsabilidad. Al aplicar este patrón se mantiene el encapsulamiento de la información, puesto que los objetos utilizan su propia información para llevar a cabo las tareas.

Este patrón se ve reflejado en la clara asignación de responsabilidades a las clases implementadas, por ejemplo, la clase *tb_ncaracteristica* definida en el archivo *tb_ncaracteristica_mdl.php.* Esta es una de las clases que pertenecen a la capa modelo del sistema, y está específicamente destinada a la manipulación de los datos relacionados a las características de los dispositivos.

Patrón creador: Este patrón esclarece quién debería ser responsable de crear una nueva instancia de alguna clase. Propone que una clase debería crear las instancias de otra si agrega o contiene sus objetos o los datos de inicialización de dichos objetos, de forma tal que una instancia u objeto sólo pueda ser creada por la clase que contiene la información necesaria para ello.

Este patrón es utilizado en las clases JavaScript que deben crear listados u otros objetos generalmente pertenecientes a la biblioteca jQuery. El siguiente ejemplo de código muestra la creación de un objeto que crea una ventana con pestañas, usando como datos las vistas devueltas por los métodos *renderListar*, el nombre de cada pestaña y un identificador.

```
cargarTab: function () {
    $('#tabs').tab({
        tabs: [
            {
                url: system.core.site_url('parte_tecnologico/dispositivo/renderListar'),
                name: this.nlsLocale.tabDispositivo,
                id: 'tabDispositivo'
            },
            {
                url: system.core.site_url('parte_tecnologico/caracteristica/renderListar'),
                name: this.nlsLocale.tabCaracteristica,
                id: 'tabCaracteristica'
            },
        ],
        initial: 0, alwaysload: true
    });
},
```

Patrón controlador: El controlador es un intermediario entre la interfaz de usuario y las clases donde reside la lógica de la aplicación definiendo cuál de estas debería encargarse de atender un evento del sistema. El controlador no realiza mucho trabajo por sí solo, recibe la solicitud del servicio desde la capa de UI[14] y coordina su realización, normalmente delegando la actividad a otros objetos.

[14] UI (User Interface, Interfaz de Usuario)

Este patrón es evidente en los archivos JavaScript, pues es aquí donde se capturan los eventos generados por el usuario y se envían a la capa de servicio que se encarga de procesar los datos y devolver una respuesta apropiada.

Un ejemplo de ello es la operación que recarga un listado cuando el usuario utiliza el campo de búsqueda:

```
reloadGrid: function () {
    var buscar = $('#inputBuscar').val();
    $('#tableGrid').setGridParam({postData: {inputBuscar: buscar}}).trigger("reloadGrid");
},
```

Patrón bajo acoplamiento: El acoplamiento es una medida de la fuerza con que un elemento está conectado, necesita o tiene conocimiento de otros elementos. Un elemento con bajo (o débil) acoplamiento no depende de muchos otros elementos. El objetivo de este patrón es asignar una responsabilidad de manera que el acoplamiento permanezca bajo. Normalmente se considera que existe acoplamiento entre 2 clases si una de ellas extiende a otra (o implementa en caso de ser interfaces), contiene un atributo o método que hace referencia a una instancia de la otra o invoca sus servicios. El bajo acoplamiento es un sinónimo de mínima dependencia y facilita la reutilización, mantenimiento y comprensión de los componentes.

Para mantener el bajo acoplamiento se evita el abuso de la herencia, y en vez de ello se emplea la agregación de objetos de las clases que se necesitan. En las clases de la capa modelo puede observarse este comportamiento, ya que todas instancian a *busqueda_mdl* para utilizar sus métodos.

Patrón alta cohesión: La cohesión o cohesión funcional es una medida del grado de focalización de las responsabilidades de un elemento y de la fuerza con la que se relacionan. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Una clase con baja cohesión representa un "grano grueso" de abstracción, tiene muchas tareas no relacionadas, o hace demasiado trabajo lo que quiere decir que se le ha asignado responsabilidades que deberían haberse delegado en otros objetos.

Para mantener la alta cohesión entre los componentes se crearon nuevas clases las cuales representan abstracciones que reducen la carga de otras. Por ejemplo, se crearon las clases area_lib, vw_area_mdl, local_lib, vw_local_mdl que manipulan los datos de locales y áreas para despojar de esta tarea a otros objetos.

2.8.4 Patrones de diseño GoF

Los patrones de diseño GoF se clasifican según su propósito en 3 grupos, patrones creacionales, estructurales y de comportamiento.

Los patrones de diseño creacionales resumen el proceso de instanciación. Ayudan a que un sistema sea independiente de cómo se crean, se componen y se representan sus objetos. Un patrón de creación de clase usa la herencia para variar la clase que se instancia, mientras que un patrón de creación de objeto delegará la creación de instancias en otro objeto [17].

Un ejemplo de este tipo de patrones es *Instancia única (Singleton)*: patrón de creación que garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia [17].

En la siguiente ilustración se muestra la clase caracteristica_lib.php que no hereda de otra clase de CodeIgniter y por tanto no puede usar los métodos *load* o *helper.* Para poder usar estos métodos debe hacer referencia a un objeto singleton que obtiene a través del método *get_instance()* en la primera línea del constructor.

```
public function __construct() {
    $this->_ci = & get_instance();
    $this->_ci->load->model('tb_ndispositivo_mdl');
    $this->_ci->load->model('tb_natributo_mdl');
    $this->_ci->load->model('tb_ncaracteristica_mdl');
}
```

Figura 4: Ejemplo de uso del patrón instancia única.

Mediador (Mediator): Su intención es definir un objeto que encapsula cómo interactúa un conjunto de objetos. Mediador promueve el bajo acoplamiento evitando que los objetos se refieran entre sí explícitamente, y permite variar su interacción de forma independiente [17]. Este patrón se ve reflejado en la función que ocupan las librerías como mediadoras de la comunicación entre las clases controladoras y las modelos o de acceso a datos.

```
public function registrar($all_post) {
    $datos = $this->setDatos($all_post);
    $datos['id_atributo'] = $all_post['id_atributo'];
    $datos['fecha_creado'] = date('Y-m-d h:i:s', time());
    return $this->_ci->tb_ncaracteristica_mdl->registrar($datos);
}
```

Figura 5: Ejemplo de uso del patrón mediator.

En figura se muestra uno de los métodos de la clase caracteristica_lib, quien recibe de la controladora los datos del formulario para registrar una característica, los transforma y envía a la clase modelo para ser insertados en la base de datos, devolviendo a la librería y esta a su vez a la controladora el resultado de la operación.

2.8.5 Patrones de diseño de base de datos

Llave subrogada (o sustituta):

Una llave subrogada es una clave que no tiene un significado contextual o de negocio y por lo tanto generalmente no son mostradas al usuario de la aplicación. Es creada artificialmente y su objetivo es proporcionar una forma única de identificar un registro cuando el esquema no proporciona una clave natural, o esta es compuesta lo que hace que se requieran más recursos en el procesamiento.

La forma más común de llave subrogada es una secuencia ascendente de números enteros o contadores por ejemplo (1,2,3,...), otros ejemplos incluyen la fecha y hora actual del sistema o una cadena alfanumérica generada aleatoriamente. Al ser producidas por el propio sistema no se crean o almacenan claves duplicadas, además, para llaves secuenciales, aleatorias o fechas no existe un límite práctico de combinaciones únicas. Como carecen de significado para el negocio, su valor no necesita ser cambiado en el tiempo lo que contribuye a la integridad referencial del esquema. Aplican reglas uniformes a cada registro, esto permite que pueda ser escrito código que automatice algunas tareas de forma independiente a la tabla.

Las claves sustitutas tienden a ser un tipo de datos compacto, como un entero de cuatro bytes, ello optimiza el espacio y resulta en un proceso de comparación más eficiente. Además, una distribución de claves no redundante hace que el índice *b-tree*[15] resultante esté completamente equilibrado. Por último el

[15] Un árbol equilibrado (*b-tree* o *balanced tree*) es uno en el que la cantidad de datos en el lado izquierdo y derecho de cada división se mantiene uniforme, de modo que la cantidad de niveles que tiene que descender para alcanzar

proceso de unión entre tablas de la base de datos es más rápido pues las tablas de clave única son menos costosas de unir (menos columnas para comparar) que las de claves compuestas.

2.8.6 Modelo de datos

Un modelo de datos es una colección de conceptos y reglas que permite comunicar la estructura y comportamiento del sistema, visualizar y controlar su arquitectura a través de entidades, atributos y las relaciones entre ellos. A continuación se muestra el diagrama entidad relación (Figura 6) que representa el modelo de datos de la propuesta de solución.

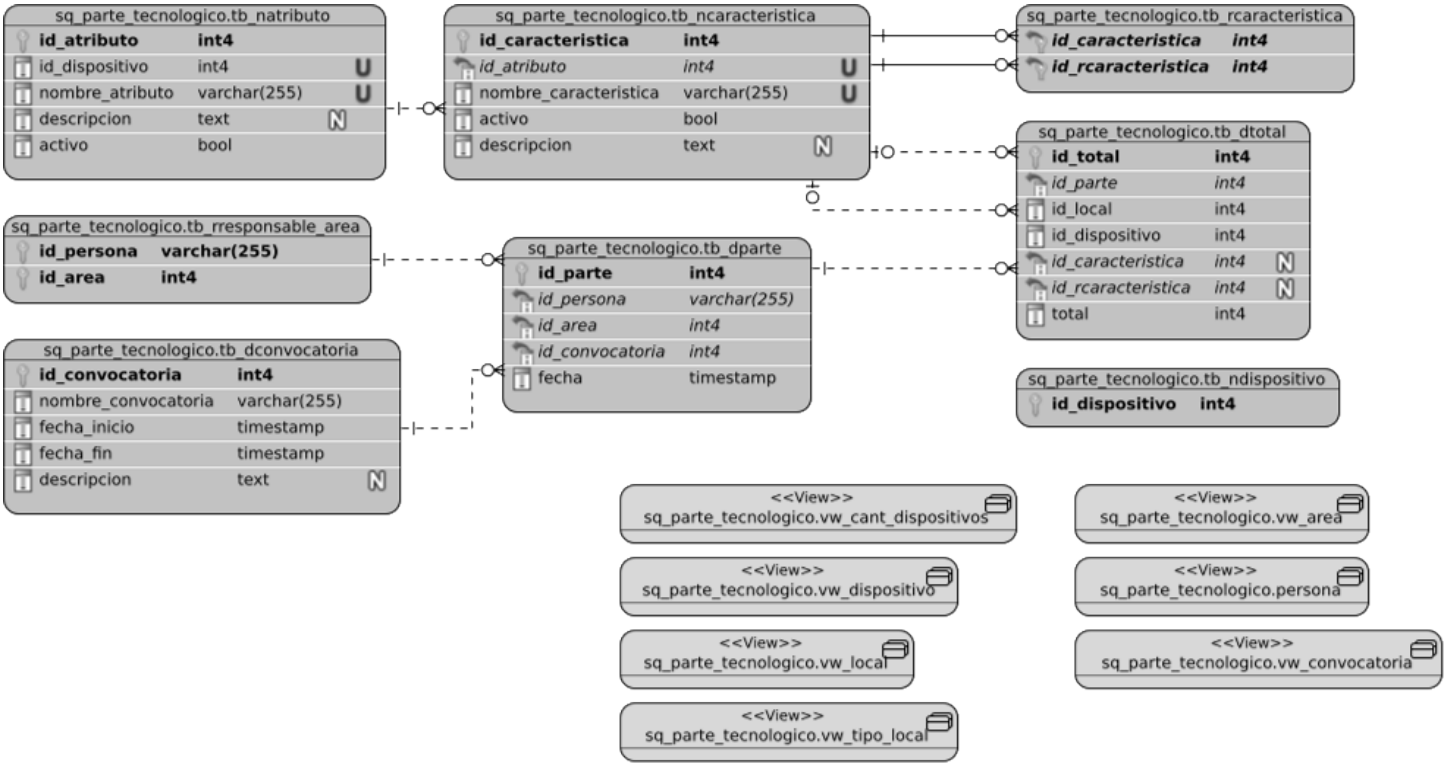


Figura 6: Diagrama Entidad-Relación del módulo Gestión de Partes tecnológicos.

El modelo físico de datos está conformado por 8 tablas (nombradas con prefijo tb_) para representar todos los datos persistentes que permiten la creación de nuevos partes y su consolidación en informes. Las vistas (nombradas con prefijo vw_) definidas en el modelo permiten la consulta de datos que provienen de

cualquier fila individual es aproximadamente igual. El árbol balanceado se puede usar para encontrar un solo valor o para escanear un rango, buscando valores clave que sean mayores que, menores que, y / o iguales a algún valor.

la base de datos *BD_sgu_nucleo* en los esquemas *activos*, *inmuebles* y *personal*. Por ejemplo, la vista 'vw_dispositivo' obtiene la llave primaria, y nombre de los dispositivos, ocultando la complejidad de una consulta que debe vincular las tablas *tb_dactivo* y *tb_nsubclasificacion* de un esquema que pertenece a otra base de datos.

En el esquema, el concepto de parte tecnológico se encuentra distribuido en las tablas *tb_dparte* y *tb_dtotal*, el primero hace referencia a la persona, estructura y convocatoria a la que responde el parte, la segunda almacena información acerca de la cantidad de dispositivos en los locales, teniendo en cuenta el tipo de dispositivo, sus características o estado técnico. La tabla *tb_ndispositivo* se utiliza para almacenar la llave primaria de los medios tecnológicos activos (aquellos que deben aparecer en el parte).

2.8.7 Diagrama de despliegue

El diagrama de despliegue es una forma de definir y documentar el entorno objetivo, se utiliza para mostrar la estructura física del sistema, incluyendo las relaciones entre el hardware y el software que se despliega, así como el *middleware* usado para conectar los diferentes componentes en el sistema. En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta:

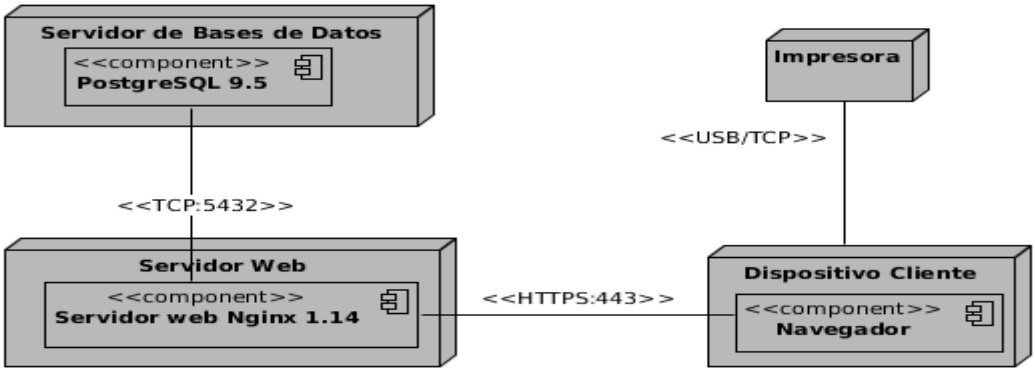


Figura 7: Diagrama de despliegue

En el diagrama, los componentes de hardware (servidor web, servidor de bases de datos, dispositivo cliente e impresora) se presentan como nodos, con los componentes de software que se ejecutan dentro de ellos presentados como artefactos.

Servidor de Bases de Datos: Dispositivo físico que ejecuta al sistema gestor de base de datos PostgreSQL que almacena la información persistente del Sistema de Gestión Universitaria.

Servidor Web: Dispositivo de hardware que alberga al servidor Nginx, el cual se encarga de ejecutar la aplicación web.

Dispositivo cliente: Representa cualquier dispositivo que permita al usuario acceder a la aplicación web a través de un navegador.

Impresora: La impresora solo es necesaria en el dispositivo cliente si el responsable de área pretende imprimir un reporte que haya generado o el director de tecnología quiera imprimir un informe consolidado.

<<**HTTPS**>>[16]: Protocolo empleado para establecer a través del puerto 443 la conexión segura entre el dispositivo de acceso cliente y el servidor de web Nginx.

<<**TCP**>>[17]: Protocolo empleado para establecer la conexión entre el servidor de web Nginx y el sistema gestor de base de datos de PostgreSQL usando el puerto 5432. Además, es usado para establecer la comunicación entre una PC cliente y una impresora.

2.9 Conclusiones del capítulo

Luego de describir las características que debe cumplir el Módulo Partes Tecnológicos y de realizar el análisis y diseño correspondientes, se concluye que los requisitos funcionales y no funcionales identificados a partir del proceso de obtención de los requisitos y los artefactos generados, constituyeron una guía fundamental para la construcción de la propuesta de solución.

Con la realización del modelo físico de datos se hizo posible materializar la visión de la base de datos a utilizar. La utilización de la arquitectura Modelo-Vista-Controlador y el uso de los patrones de diseño y de base de datos propuestos, garantizará una mayor organización, reutilización de funciones y código más legible.

Por otro lado la realización del modelo de despliegue contribuyó a ilustrar la comunicación entre los componentes del módulo y precisar la distribución de estos en los distintos servidores del sistema.

[16] HTTPS (*Hypertext Transfer Protocol Secure*, Protocolo de Transferencia Segura de Hipertexto)

[17] TCP (*Transmission Control Protocol*, Protocolo de Control de Transmisión)

Capítulo 3: Implementación y validación de la propuesta de solución.

3.1 Introducción

En el presente capítulo se describe la implementación del software, fase donde finalmente se materializa el producto y cumple con los requisitos obtenidos al inicio de la investigación. Se definen los estándares de codificación que debe cumplir el equipo de desarrollo, así como los métodos y técnicas para la realización de las pruebas, con el propósito de validar la solución. El proceso de pruebas está dirigido a componentes del software, con el objetivo de medir el grado en que se cumplen los requisitos exigidos por el cliente y si posee la calidad requerida.

3.2 Estándares de codificación

Los estándares de codificación son reglas de codificación que permiten tener una programación homogénea, comprendiendo todos los aspectos de la generalización del código, pues la aplicación debe parecer estar implementada como si un único programador escribiera el código. El uso de estos permite conservar el código fuente fácil de entender y mantener, además, de mejorar la forma en la que se programa.

Para el desarrollo del componente se utilizaron los estándares de codificación establecidos por la DIN con el propósito de estandarizar las nomenclaturas en la implementación del mismo y obtener un producto estable.

Indentación, llaves de apertura y cierre, y tamaño de líneas:

Se utilizan las configuraciones por defecto del IDE de desarrollo Netbeans para la indentación[18] del código, ajuste de tabuladores, nuevas líneas y generación automática de llaves las cuales constituyen estándares internacionales. Al ajustar automáticamente el código este puede cambiar de apariencia si se utiliza otra configuración, lo que favorece la flexibilidad de los estándares que se escojan. Otras pautas son la longitud de las líneas de código es aproximadamente de 75-80 caracteres para mantener la legibilidad del código.

[18] **Indentación**: Es un anglicismo de la palabra inglesa *indentation*, de uso común en la informática. No es un término reconocido por la Real Academia Española. En los lenguajes de programación es un tipo de notación secundaria utilizado para mejorar la legibilidad del código fuente por parte de los programadores.

```
$('#formulario').form({
    rules: {
        inputNombre: {
            required: true,
            alphanumeric: true,
            wordlen: true,
            maxlength: 50
        },
        textareaDescripcion: {
            wordlen: true
        }
    },
```

Figura 8: Ejemplo de Indentación

3.2.1 Convención de nomenclatura

Los nombres de variables, métodos y tablas de la base de datos claros y sugerentes, que no sean tan largos y descriptivos que dificulten su lectura, ni tan cortos y genéricos que sea difícil interpretarlos. Las variables y métodos se rigen según la nomenclatura *Lower Camel Case*[19], mientras que los objetos de la base de datos utilizan *Snake Case*[20] con todas las letras en minúscula. Para nombrar las clases se utiliza *Snake Case* comenzando con mayúsculas.

El nombre de los ficheros debe ser siempre en minúscula y en caso de nombres compuestos se usa el guión bajo, seguido del sufijo definido para cada tipo de fichero.

- **Vistas**: intuitivo y relacionado con el formulario y/o vista que representa. Sufijo "_view". Ejemplo: registrar_archivo_view
- **Modelos**: mismo nombre de la tabla de la base de datos para la cual se crea. Sufijo "_mdl". Ejemplo: tb_darchivo_mdl
- **Librerías**: mismo nombre de la clase que representa. Sufijo "_lib". Ejemplo: archivo_lib
- **Controladoras**: mismo nombre de la clase que representa. Ejemplo: archivo

[19] *Lower Camel Case*: Una convención de nomenclatura de la familia *CamelCase* en la que se juntan varias palabras, donde la primera letra de la palabra entera está en minúscula, pero las primeras letras subsiguientes están en mayúscula. Contrasta esto con *UpperCamelCase* (también conocido como *PascalCase*), donde la primera letra de cada palabra está en mayúscula.

[20] *Snake case* (o *snake_case*) es la práctica de escribir palabras o frases compuestas en las que los elementos están separados con un carácter de subrayado (_) y sin espacios, con la letra inicial de cada elemento generalmente en minúscula dentro del compuesto y la primera letra ya sea superior o minúscula.

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplir con el siguiente bloque al principio de cada clase:

```
/**
 * @category Modelo
 * @package Archivo
 * @author José Carlos Rodríguez Texidor jctexidor@estudiantes.uci.cu
 */
```

Figura 9: Documentación de una clase.

3.3 Estrategia de pruebas de software

Según Pressman una **estrategia de pruebas** envuelve las actividades de revisión y ejecución de las pruebas de software, cuyo objetivo es descubrir errores en el contenido, función, usabilidad, navegavilidad, rendimiento, capacidad y seguridad de la aplicación web.

Luego del desarrollo de la propuesta de solución, se hace necesario verificar y validar el sistema implementado a través de una estrategia de pruebas que permita comprobar el cumplimiento de las especificaciones del diseño y de la codificación e identificar los posibles errores cometidos. Las pruebas de software intentan demostrar que un programa hace lo que se intenta que haga, así como descubrir defectos en el programa antes de usarlo. Al probar el software, se ejecuta un programa con datos artificiales. Hay que verificar los resultados de la prueba que se opera para buscar errores, anomalías o información de atributos no funcionales del programa [18].

Se diseñó una estrategia de prueba para el módulo Partes Tecnológicos de SGU, en función de garantizar su seguridad, verificar la conformidad de los requisitos establecidos y validar que el software cumpla con lo que el usuario espera.

La **prueba de software** es una actividad donde un sistema o uno de sus componentes se ejecuta en circunstancias previamente descritas, registrándose los resultados obtenidos. El principal objetivo del diseño de casos de prueba es obtener un conjunto de pruebas que tengan la mayor probabilidad de descubrir los defectos del software. Las pruebas se ejecutan en la aplicación y mediante técnicas experimentales se trata de descubrir que errores presenta, para determinar el nivel de calidad y comprobar el grado de cumplimiento respecto de las especificaciones iniciales del sistema [16].

En la investigación se utilizan las pruebas recomendadas por Pressman en su libro “Ingeniería de software: Un enfoque práctico”, donde se recomienda que el sistema sea probado desde el más bajo nivel del software hasta la más alto en forma de espiral o sea pruebas unitarias, de integración, validación y sistema, en ese orden.

Para comprobar que la solución implementada es válida y que cumple con los requisitos acordados con el cliente, se propone la siguiente estrategia de pruebas:

Tabla 6: Estrategia de pruebas (Elaboración propia).

Tipo de prueba	Método	Técnica
Pruebas unitarias	Caja blanca	Prueba del camino básico.
Pruebas de integración	Cala blanca	Enfoque incremental.
Pruebas del sistema • Prueba de carga • Prueba de estrés	Caja negra	Automáticas
Prueba de validación	Caja negra	Partición equivalente.
Prueba de aceptación	Caja negra	Alfa y beta.

3.3.1 Pruebas unitarias

Las pruebas unitarias están orientadas a la verificación de la unidad más pequeña del diseño del software; los componentes de software o módulos. Las pruebas unitarias se enfocan en el procesamiento lógico interno y las estructuras de datos dentro de los límites del componente. Este tipo de pruebas se aplica al código fuente por lo que constituye un ejemplo de prueba de caja blanca.

Pruebas de caja blanca: se centran en la estructura de control del programa. Mediante los métodos de prueba de caja blanca, el ingeniero del software puede obtener casos de prueba que garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada componente, ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa, ejecuten todos los bucles en sus límites y con sus límites operacionales y ejerciten las estructuras internas de datos para asegurar su validez [16].

Existen 2 aproximaciones a la hora de realizar pruebas unitarias, manual y automatizada. En el enfoque manual las pruebas unitarias son realizadas por personas, mientras que, en el automatizado son ejecutadas de forma automática por un software. Es recomendable usar la prueba unitaria manual en vez

de la automatizada, cuando la verdadera intervención del usuario es necesaria, la automatización es demasiado costosa o la prueba se ejecuta un pequeño número de veces.

Para ejecutar las pruebas unitarias al sistema se emplearon ambos enfoques. Las pruebas automatizadas fueron realizadas utilizando la biblioteca unit_test que forma parte del marco de trabajo CodeIgniter la librería permite determinar con certeza si un código específico produce el tipo de dato y resultado esperado.

Como el resto de las clases de CodeIgniter esta biblioteca debe inicializarse en el controlador de la siguiente manera.

- *$this→load→library('unit_test');*

Una vez que ha sido cargada para ejecutar la prueba se utiliza la línea de código:

- $this->unit->run(*código*, *resultado esperado*, '*nombre de prueba*');

Donde *código* es el segmento de código que se desea probar, *resultado esperado* es lo que debe devolver la evaluación del código que puede ser un tipo de dato o un valor literal y *nombre de la prueba* es el nombre que se le puede dar a la prueba.

Tabla 7: Prueba unitaria realizada a la funcionalidad crear convocatoria (Elaboración propia).

Prueba estructural de caja blanca	Código de caso de prueba
	Modificar
Probador	José Carlos Rodríguez Texidor
Tipo de dato esperado	Boolean
Código al que se aplica	

```
            public function setDatos($all_post) {
                $datos['nombre_convocatoria'] = $all_post['inputNombre'];
                $datos['descripcion'] = $all_post['textareaDescripcion'];
                $datos['fecha_inicio'] = $all_post['calendarFechaInicio'];
                $datos['fecha_fin'] = $all_post['calendarFechaFin'];
                return $datos;
            }



        public function modificar($all_post) {
            $ids['id_convocatoria'] = $all_post['id'];
            $datos = $this->setDatos($all_post);
            return $this->_ci->tb_dconvocatoria_mdl->modificar($ids, $datos);
        }
```

Caso de prueba para los caminos básicos	
Tipo de dato esperado	Boolean
Función de evaluación	

```
public function evalModificar() {
    $all_post['id'] = 1;
    $all_post['inputNombre'] = 'nombre_convocatoria';
    $all_post['textareaDescripcion'] = 'descripcion';
    $all_post['calendarFechaInicio'] = '23/4/2019';
    $all_post['calendarFechaFin'] = '25/4/2019';
    echo $this->unit->run($this->convocatoria_lib->modificar($all_post), TRUE);
}
```

Resultados de la prueba

Test Name	Undefined Test Name
Test Datatype	Boolean
Expected Datatype	Boolean
Result	Passed
File Name	/home/jctexidor/www/akademos/proyecto_sgu/base/application/tecnologia/parte_tecnologico/controllers/convocatoria.php
Line Number	33

Evaluación del caso de prueba	Satisfactoria

Las pruebas unitarias desarrolladas arrojaron resultados positivos y todas las inconformidades detectadas durante este ejercicio fueron corregidas satisfactoriamente antes de que el producto fuese entregado al cliente.

3.3.2 Pruebas de integración

La prueba de integración es una técnica sistemática para construir la estructura del programa mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interacción. El objetivo es coger los componentes probados anteriormente y construir una estructura de programa que esté de acuerdo con lo que dicta el diseño [17].

Este tipo de pruebas tiene 2 enfoques: En el enfoque Big Bang se combinan todos los componentes y el programa es probado en todo su conjunto. El resultado puede ser caótico con un gran conjunto de fallos y la consiguiente dificultad para identificar el componente (o componentes) que los provocó. Por el contrario, se puede aplicar la integración incremental en la que el programa se construye y prueba en pequeños segmentos en los que los errores son más fáciles de aislar y de corregir, es más probable que se puedan probar completamente las interfaces y se puede aplicar un enfoque de prueba sistemática [16].

Tabla 8: Caso de prueba de integración del componente Partes Tecnológicos (Elaboración propia).

Caso de prueba de integración del componente Partes Tecnológicos**.**

Sistema/componente al que se integra:	Activos.
Condiciones de ejecución	Exista conexión con la base de datos “núcleo” y el esquema “activos”.
Descripción de la prueba	Se realiza la petición a nivel de base de datos mediante la extensión plproxy para obtener los tipos de dispositivos, que se gestionan en el componente Partes Tecnológicos.
Resultado esperado	El componente Partes Tecnológicos tiene acceso a los datos del esquema activos que permiten a los usuarios tienen acceso a las funcionalidades de acuerdo al rol y sus responsabilidades.
Evaluación	Prueba satisfactoria.

Con la aplicación de pruebas de integración continua se logró la correcta integración de todos los componentes del módulo y el Sistema de Gestión Universitaria. Lo cual permitió asegurar una adecuada ejecución de las funcionalidades que requerían del acceso a datos externos y comunicación con recursos compartidos.

3.3.3 Pruebas de validación

Las pruebas funcionales tienen por objetivo probar que los sistemas desarrollados cumplen con los requisitos funcionales del software. Los probadores o analistas de pruebas no enfocan su atención en cómo se generan las respuestas del sistema, sino en el funcionamiento de la interfaz del sistema [16].

Estas pruebas son consideradas **pruebas de caja negra** las cuales son diseñadas para validar los requisitos funcionales sin fijarse en el funcionamiento interno de un programa. La prueba de caja negra permite al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca; se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores que los métodos de caja blanca no pueden.

Intenta encontrar errores en las siguientes categorías: (1) funciones incorrectas o ausentes, (2) errores de interfaz, (3) errores en estructuras de datos o en accesos a bases de datos externas, (4) errores de comportamiento o rendimiento y (5) errores de inicialización y terminación. Las técnicas de prueba de caja negra se centran en el ámbito de información de un programa, de forma que se proporcione una cobertura completa de la prueba [16].

Partición equivalente es una técnica de prueba de caja negra que divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software. Se dirige a la definición de casos de

prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Un caso de prueba ideal descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico [16].

A continuación se describen 1 de los casos de prueba empleados en la estrategia de pruebas para validar la aplicación. Se exponen los escenarios para los datos correctos e incorrectos.

Tabla 9: Caso de prueba insertar convocatoria (Elaboración propia).

Escenario	**Descripción**	**Nombre**	**Fecha inicio**	**Fecha fin**	**Respuesta del sistema**	**Flujo central**
EC 1.1 Insertar datos correctos	Mediante este escenario se inserta en el sistema una convocatoria.	V	V	V	El sistema crea la nueva convocatoria y muestra el mensaje: “El elemento ha sido creado satisfactoriamente.	El usuario una vez autenticado en el Sistema de Gestión Universitaria selecciona el Subsistema de Tecnologías y luego el módulo Partes Tecnológicos. -El sistema muestra las opciones de menú. -El usuario selecciona en la agrupación funcional "Configuración" la funcionalidad "Convocatorias". -El sistema muestra las convocatorias registradas. -El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos correctamente y presiona el botón Aceptar.
		Convocatoria1	15/3/2019	29/4/2019		
		Vacío				
EC 1.2 Insertar datos incorrectos	Mediante este escenario se introducen datos incorrectos.	I	NA	NA	El sistema muestra en rojo el mensaje encima del componente: “Entre solo letras, números, apóstrofe, paréntesis guion y espacio o guion bajo entre palabras.”	El usuario una vez autenticado en el Sistema de Gestión Universitaria selecciona el Subsistema de Tecnologías y luego el módulo Partes Tecnológicos. -El sistema muestra las opciones de menú. -El usuario selecciona en la agrupación funcional "Configuración" la funcionalidad "Convocatorias". -El sistema muestra las convocatorias registradas. -El usuario selecciona en el área de iconos flotantes la opción: Crear, llena los datos de forma incorrecta y presiona el botón Aceptar.
		"Convocatoria"				
		I	NA	NA	El sistema	

		C			muestra en color rojo encima del componente el mensaje: “Entre al menos 2 caracteres.”	
		I	NA	NA	El sistema no permite la entrada de más caracteres.	
		Escribir más de 250 caracteres				
		NA	I	NA	El sistema muestra en rojo encima del componente: “Solo admite espacio entre palabras.”	
			Escribir una descripción que termine con espacio.			
		I	I	NA	El sistema muestra en rojo encima del componente: Ha excedido el número de caracteres permitidos para una palabra.	
		El usuario entra una palabra con más de 30 caracteres	El usuario entra una palabra con más de 30 caracteres.			

Se llevaron a cabo tres iteraciones para detectar no conformidades, las cuales arrojaron los resultados que se muestran en la siguiente tabla.

Tabla 10: Iteraciones de las pruebas de validación (Elaboración propia).

Iteración	Cantidad de no conformidades	Asociadas a

Primera	10	Errores de validación, entradas y salidas incorrectas. Errores de inserción en base de datos. Errores de excepción con poca información o incorrecta.
Segunda	3	Errores de validación
Tercera	0	-

Estos datos demuestran que se han detectado y corregido los errores detectados en el módulo, asegurando al cliente la calidad del mismo, sin embargo, esto no garantiza que en futuras revisiones no se encuentren nuevos errores o mejoras en la implementación. Este debido a que se pueden introducir nuevos errores a medida que se corrigen los ya detectados o la tecnología que se usó en el desarrollo quede obsoleta.

3.3.4 Pruebas de sistema

Las pruebas de sistema están constituidas por una serie de pruebas diferentes cuyo propósito primordial es ejercitar profundamente el sistema basado en computadora. Aunque cada prueba tiene un propósito diferente, todas trabajan para verificar que se han integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas [16].

Entre las pruebas de sistema que se realizan están las pruebas de integridad de datos, rendimiento, resistencia y seguridad.

Las pruebas de integridad de datos y base de datos tienen como objetivo asegurar que los métodos de acceso y procesos funcionan adecuadamente y sin ocasionar corrupción de datos. Una forma sencilla de aplicar esta prueba es revisar de forma manual que al insertar o manipular una fila en base de datos la misma contenga valores acorde a su dominio y que correspondan a la acción realizada sin alterar otra fila o conjunto de datos.

Esta prueba fue aplicada para cada uno de los requisitos funcionales una vez que el sistema se encontraba integrado sin obtener no conformidades. El principal motivo de la ausencia de no conformidades al realizar esta prueba es que durante la integración paulatina del módulo se fueron realizando pruebas similares.

Las **pruebas de resistencia** están diseñadas para enfrentar a los programas con situaciones anormales. La prueba de resistencia ejecuta un sistema de forma que demande recursos en cantidad, frecuencia o volúmenes anormales [16].

Las **pruebas de rendimiento** están diseñadas para probar el rendimiento del software en tiempo de ejecución dentro del contexto de un sistema integrado. La prueba de rendimiento se da durante todos los pasos del proceso de la prueba [16].

Las **pruebas de seguridad** intentan verificar que los mecanismos de protección que se construyen en un sistema lo protegerán realmente de cualquier irrupción inapropiada. Buscan medir la confidencialidad, integridad y disponibilidad de los datos. Se diseñan para detectar las vulnerabilidades del entorno en el sistema, las comunicaciones de red que ocurren conforme los datos pasan del Paquete de Servicios al sistema y viceversa, y el entorno en el Paquete de Servicios. Cada uno de estos dominios puede atacarse, y es tarea del examinador de seguridad descubrir las debilidades que puedan explotar quienes tengan intención de hacerlo [16].

Se realizaron las pruebas para comprobar que el tiempo de respuesta del componente no exceda a los 5 segundos de forma general, así como el tiempo de respuesta por cada transacción promedio sea de un mínimo de 1,5 segundos y un máximo de 5 segundos. A continuación se muestran las pruebas de desempeño realizadas en el navegador web Mozilla Firefox.

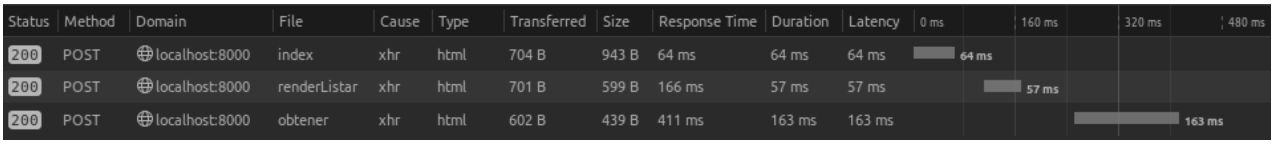


Figura 10: Prueba de rendimiento al requisito listar dispositivos.

3.3.5 Pruebas de aceptación

Cuando se construye software para un cliente, se llevan a cabo una serie de pruebas de aceptación para permitir que el cliente valide todos los requisitos. Son ejecutadas antes de que la aplicación sea instalada dentro de un ambiente de producción. La prueba de aceptación es conducida a determinar como el sistema satisface sus criterios de aceptación validando los requisitos que han sido levantados para el desarrollo, incluyendo la documentación y procesos de negocio [16].

La **prueba alfa** se lleva a cabo por un grupo representativo de usuarios finales en el sitio de desarrollo. El software es usado de forma natural con el desarrollador como observador de los usuarios y registrando los errores y los problemas de uso. Las pruebas alfa se llevan a cabo en un entorno controlado [16].

La **prueba beta** se lleva a cabo en los sitios de trabajo de los usuarios finales. A diferencia de las pruebas alfa, el desarrollador generalmente no está presente. Debido a esto la prueba beta es una aplicación “viva” del software en un ambiente no controlado por el desarrollador. El cliente registra todos los problemas que encuentra durante la prueba beta e informa a intervalos regulares al desarrollador. Como resultado de los problemas informados durante la prueba beta, el desarrollador del software lleva a cabo modificaciones y prepara una versión del producto de software para toda la base de clientes [16].

3.4 Validación de la hipótesis (Criterio de expertos)

Para realizar la validación de la investigación se aplicó el método Criterio de expertos a través del cumplimiento de los pasos siguientes:

- Identificación de los posibles expertos.
- Determinación del coeficiente de competencia de los candidatos a expertos.
- Selección de los expertos.
- Realización del cuestionario a los expertos.
- Aplicación del escalamiento de *Likert*.

Identificación de los posibles expertos: Se tuvieron en cuenta, la experiencia profesional en el tema, de modo que estuvieran en capacidad de ofrecer valoraciones y hacer recomendaciones pertinentes, en relación con los aspectos que le serían consultados.

Determinación del coeficiente de competencia de los candidatos a expertos: Una vez identificados los posibles expertos, se les realizó una encuesta para valorar el grado de conocimiento y el grado de influencia que cada una de las fuentes ha tenido en ese conocimiento. El procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos, puede ser consultado en el Anexo 2: .

Selección de expertos: Fueron seleccionados expertos con nivel alto y medio del coeficiente de competencia, dado que el coeficiente de competencia promedio del grupo de expertos no era inferior a 0.7 manteniéndose un nivel alto en el conjunto. En la siguiente tabla se muestran los expertos seleccionados.

Tabla 11: Selección de expertos para la validación de la hipótesis (Elaboración propia).

Experto	Área	Cargo	Coeficiente
Alexander Rodríguez Mompié	Dirección de Informatización	Director	
Dianly Santiler Alvarez	Dirección de Informatización	Especialista "A" en Ciencias Informáticas	
Froilán Morales Medina	Dirección de Informatización	Especialista Superior	
José Alejandro Garcia Calderón	Dirección de Informatización	Jefe de Departamento	
Roberkis Terreno Galano	Dirección de Informatización	Profesor	
Yamila Mateu Romero	Dirección de Informatización	Profesor	
Lester Rodriguez Vallejo	Dirección de Gestión Tecnológica	Director	

Realización del cuestionario a los expertos: Una vez seleccionado los expertos se les realizó un cuestionario compuesto por dos (2) afirmaciones evaluativas que permiten conocer la opinión del experto. Los parámetros evaluativos empleados fueron: MA (muy de acuerdo), DA (de acuerdo), N (Ni de acuerdo ni en desacuerdo), ED (en desacuerdo), CD (completamente en desacuerdo). El cuestionario y las respuestas dadas por los expertos pueden ser consultados en los anexos. Los parámetros evaluativos se cuantificaron a través de la siguiente escala:

- Muy de acuerdo (5)
- De acuerdo (4)
- Ni de acuerdo ni en desacuerdo (3)
- En desacuerdo (2)
- Completamente en desacuerdo (1)

Aplicación del escalamiento de *Likert*: Para el procesamiento de los resultados se empleó un método que consiste en identificar la media de las diferentes categorías de la escala *Likert*. Dicho método se aplicó por separado a cada una de las afirmaciones. Luego se obtuvo la media de las valoraciones individuales para obtener una valoración integral del sistema.

El proceso realizado a través del escalamiento *Likert* evidencia que tanto los elementos teóricos de la investigación como la propuesta solución, tienen una alta valoración por parte de los expertos. Durante el proceso se constataron criterios favorables en el uso del módulo para mejorar la eficiencia en el proceso de registro de medios tecnológicos personales. Se obtuvo un resultado satisfactorio de un alto por ciento de aprobación por parte de los expertos.

3.5 Conclusiones del capítulo

En el presente capítulo se abordaron elementos de la implementación y prueba del módulo para la creación de partes e informes tecnológicos en la Universidad de las Ciencias Informáticas.

La identificación de buenas prácticas de codificación, estándares y nomenclatura de las tablas de la base de datos guiaron un desarrollo claro y fluido de las funcionalidades del módulo permitiendo una mejor legibilidad y comprensión del código, facilitando el mantenimiento del sistema de manera íntegra.

La ejecución de la estrategia de pruebas realizada permitió detectar y corregir oportunamente errores en el módulo asegurando la calidad, seguridad y usabilidad del software. La validación de hipótesis aplicando el método criterio de expertos arrojó un resultado aceptable con un alto por ciento de aprobación, lo que evidenció el correcto cumplimiento de los objetivos planteados en la presente investigación.

Conclusiones generales

Como resultado de la investigación se arribó a las siguientes conclusiones:

El estudio de los sistemas homólogos, permitió caracterizar las funcionalidades para el desarrollo de la solución, las cuales fueron ajustadas a las necesidades y características del negocio. Cumpliendo con la integridad y veracidad de los datos que se manejan, estando integrados con el sistema económico financiero de la universidad.

La aplicación de la metodología de desarrollo de software *Scrum*, facilitó la obtención y especificación de los requisitos, así como la elaboración de los artefactos propuestos y el diseño del módulo, permitiendo que la documentación del mismo perdure en el tiempo.

Con el desarrollo del módulo Partes Tecnológicos del SGU, se lograron implementar los 37 requisitos funcionales identificados, dando la posibilidad de gestionar los atributos que se van a recoger en el parte de cada medio en cada uno de los locales con los que cuentan las estructuras administrativas de la universidad.

Finalmente, se realizaron las pruebas de caja blanca, caja negra y sistema para garantizar que el módulo cumple con los requisitos establecidos y satisface las necesidades del cliente. Se aplicaron además todos los casos de pruebas para validar que el resultado que se muestra se corresponde con las descripciones de los requisitos.

Recomendaciones

Para garantizar el perfeccionamiento progresivo de la solución propuesta se proponen las siguientes recomendaciones que tributarán a un producto de mejor calidad. Las funcionalidades que se recomiendan son:

- Configurar las notificaciones y alertas para mantener informados a los usuarios de las acciones que ocurren en el sistema.

Bibliografía

[1] “ASSETS: Sistema de Gestión Integral.” [Online]. Available: http://www.assets.co.cu/fijos.asp. [Accessed: 03-Dec-2018].

[2] “IT Asset Management Software, ITAM, Asset Lifecycle Management - ManageEngine AssetExplorer.” [Online]. Available: https://www.manageengine.com/products/asset-explorer/index.html?utm_source=capterra&utm_medium=ppc&utm_campaign=AE&utm_source=capterra. [Accessed: 31-Jan-2019].

[3] “IT Asset Management Software (ITAM) - Start Free Trial Today,” *SysAid*. .

[4] “SIGE 3.0 | Universidad de las Ciencias Informáticas.” [Online]. Available: https://www.uci.cu/investigacion-y-desarrollo/productos/xedro/sige-30. [Accessed: 03-Dec-2018].

[5] “How has this term impacted your life?,” *BusinessDictionary.com*. [Online]. Available: http://www.businessdictionary.com/definition/methodology.html. [Accessed: 03-Dec-2018].

[6] “CodeIgniter at a Glance — CodeIgniter 3.1.10 documentation.” [Online]. Available: https://www.codeigniter.com/user_guide/overview/at_a_glance.html. [Accessed: 31-Jan-2019].

[7] “About - Git,” 16-Nov-2018. [Online]. Available: https://git-scm.com/about. [Accessed: 16-Nov-2018].

[8] “What is NGINX?,” *NGINX*. [Online]. Available: https://www.nginx.com/resources/glossary/nginx/. [Accessed: 03-Dec-2018].

[9] “Welcome to NGINX Wiki! | NGINX.” [Online]. Available: https://www.nginx.com/resources/wiki/. [Accessed: 03-Dec-2018].

[10] “Web Console,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools/Web_Console. [Accessed: 27-Mar-2019].

[11] “Network Monitor,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools/Network_Monitor. [Accessed: 27-Mar-2019].

[12] “Performance,” *MDN Web Docs*. [Online]. Available: https://developer.mozilla.org/en-US/docs/Tools/Performance. [Accessed: 27-Mar-2019].

[13] “Home - Pencil Project.” [Online]. Available: https://pencil.evolus.vn/. [Accessed: 24-Mar-2019].

[14] “Features - Pencil Project.” [Online]. Available: https://pencil.evolus.vn/Features.html. [Accessed: 24-Mar-2019].

[15] “What is conceptual model? definition and meaning,” *BusinessDictionary.com*. [Online]. Available: http://www.businessdictionary.com/definition/conceptual-model.html. [Accessed: 24-Mar-2019].

[16] Roger S. Pressman, *Software Engineering: A Practitioner'S Approach, 8th Edition*, 8th Edition. 2014.

[17] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education India, 1995.

[18] I. Sommerville, *Software engineering 9th Edition*, vol. 137035152. 2011.

[19] "Definición de recursos tecnológicos — Definicion.de," *Definición.de*, 15-Nov-2018. [Online]. Available: https://definicion.de/recursos-tecnologicos/. [Accessed: 15-Nov-2018].

[20] "Fishbowl Inventory Software - 2018 Reviews & Pricing," 15-Nov-2018. [Online]. Available: https://www.softwareadvice.com/inventory-management/fishbowl-inventory-manufacturing-profile/. [Accessed: 15-Nov-2018].

[21] "Small Business Inventory | Free Online Inventory System | Inventory Management," *Online Inventory System | Small Business Inventory Software*, 15-Nov-2018. [Online]. Available: http://www.thecanvus.com/. [Accessed: 15-Nov-2018].

[22] "JavaScript basics - Learn web development | MDN," 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics. [Accessed: 26-Oct-2018].

[23] "How CSS works," *MDN Web Docs*, 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS/How_CSS_works. [Accessed: 26-Oct-2018].

[24] "What is JavaScript?," *MDN Web Docs*, 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Accessed: 26-Oct-2018].

[25] "Introduction to CSS," *MDN Web Docs*, 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS. [Accessed: 26-Oct-2018].

[26] "NetBeans IDE - Overview," 26-Oct-2018. [Online]. Available: https://netbeans.org/features/index.html. [Accessed: 26-Oct-2018].

[27] J. F.- js.foundation, "jQuery," 26-Oct-2018. .

[28] "Introduction — pgAdmin III 1.22.2 documentation," 26-Oct-2018. [Online]. Available: https://www.pgadmin.org/docs/pgadmin3/1.22/introduction.html. [Accessed: 26-Oct-2018].

[29] "Getting started with HTML - Learn web development | MDN," 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Getting_started. [Accessed: 26-Oct-2018].

[30] “DOM,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/DOM. [Accessed: 02-Nov-2018].

[31] “HTML,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/HTML. [Accessed: 02-Nov-2018].

[32] “Tag,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Tag. [Accessed: 02-Nov-2018].

[33] “Element,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Element. [Accessed: 02-Nov-2018].

[34] “CSS,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/CSS. [Accessed: 02-Nov-2018].

[35] “JavaScript,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/JavaScript. [Accessed: 02-Nov-2018].

[36] “Attribute,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Attribute. [Accessed: 02-Nov-2018].

[37] “Browser,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Browser. [Accessed: 02-Nov-2018].

[38] “XML,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/XML. [Accessed: 02-Nov-2018].

[39] “API,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/API. [Accessed: 02-Nov-2018].

[40] “SQL,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/SQL. [Accessed: 02-Nov-2018].

[41] “AJAX,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/AJAX. [Accessed: 02-Nov-2018].

[42] “Git,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/Git. [Accessed: 02-Nov-2018].

[43] “HTML5,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/HTML5. [Accessed: 02-Nov-2018].

[44] “JSON,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/JSON. [Accessed: 02-Nov-2018].

[45] “jQuery,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/jQuery. [Accessed: 02-Nov-2018].

[46] “PHP,” *MDN Web Docs*, 02-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Glossary/PHP. [Accessed: 02-Nov-2018].

[47] “Welcome! - The Apache HTTP Server Project,” 05-Nov-2018. [Online]. Available: https://httpd.apache.org/. [Accessed: 05-Nov-2018].

[48] “HTTPD - Apache2 Web Server,” 05-Nov-2018. [Online]. Available: https://help.ubuntu.com/lts/serverguide/httpd.html. [Accessed: 05-Nov-2018].

[49] “Projects,” *JS Foundation*, 07-Nov-2018. .

[50] “Home | Scrum Guides,” 07-Nov-2018. [Online]. Available: https://www.scrumguides.org/. [Accessed: 07-Nov-2018].

[51] “Scrum Guide | Scrum Guides,” 07-Nov-2018. [Online]. Available: https://scrumguides.org/scrum-guide.html. [Accessed: 07-Nov-2018].

[52] “Welcome to CodeIgniter — CodeIgniter 3.1.9 documentation,” 07-Nov-2018. [Online]. Available: https://www.codeigniter.com/user_guide/general/welcome.html. [Accessed: 07-Nov-2018].

[53] “PostgreSQL: Documentation: 11: 1. What is PostgreSQL?,” 07-Nov-2018. [Online]. Available: https://www.postgresql.org/docs/11/intro-whatis.html. [Accessed: 07-Nov-2018].

[54] R.- ASALE and R.- ASALE, “Diccionario de la lengua española - Edición del Tricentenario,” *Diccionario de la lengua española - Edición del Tricentenario*, 07-Nov-2018. [Online]. Available: http://dle.rae.es/. [Accessed: 07-Nov-2018].

[55] “Getting started with the Web - Learn web development | MDN,” 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web. [Accessed: 26-Oct-2018].

[56] “HTML basics - Learn web development | MDN,” 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/HTML_basics. [Accessed: 26-Oct-2018].

[57] “CSS basics - Learn web development | MDN,” 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/CSS_basics. [Accessed: 26-Oct-2018].

[58] “Introduction to HTML - Learn web development | MDN,” 26-Oct-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML. [Accessed: 26-Oct-2018].

[59] “Misión | Universidad de las Ciencias Informáticas,” 14-Nov-2018. [Online]. Available: http://www.uci.cu/universidad/mision. [Accessed: 14-Nov-2018].

[60] “Centros de Desarrollo | Universidad de las Ciencias Informáticas,” 14-Nov-2018. [Online]. Available: http://www.uci.cu/investigacion-y-desarrollo/centros-de-desarrollo. [Accessed: 14-Nov-2018].

[61] “La UCI de un vistazo | Universidad de las Ciencias Informáticas,” 14-Nov-2018. [Online]. Available: http://www.uci.cu/universidad/la-uci-de-un-vistazo. [Accessed: 14-Nov-2018].

[62] “Campus | Universidad de las Ciencias Informáticas,” 14-Nov-2018. [Online]. Available: https://www.uci.cu/universidad/campus. [Accessed: 14-Nov-2018].

[63] “Universidad de las Ciencias Informáticas,” *Universidad de las Ciencias Informáticas*, 14-Nov-2018. [Online]. Available: http://www.uci.cu. [Accessed: 14-Nov-2018].

[64] J. A. Stoner, R. E. Freeman, and D. R. Gilbert, “Management, Fith Edition,” 1992.

[65] “La informatización de la sociedad, una prioridad para Cuba,” *MINCOM | Ministerio de las Comunicaciones en Cuba*, 14-Nov-2018. [Online]. Available: http://www.mincom.gob.cu/es/node/1434. [Accessed: 14-Nov-2018].

[66] “Top 6 Free and Open Source Inventory Management Software Systems,” 15-Nov-2018. .

[67] “ABC Inventory Reviews and Pricing - 2018,” 15-Nov-2018. [Online]. Available: https://www.capterra.com/p/170315/ABC-Inventory/. [Accessed: 15-Nov-2018].

[68] A. Hillsberg, “Best Free Inventory Management Software Solutions to Consider in 2018,” *Financesonline.com*, 15-Nov-2018. [Online]. Available: https://financesonline.com/best-free-inventory-management-software-solutions-consider-2017/. [Accessed: 15-Nov-2018].

[69] “Bootstrap · The world’s most popular mobile-first and responsive front-end framework. 3.3.6 Documentation - BootstrapDocs,” 16-Nov-2018. [Online]. Available: https://bootstrapdocs.com/v3.3.6/docs/. [Accessed: 16-Nov-2018].

[70] “SQL Tutorial,” 16-Nov-2018. [Online]. Available: https://www.w3schools.com/sql/. [Accessed: 16-Nov-2018].

[71] “NetBeans IDE - Base IDE Features,” 16-Nov-2018. [Online]. Available: https://netbeans.org/features/ide/index.html. [Accessed: 16-Nov-2018].

[72] “Overview of new features in Apache HTTP Server 2.4 - Apache HTTP Server Version 2.4,” 16-Nov-2018. [Online]. Available: https://httpd.apache.org/docs/2.4/new_features_2_4.html. [Accessed: 16-Nov-2018].

[73] “What is a web server?,” *MDN Web Docs*, 16-Nov-2018. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server. [Accessed: 16-Nov-2018].

[74] J. Palacio and C. Ruata, “Scrum Manager,” *Gestión de proyectos*, 2010.

[75] “Técnicas de Administración de Inventarios para empresas comercializadoras en Cuba - GestioPolis.” [Online]. Available: https://www.gestiopolis.com/tecnicas-administracion-inventarios-empresas-comercializadoras-cuba/. [Accessed: 28-Nov-2018].

[76] “Asset Management [Capterra],” *SysAid*. .

[77] J. Chaffer and K. Swedberg, *Learning jQuery Fourth Edition*, Fourth Edition. Packt Publishing Ltd., 2013.

[78] G. Smith, *PostgreSQL 9.0: High Performance*. Packt Publishing Ltd, 2010.

Anexos

Anexo 1: Entrevista de levantamiento de requisitos

Entrevista al cliente para conocer la necesidad del desarrollo de la propuesta de solución y definir los requisitos funcionales y no funcionales.

Estimado especialista: Se necesita de su cooperación en una investigación para una tesis de pregrado.

Por ello, sería de gran ayuda que respondiera lo siguiente:

1. ¿Considera que en la universidad el proceso de creación de partes tecnológicos e informes consolidados se realiza de la mejor manera?
2. Respecto a la manera en que se lleva a cabo el proceso de creación de los partes tecnológicos en la UCI en la actualidad,
    I. ¿Cuáles son las áreas de responsabilidad que deben crear estos partes?
    II. ¿Debería una persona responsable de un área pertenecer a ella?
    III. ¿Existen características comunes predefinidas para todos los dispositivos?
3. ¿Cuáles son las causas que podrían estar influyendo en la descoordinación de ese proceso?
4. ¿Existe alguna herramienta en la Universidad para la creación de documentos?
5. ¿Cuáles son sus características?
6. A partir de estas características, ¿considera que la herramienta se ajusta a las necesidades actuales de la universidad?
7. ¿Cómo deberían firmarse digitalmente los documentos en la universidad?
8. ¿Qué otras características, considera que deba presentar el sistema, en cuanto a la usabilidad, seguridad, interfaz u otro aspecto que garantice su calidad?

Muchas gracias por su colaboración.

Anexo 2: Encuesta para determinar el coeficiente de competencias de los expertos

Compañero (a):

Usted ha sido seleccionado como posible experto para ser consultado respecto a temas relacionados a la creación de partes e informes consolidados sobre el estado de la tecnología en la Universidad de Ciencias Informáticas. Agradecemos sinceramente su valiosa cooperación.

Marque con una cruz (X) en la tabla siguiente el valor que se corresponde con el grado de conocimiento que usted posee sobre la sincronización entre el sistema de gestión universitaria y el sistema contable financiero, la creación de partes y los informes tecnológicos que la universidad debe entregar al Ministerio de educación Superior. (Escala ascendente).

0	1	2	3	4	5	6	7	8	9	10

2. Realice una autoevaluación del grado de influencia que cada una de las fuentes que le presentamos a continuación ha tenido en su conocimiento y criterio sobre el proceso de generación de los partes tecnológicos en la universidad. Marque con una cruz (X) según corresponda en A (alto), M (medio) o B (bajo).

No.	Fuente de argumento	Grado de influencia de cada una de las fuentes		
		A (alto)	M (medio)	B (bajo)
1	Estudios teóricos realizados por usted.			
2	Experiencia adquirida durante su vida profesional.			
3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.			
4	Conocimiento propio sobre el estado del tema de investigación.			
5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, entre otros.			
6	Intuición.			

Anexo 3: Procedimiento empleado para determinar el coeficiente de competencia de los candidatos a expertos.

Cálculo del coeficiente de competencia de los expertos que evaluaron el sistema informático desarrollado. El cálculo de dicho coeficiente se realiza de la forma siguiente:

Kcomp = 1⁄2 (Kc + Ka)

Donde:

Kcomp: Coeficiente de competencia.

Kc: Coeficiente de conocimiento o información que tiene el experto acerca del problema, calculado sobre la valoración del propio experto en una escala de 0 a 10 y multiplicado por 0,1.

Ka: Coeficiente de argumentación o fundamentación de los criterios del experto, obtenido como resultado de la suma de los puntos de acuerdo a la siguiente tabla patrón:

Tabla 12: Fuentes de argumentación del conocimiento de los expertos

No.	Fuente de argumento	Grado de influencia de cada una de las fuentes		
		A (alto)	M (medio)	B (bajo)
1	Estudios teóricos realizados por usted.	0,30	0,20	0,10
2	Experiencia adquirida durante su vida profesional.	0,50	0,37	0,30
3	Conocimiento de investigaciones y/o publicaciones nacionales e internacionales.	0,05	0,04	0,03
4	Conocimiento propio sobre el estado del tema de investigación.	0,05	0,04	0,03
5	Actualización en cursos de postgrado, diplomados, maestrías, doctorado, entre otros.	0,05	0,03	0,03
6	Intuición.	0,05	0,02	0,01
	Total	1,00	0,70	0,50

Se plantea entonces que:

La Competencia del experto es de Alta (A): Si Kcomp > 0,7.

La Competencia del experto es Media (M): Si 0,5 < Kcomp =< 0,7

La Competencia del experto es Baja (B): Si Kcomp =< 0,5