UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

Sistema de administración para la gestión de la información de la plataforma NOX para la Universidad de Ciencias Informáticas

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

Autora:

Yannia Alexandra Gastón González

Tutores:

MsC. Vladimir Milán Núñez

Lic. Raynel Batista Téllez

Proyecto CTI “Herramientas para el análisis de influencia de redes de innovación tecnológica”

Programa de Prioridad Nacional CITMA “Ciencia, Tecnología e Innovación”

La Habana, junio de 2019

“Año 61 de la Revolución”

“Elige un trabajo que te guste y nunca tendrás que trabajar ni un solo día de tu vida”

Confucio

Declaración de autoría

Declaro ser la autora de la presente tesis que tiene por título: “Sistema de administración para la gestión de la información de la plataforma NOX para la Universidad de Ciencias Informáticas.” y se reconoce a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____

Yannia Gastón González

Autora

_____

MsC. Vladimir Milán Núñez

Tutor

_____

Lic. Raynel Batista Téllez

Tutor

Datos de Contacto

Datos de la Autora:

Yannia Alexandra Gastón González

Universidad de las Ciencias Informáticas, Cuba.

Correo: yagaston@estudiantes.uci.cu

Datos de los tutores:

Tutor: Vladimir Milán Núñez

Correo: vladimir@uci.cu

Síntesis del tutor: Ingeniero en Ciencias Informáticas, de la Universidad de las Ciencias Informáticas desde el año 2008. Máster en Ciencias en la Universidad de las Ciencias Informáticas desde 2018. Profesor Asistente de las disciplinas física y matemática aplicada de la Universidad de las Ciencias Informáticas. Miembro del grupo de investigación de Inteligencia Artificial y Reconocimiento de Patrones desde el año 2008. Líder de proyecto y líder técnico (Arquitecto de software principal) de varios proyectos del centro de desarrollo de Telemática de la UCI desde 2008 hasta 2011. Miembro de la Asociación Cubana de Reconocimiento de Patrones (ACRP) desde el año 2013 y presidente en funciones de la delegación de base de la UCI. Editor Asociado de la Revista Cubana de Ciencias Informáticas. Miembro del comité organizador y del comité científico de varios eventos. Autor y co-autor de varias publicaciones en revistas y memorias de eventos. Tutor de 24 tesis de pregrado.

Tutor: Raynel Batista Téllez

Correo: raynel@uci.cu

Síntesis del tutor: Sociólogo, editor, Coordinador General de Ediciones Futuro, sello editorial de la Universidad de las Ciencias Informáticas. Profesor Auxiliar. Ha impulsado proyectos interdisciplinares que vinculan las ciencias sociales y el cálculo computacional en temas como sociocibernética, interacción hombre-computadora y social media. Es miembro de la Junta Nacional Editorial del Ministerio de Cultura, Organización Internacional de Sociología, Sociedad Cubana de Derecho e Informática, Asociación Cubana de Comunicadores Sociales, Organización Internacional de Editores, Junta Editorial de la Revista Cubana de Ciencias Informáticas. Ha mostrado interés por los nuevos entornos de lectura, las redes académicas de interacción social y la ciencia de los datos. Sus principales resultados se enmarcan en la sociología de la innovación y la antropología digital, con especial interés en la curación de contenidos digitales.

Agradecimientos

Primeramente, quiero agradecer a mi mamá, esa persona que quiero con la vida, por creer siempre en mi ya que, sin su apoyo incondicional, sus consejos y sus regaños este logro no hubiese sido posible. Gracias mami. Gracias por tantos sacrificios, por luchar tanto por mí y por quererme tanto. Espero que hoy te sientas muy orgullosa de ver como logro mis objetivos y quiero que sepas que si lo hice... todo fue gracias a ti. ¨

A mi papá por ser esa estrella que ilumina mi camino, por todo lo que me enseñó el tiempo que estuvo a mi lado, por ser mi guía en muchas cosas, por ser quien soy, Sé que donde quiera que se encuentre está muy orgulloso de mi y que si pudiera correría a abrazarme y darme un beso.

A mi hermana querida por estar siempre ahí para lo que necesite sin importar lo que sea, por apoyarme tanto y preocuparse siempre por mí en cada momento de mi carrera y de mi vida, por demostrarme que es incondicional... por todo... muchas gracias.

A esa personita que sólo lleva 5 añitos en mi vida, gracias por hacerme feliz y poner siempre una sonrisa en mi rostro en cada momento de estrés y de tristeza... a mi sobri bello Yues Daniel.

A Tión por preocuparse siempre por mí, por mimarme tanto, por estar siempre al tanto de mí, por sus consejos, por todo... muchas gracias. Me alegra saber que estás muy orgulloso de mi, al fin tu sobrina es ingeniera.

A mi tía Ali que sé que me quiere con locura y que estuvo al tanto de mí en todo momento de mi carrera, gracias por tu apoyo y preocupación.

A toda mi familia gracias por su apoyo, por su preocupación, por su cariño... por estar ahí.

A mi novio Manuel por estar siempre a mi lado e intentar relajarme en cada momento de estrés en la realización de este trabajo, por siempre hacerme saber que todo iba a salir bien, por su apoyo incondicional, por aguantar mi carácter, por mimarme tanto... muchas gracias.

A mis ¨colegas¨ con las que he pasado tantas cosas que sería imposible hablar de mi vida universitaria sin hablar de ellas. Gracias por haber estado a mi lado en estos 5 años, si hay algo que nunca olvidaré son las bromas, las largas horas de estudio y las estrategias para las pruebas.

A Mimí, esa amiga que con el paso del tiempo se fue ganando mi confianza, por regañarme cuando me equivoco y por tener tanta paciencia conmigo.

A Miyi por acompañarme siempre en todas mis locuras, por seguirme en cada una de mis ocurrencias y por alegrarme siempre el día, aunque fuera el peor.

A mi cuadro favorito Omarito, por ser esa persona con la que siempre puedo contar sea lo que sea, por cada vez que me descuidaba un poco intentar que fuese una persona aplicada... eso te lo agradeceré siempre.

A mi amigo Alejito, por desde el primer día demostrarme la gran persona que es, por sus recomendaciones para exponer mejor y por ser un excelente amigo.

A mi mala influencia Luis Hansel, por apoyarme tanto, por sus consejos y por ser un amigo excepcional.

A mis compañeros de año por recorrer juntos este camino.

A todos mis profesores por enseñarme.

A mis tutores Raynel y Vladimir por aceptarme cuando fui a pedirles un tema de tesis, por las horas en la biblioteca, por su ayuda, por sus consejos, por su entrega y dedicación... gracias.

A mí, por creer en mi misma y por siempre saber que podía hacerlo.

Dedicatoria

Quiero dedicar más que este trabajo, este logro a las personas más importantes de mi vida:
A lo más grande que tengo a mi mamá, esto es para ti.
A mi papá
A ti mi hermana querida.
A ti Tión.
A ti mi sobri, para demostrarte que cuando uno se propone una meta, siempre la puede cumplir, ojalá y un día sea un ejemplo para ti.

Resumen

La plataforma NOX es una red de colaboración científica propuesta para la comunidad de investigadores de la Universidad de las Ciencias Informáticas en la cual cada investigador puede identificar información relevante sobre la actividad científica e interactuar con otros investigadores de su perfil. NOX registra, consume y almacena grandes volúmenes de datos de diversas fuentes de información. El presente trabajo expone NOX-Admin, un sistema desarrollado con el objetivo de administrar los datos para la gestión de la información en la plataforma NOX. El mismo contribuye a la segmentación, seguridad, consistencia y disponibilidad de la información de forma segura y ordenada. La solución favorece a la automatización del proceso de estudio de la actividad científica en el contexto universitario, así como, al análisis de influencia de redes de innovación tecnológica. Para llevar a cabo su implementación se hace uso de la metodología de desarrollo Proceso Unificado Ágil en la versión de la Universidad de Ciencias Informáticas, se utiliza el lenguaje de programación Python v3.5.1, para el cual su emplea el marco de trabajo Django v2.0.3 con el IDE PyCharm v1.2. Además, fue utilizada la herramienta Visual Paradigm v5.0 para el modelado y diseño del sistema.

Palabras Clave: gestión de la información, sistemas de administración de datos, análisis de actividad científica.

Abstract

The NOX platform is a scientific collaboration network proposed for the research community of the University of Informatics Sciences in which each researcher can identify relevant information about the scientific activity and interact with other researchers of his profile. NOX records, consumes and stores large volumes of data from various sources of information. The present work exposes NOX-Admin, a system developed with the objective of managing data for the management of information on the NOX platform. It contributes to the segmentation, security, consistency and availability of information in a safe and orderly manner. The solution favors the automation of the process of studying scientific activity in the university context, as well as the analysis of the influence of technological innovation networks. To carry out its implementation, the Agile Unified Process development methodology is used in the version of the University of Computer Sciences, using the Python programming language v3.5.1, for which its uses the Django v2 framework. 0.3 with the PyCharm IDE v1.2. In addition, the Visual Paradigm v5.0 tool was used for the modeling and design of the system.

Keywords: information management, data management systems, scientific activity analysis.

Índice de contenidos

Índice de tablas

Índice de figuras

Introducción

El avance de las Tecnologías de la Información y las Comunicaciones (TIC) ha servido como punto de apoyo para el desarrollo de la sociedad actual, contribuyendo con la formación y la productividad del hombre. A su vez, se ha incrementado la actividad científica con la aparición de redes sociales académicas, entre las que se pueden mencionar: ResearchGate**,** Academia.edu**,** Mendeley**,** My Science Work y Linkedin, las cuales constituyen un nuevo modelo de divulgación de la ciencia y han experimentado un gran crecimiento en los últimos años, debido precisamente a sus potencialidades para la socialización de contenidos (OpenMind, 2019).

Debido a la existencia de diversas redes académicas, se produce una dispersión de los datos de los investigadores cubanos que pertenecen a las mismas; ya que un mismo investigador puede tener un perfil en cada una de estas redes, pero cada perfil cuenta con datos distintos. En Cuba en los últimos tiempos se ha incrementado la incorporación del uso de las TIC a la vida cotidiana del pueblo. El país se ha trazado como objetivo alcanzar un nivel relevante en la producción y desarrollo de software, el cual consiste en resolver problemas que existen en la vida cotidiana del hombre mediante el uso avanzado de las TIC, en aras de aumentar las relaciones entre el mundo informático y la población trabajadora (Diaz Lazo , y otros, 2011).

La Universidad de Ciencias Informáticas (UCI) como una entidad educativa, centro clave en la investigación y producción de software para la informatización del país, posee múltiples líneas investigativas que a la vez contienen grupos y proyectos de investigación. La información de estos grupos, líneas, proyectos e investigadores, al momento de este estudio se encuentra dispersa en las diversas redes de colaboración antes mencionadas, las cuales poseen en ocasiones, investigaciones y datos comunes; por lo que, el proyecto SNA (Análisis de redes sociales, del inglés: *Social Network Analysis*) de la universidad propone como iniciativa la creación de una red de interacción social para la innovación: la plataforma NOX.

La plataforma NOX es una red de colaboración científica, en la cual cada investigador puede identificar los logros y avances de los proyectos y grupos existentes, así como interactuar y conectar con otros investigadores de su perfil. NOX contiene toda la información referente al trabajo investigativo de la universidad la cual posee nueve principales líneas de investigación, diecinueve grupos y veintinueve proyectos (Akademos, 2019), así como diversas publicaciones en distintas fuentes. Además, tiene un alto nivel de interacción debido a la cantidad de investigadores que tienen la posibilidad de registrarse en la misma e interactuar entre sí. Por lo que, representa una estructura compleja debido a la cantidad de información que gestiona y el volumen de usuarios que registran publicaciones y realizan actividades.

Debido a que los sistemas basados en la web, están siendo cada vez más accedidos a través de las redes por usuarios y sistemas, estos son susceptibles al acceso de personas no autorizadas. Por ello es necesario tener en cuenta la seguridad y la protección de los datos, aspectos fundamentales para lograr la integridad y disponibilidad de la información.

NOX compromete la disponibilidad de sus datos y no garantiza su completa seguridad, ya que se centra más en el flujo de información que en su protección en sí. El objetivo principal de la plataforma es la centralización y presentación de la información científico-técnica de la universidad y no la definición de quién puede visualizar y modificar esa información. Esto trae consigo que la gestión de permisos según los roles de los investigadores sea limitada, debido a que no se manejan por las responsabilidades que debe tener cada cual, ya que NOX cuenta con un único rol el cual puede acceder y modificar cualquier tipo de información sin importar si está autorizado para ello o no.

El control de los datos en la plataforma no se realiza de manera eficiente. La actualización de la información relacionada con este proceso se realiza a nivel de base de datos, provocando que sean los desarrolladores del sistema los que permitan la disponibilidad de la misma y no los especialistas asociados al seguimiento y control de la actividad científica en la plataforma. Cuando un investigador propone un grupo o línea de investigación, este debe ponerse en contacto con los directivos encargados del control de la información científico-técnica en la universidad, estos al no tener forma de aceptar dicha propuesta deben acudir al equipo de desarrollo, los cuales directamente desde la base de datos realizan dicha aceptación. Esto conlleva a que se puedan cometer errores en el control de los datos, atrasos en la respuesta a peticiones de los usuarios y demora en la disponibilidad de la información.

Actualmente, NOX no permite analizar y resumir de una forma detallada el gran cúmulo de datos relacionados con la información científico-técnica que maneja, de modo que esta sea monitorizada y controlada por las personas encargadas. Por lo que no es posible conocer qué está sucediendo en la plataforma, cuántos investigadores se registran y acceden a la misma, cuáles son sus temas de preferencia y sus necesidades, para poder trabajar en base a los intereses que presenten.

A partir del contexto anterior, se identifica el siguiente **problema a resolver**: el modo en que la plataforma NOX controla sus datos limita la seguridad y disponibilidad de la información que gestiona.

Se considera como **objeto de estudio**: sistemas de administración para la gestión de información científico-técnica.

Para darle solución a la problemática existente se establece como **objetivo general**: desarrollar un sistema de administración que mediante el control de los datos contribuya a la seguridad y disponibilidad de la información en la plataforma NOX.

A partir del objetivo general se derivan los siguientes **objetivos específicos**:

- Caracterizar los principales conceptos, contribuciones y tecnologías relacionados con la administración de sistemas para la gestión de la información científico-técnica.
- Implementar el sistema para la administración de la plataforma NOX en correspondencia con los requisitos, características, herramientas y tecnologías seleccionadas.
- Validar la solución propuesta mediante la aplicación de pruebas de software.

Como **campo de acción** se precisa: control de datos en la plataforma NOX.

Como **idea a defender** se plantea: si se desarrolla un sistema de administración para el control de los datos en la plataforma NOX este contribuiría a la seguridad y disponibilidad de la información.

Para posibilitar el cumplimiento de los objetivos trazados se utilizan métodos teóricos y empíricos.
De los **Métodos Teóricos** se emplean los siguientes:

- Analítico–Sintético:

Para realizar el estudio del estado del arte de tecnologías, herramientas y metodologías para desarrollar aplicaciones web, y a partir del estudio realizado obtener los elementos significativos y arribar a conclusiones para dar solución al problema.

- Histórico–Lógico:

Para conocer la evolución y desarrollo del tema tratado en la investigación, y además realizar un estudio crítico de trabajos anteriores como puntos de referencia y de comparación con los resultados alcanzados.

De los **Métodos Empíricos** se emplean los siguientes:

- Entrevista:

Facilita la obtención de información acerca de las actividades que realizan los usuarios y cuáles son las principales deficiencias que presenta la plataforma NOX.

- Encuesta:

Permite recopilar gran cantidad de información mediante cuestionarios realizados a especialistas del proyecto SNA, investigadores destacados y líderes científicos, sobre las herramientas de gestión de información científico-técnica y la importancia que tienen las mismas.

Para lograr un mejor entendimiento del presente trabajo de diploma, el mismo se ha distribuido de la forma descrita a continuación:

Capítulo 1 Fundamentación teórica: Constituye la base teórica de la investigación en cuestión. Se realiza un estudio de los principales conceptos relacionados a la gestión de la información científico-

técnica. Se hace un análisis de herramientas y sistemas para el control de datos en aplicaciones web y se describen las principales herramientas y tecnologías a utilizar para el desarrollo de la solución.

Capítulo 2 Análisis, Diseño e implementación de la solución propuesta: Se describe la solución que se propone, dando a conocer primeramente el modelo conceptual. Se definen los requisitos funcionales y no funcionales que ofrecen una solución al problema de investigación. Además, se generan los artefactos según la metodología seleccionada AUP versión UCI en su escenario número dos: casos de uso del sistema.

Capítulo 3 Implementación y pruebas: Se exponen el modelo de implementación mediante los diagramas de componentes y de despliegue. Se describen los resultados de la implementación de la aplicación y su funcionamiento. Finalmente se realiza la validación de la investigación y se realizan pruebas comprobando el cumplimiento del objetivo general.

Capítulo 1: Fundamentación teórica

En el presente capítulo se exponen diferentes sistemas y herramientas relacionadas con la gestión de la información en sistemas de administración. De estos se abordan sus principales características para ayudar en la solución del problema en cuestión. Por lo que uno de los objetivos que se persigue en el capítulo es la realización de un estudio referente al estado del arte, así como de las tecnologías, sus tendencias, herramientas, metodología aplicada y lenguajes que serán utilizados para el desarrollo de la solución.

1.1 Conceptos fundamentales

En la presente sección se muestran los conceptos fundamentales relacionados con la investigación, con el fin de lograr un mayor entendimiento del problema en cuestión.

- **Seguridad**

La Seguridad Informática es un conjunto de características y condiciones de sistemas de gestión de la información, para garantizar su confidencialidad, integridad y disponibilidad (Oracle, 2011).
La autorización es el procedimiento para determinar si el usuario o proceso previamente identificado y autenticado tiene permitido el acceso a los recursos (Oracle, 2011). Se implementa con uno de los siguientes modelos de control de acceso (Commerce, 2012):

- ✓ Control de Acceso Obligatorio (MAC): en este modelo es el sistema quien protege los recursos, comparando las etiquetas del sujeto que accede frente al recurso accedido, o sea, la autorización para que un sujeto acceda a un objeto depende de los niveles de seguridad que tenga, ya que estos indican, qué permiso de seguridad tiene el sujeto y el nivel de sensibilidad del objeto.
- ✓ Control de Acceso Discrecional (DAC): este modelo se ha venido usando de forma abundante en sistemas operativos de propósito general, en la mayoría de los sistemas de base de datos y en sistemas de comunicaciones de propósito comercial. Un usuario bien identificado, por lo general el creador o propietario del recurso, decide cómo protegerlo estableciendo cómo compartirlo, mediante controles de acceso impuestos por el sistema. Lo fundamental es que el propietario del recurso puede cederlo a un tercero.
- ✓ Control de Acceso Basado en Roles (RBAC): este modelo es un intento de unificar los modelos clásicos DAC y MAC, logrando un sistema que impone el control de acceso, pero sin las restricciones rígidas impuestas por las etiquetas de seguridad.

La seguridad informática consiste en asegurar que los recursos del sistema de información de una organización se utilizan de la manera que se decidió y que el acceso a la información allí contenida, así

como su modificación, solo sea posible para las personas que se encuentren acreditadas y dentro de los límites de su autorización (Martinez Sánchez, 2018).

A partir del anterior análisis la autora de la investigación define que la seguridad informática se base en proteger la información y recursos de un sistema, esta tiene 3 aspectos fundamentales:

- ✓ Confidencialidad
- ✓ Integridad
- ✓ Disponibilidad

Para lograr una mayor seguridad en un software este debe contar con roles y permisos para limitar el acceso al sistema a través de un modelo de control de acceso.

- **Disponibilidad**

Propiedad de los activos de información, mediante la cual aquellas entidades o procesos autorizados tienen acceso a los mismos cuando lo requieren. Es aquella cualidad de la información electrónica que permite su localización, recuperación, presentación e interpretación (Franco Espiño, y otros, 2015).

Se trata de la capacidad de un servicio, de unos datos o de un sistema a ser accesible y utilizable por los usuarios o procesos autorizados cuando lo requieran. También se refiere a la capacidad de que la información pueda ser recuperada en el momento que se necesite (Martinez Sánchez, 2018) .

Para la presente investigación, teniendo en cuenta las definiciones estudiadas se tiene como disponibilidad: la condición que posee la información o un servicio de estar disponible cuando se requiera para las personas autorizadas.

- **Sistema de administración**

El conjunto de componentes que interactúan entre sí y se encuentran interrelacionados recibe el nombre de sistema. Administrativo, por su parte, es aquello vinculado a la administración, es el acto de administrar: organizar o gestionar recursos (Pérez Porto , y otros, 2014). En su acepción más amplia, un sistema administrativo es una red o un esquema de procesos cuya finalidad es favorecer el cumplimiento de los objetivos de una organización (Pérez Porto , y otros, 2014).

Para garantizar el cumplimiento de los objetivos de las aplicaciones web, estas deben contar con una adecuada administración, que brinde soporte a todas las actividades que se desarrollen en dicha plataforma, donde se hacen necesarias funcionalidades como: la configuración de los usuarios, la definición de servidores y bases de datos a acceder, la definición de dominios, la gestión de sesiones activas y estadísticas para el cómputo de carga y optimizar el tiempo de respuesta, buscador de objetos y herramientas colaborativas entre desarrolladores, con el objetivo de refinar todo su funcionamiento (Carbonell, 2012).

A partir de los planteamientos anteriores para el desarrollo de la presente investigación, se considera significativo dentro de la administración de aplicaciones web, la gestión de roles y permisos además de la gestión de la información.

La administración de usuarios y roles por funcionalidades es una técnica muy utilizada en la actualidad. Los sistemas en los que la seguridad de la información constituye un aspecto primordial tienen en cuenta este procedimiento con el fin de garantizar solo el acceso necesario de los usuarios a toda o parte de la información. La administración tiene que garantizar esa seguridad a partir de la concesión de permisos (Guglielmetti, 2012).

- **Gestión de la información**

La gestión de la información es el conjunto de las actividades que se realizan con el propósito de adquirir, procesar, almacenar y finalmente recuperar, de manera adecuada, la información que se produce o se recibe en una organización y que permite el desarrollo de su actividad (Suárez Alfonso, y otros, 2015). También, se puede definir como el proceso que se encarga de suministrar los recursos necesarios para la toma de decisiones, permitiendo mejorar los procesos, productos y servicios de la organización. Realizar procesos de gestión de información posibilita entre otras cosas identificar, organizar, representar y recuperar información dispersa en áreas. La gestión de la información implica (Santa Cruz Pacheco, 2015):

- ✓ Determinar la información que se precisa.
- ✓ Recoger y analizar la información.
- ✓ Registrarla y recuperarla cuando sea necesaria.
- ✓ Utilizarla.
- ✓ Divulgarla.

1.2 Gestión de la información científico-técnica

La gestión de información es una disciplina que se ocupa de utilizar los recursos básicos (económicos, físicos, humanos y materiales) para manejar información dentro y para la sociedad a la que sirve. Por otra parte, dentro de una organización se pueden notificar diferentes tipos de información según las funciones que se realizan, ésta se debe manejar y utilizar de forma adecuada y sistemática para obtener los mejores beneficios para la institución (Dante, 2005).

La gestión de la información científico-técnica integra en su organización el conjunto de instancias que la constituyen en función de hacer cumplir: cómo la información se adquiere, registra, almacena, distribuye y usa, cómo el personal designado maneja y hace llegar la información a los usuarios directos, cómo las personas usan la información, desarrollan habilidades informativas y se convierten en divulgadores de la misma, cómo las tecnologías de la información se incorporan y perfeccionan los

diferentes procesos de la gestión, cómo la captación y uso de la información incide en el crecimiento humano y organizacional con mejores resultados, todo lo que incide en los costos y beneficios de la organización (Medina Rodríguez, y otros, 2016).

Teniendo en cuenta lo planteado por Dante (2005), los objetivos de la gestión de la información científico-técnica son:

- ✓ Maximizar el valor y los beneficios derivados del uso de la información.
- ✓ Minimizar el costo de adquisición, procesamiento y uso de la información.
- ✓ Determinar responsabilidades para el uso efectivo, eficiente y económico de información.
- ✓ Asegurar un suministro continuo de la información.

Además, se evidencian tendencias que van centrando el compendio de "buenas prácticas" en la gestión de la información en las organizaciones.

- ✓ Tendencia 1: Hacia la gestión de los contenidos.
- ✓ Tendencia 2: Hacia la aceptación de los documentos electrónicos.
- ✓ Tendencia 3: Hacia la necesidad de proceso de información no estructurada.
- ✓ Tendencia 4: Hacia el reconocimiento de la tecnología como herramienta.
- ✓ Tendencia 5: Hacia la máxima importancia de la accesibilidad.
- ✓ Tendencia 6: Hacia los planteamientos a medio y largo plazo.

Teniendo en cuenta los conceptos planteados anteriormente, la gestión de la información científico-técnica, es el proceso específico dentro de la gestión de la información en el cual el investigador logra conceptualizar, formalizar y encontrar las respuestas más idóneas que resuelven una problemática que parte de la ausencia de conocimiento. Esta gestión deberá suplir las necesidades para que el mismo pueda:

- ✓ Identificar necesidades informativas.
- ✓ Realizar búsquedas de confianza.
- ✓ Confeccionar su estrategia de búsqueda.
- ✓ Seleccionar y clasificar la información en dependencia a su interés de investigación.
- ✓ Establecer comparativas referenciales.
- ✓ Recopilación y confección de sus investigaciones científicas.

1.3 Análisis de soluciones similares a la problemática

Teniendo en cuenta el estudio de la gestión de la información científico-técnica y enfocando la investigación en el campo de acción: "control de los datos en la plataforma NOX" se realiza un análisis de soluciones relacionadas con la problemática con el objetivo de definir las funcionalidades propias del sistema a desarrollar y adquirir nuevos conocimientos acerca del control de los datos para la gestión de la información científico-técnica.

Django Admin

Herramienta que utiliza el marco de trabajo (*framework*) Django para administrar aplicaciones web Dicha herramienta funciona leyendo los meta-datos en los modelos, accediendo al código, e interpretándolos para brindar una interfaz potente y orientada a desarrolladores la cual permite (Kaplan-Moss, 2013):

- ✓ La gestión de entidades.
- ✓ La gestión de usuarios, así como sus roles y permisos.

Wagtail

Wagtail es un Sistema de Gestión de Contenido (CMS, del inglés: *Content Management System*) de código abierto escrito en Python y construido por la empresa TorchBox utilizando el *framework* Django. Ofrece una atractiva interfaz rápida orientada a editores de contenido, donde se puede crear y estructurar contenido de manera intuitiva. Algunas de sus características esenciales son (Torchbox, 2019):

- ✓ Integración con aplicaciones que usan Django como *framework*.
- ✓ Configuración de los distintos tipos de contenidos a través de los modelos estándar de Django.
- ✓ Control sobre el diseño a través de las plantillas de Django.
- ✓ Sistema de permisos y roles configurable.
- ✓ Plataforma con soporte para varios idiomas.

Mezzanine

Mezzanine es un CMS desarrollado en Django. Es un editor que permite modificar el código para implementar cambios, es extensible y sencillo a la vez porque gran parte de sus funcionalidades vienen incorporadas por defecto dentro del CMS. Su licencia es de distribución de software Berkeley (BSD del inglés: *berkeley software distribution*), la cual es de software libre. Algunas características destacadas de Mezzanine son (BBVA, 2016):

- ✓ Dispone de función de borrador, vista previa y programación de publicaciones.
- ✓ Construcción de localizadores uniformes de recursos (url, del inglés: uniform resource locator) amigables, que son fáciles de entender por el usuario.
- ✓ Gestión de roles de usuario.
- ✓ Integración con aplicaciones desarrolladas con Django.

Drupal

Drupal es un proyecto comunitario de software libre descrito como " un potentísimo sistema de gestión de contenidos, que ofrece más que probadas capacidades para la creación, desarrollo y mantenimiento de servicios y productos de información digital " (Tramullas, 2010).

Drupal cuenta con licencia GPL (Licencia publica general, del inglés: *General public license*), es software libre y está escrito en PHP (Lenguaje de programación interpretado, del inglés: *Hypertext Pre-Processor*); además de ser desarrollado y mantenido por una activa comunidad de usuarios. Con Drupal se puede construir casi cualquier tipo de recurso web, donde la pieza fundamental son los contenidos que presentan un enfoque estructurado, que permite definir tipos de contenidos diferentes, sobre los que se aplican diferentes gestiones administrativas como permisos, flujos de publicación, categorías y listados. Drupal ofrece una gestión de usuarios avanzada, donde los usuarios se agrupan en roles, que permiten manejar los privilegios para cada una de las funcionalidades y módulos. Cada módulo de Drupal ofrece sus propias opciones de permisos, que se aplican por roles. Además permite la obtención de reportes mediante el uso de gráficos (Pérez, 2010).

Wordpress

WordPress es un CMS enfocado a la creación de cualquier tipo de página web. Está desarrollado en el lenguaje PHP para entornos que ejecuten MySQL y Apache, bajo licencia GPL y es software libre. Algunas de las principales características de Wordpress son (Cambronero, 2016):

- ✓ Fácil instalación, actualización y personalización.
- ✓ Múltiples autores o usuarios, junto con sus roles o perfiles que establecen distintos niveles de permisos.
- ✓ Permite ordenar artículos y páginas estáticas en categorías, subcategorías y etiquetas.
- ✓ Permite comentarios, notificaciones y herramientas de comunicación entre blogs
- ✓ Gestión y distribución de enlaces.
- ✓ Subida y gestión de datos.
- ✓ Permite realizar reportes estadísticos mediante el uso de gráficos y tablas.

Valoración a partir de las soluciones estudiadas

A partir de la información obtenida con respecto al análisis de soluciones relacionadas con la problemática, se realizó un estudio de las mismas en aras de conocer como gestionan y controlan la información. Dicho estudio arrojó como resultado que ninguna es viable para utilizarla con el fin de controlar los datos en la plataforma NOX, debido a una serie de inconvenientes que se exponen a continuación.

Tabla 1 Valoración de soluciones similares a la problemática

Sistemas – Criterio	Django-Admin	Wagtail	Mezzanine	Drupal	Wordpress
Permite la gestión de usuarios, así como sus roles y permisos.	✓	✓	✓	✓	✓
Posibilita la integración nativa (utiliza el mismo lenguaje de programación y marco de trabajo) con la plataforma NOX.	✓	✓	✓	-	-
Permite la gestión de la información.	✓	✓	✓	✓	✓
Muestra reportes estadísticos.	-	-	-	✓	✓
Permite la gestión de notificaciones.	-	-	-	✓	✓
Posee licencia libre.	✓	✓	✓	✓	✓
Posee una interfaz orientada a personas con conocimientos básicos en el campo de la informática.	-	-	-	-	-

- ✓ Las soluciones **Drupal** y **Wordpress** se pudieran integrar a la plataforma, pero no de forma nativa, por lo tanto, se dificultaría el mantenimiento de la misma en un futuro ya que si no se trabaja con el mismo equipo de desarrollo se tendría que buscar especialistas que dominen ambas tecnologías, ya que estas soluciones están escritas en el lenguaje de programación PHP y la plataforma NOX en Python.
- ✓ Las soluciones **Django-Admin**, **Wagtail** y **Mezzanine** no permiten la obtención de reportes estadísticos mediante el uso de gráficos.
- ✓ Las soluciones **Django-Admin**, **Wagtail**, **Mezzanine**, **Drupal** y **Wordpress** permiten la gestión de la información, pero esta forma de gestión no se encuentra enfocada al marco y al proceso investigativo de la universidad.
- ✓ Las soluciones **Django-Admin**, **Wagtai**l y **Mezzanine** no permiten la gestión de notificaciones para posibilitar la comunicación con la plataforma NOX.

Sin embargo, su análisis permitió identificar varias funcionalidades significativas para la gestión y el control de los datos en aplicaciones web como:

- ✓ Las soluciones **Drupal** y **Wordpress**, brindan gráficas que ayudan al monitoreo y a la toma de decisiones por parte de los directivos.

- ✓ Las soluciones **Django-Admin**, **Wagtail**, **Mezzanine**, **Drupal** y **Wordpress** permiten la gestión de usuarios y perfiles lo que posibilita la asignación de roles y permisos.

Por todas estas razones existe la necesidad de implementar un sistema que contribuya al control de los datos en la plataforma NOX que no utilice ni se ejecute sobre tecnologías privativas. Razones que impulsaron a la utilización de las herramientas y tecnologías descritas en el posterior epígrafe. De esta forma se desarrolla un entorno propio con características específicas para un producto que responda satisfactoriamente a las necesidades del cliente.

1.4 Herramientas y tecnologías

En este punto se está en condiciones de seleccionar las tecnologías y herramientas para el desarrollo del sistema, después de un análisis de las funcionalidades que requiere el mismo. Se elige la plataforma que se usa para implementar el sistema, la metodología de desarrollo, el *framework* de desarrollo, los lenguajes de programación y el sistema gestor de bases de datos.

1.4.1 Plataforma

Se denomina aplicación web a aquellas herramientas que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador (Berti, y otros). Se selecciona como plataforma para el desarrollo de la solución una aplicación web ya que actualmente resulta más práctico y aconsejable el uso de estas siempre que se necesite el acceso desde diferentes ubicaciones. La interfaz de una aplicación web no es una desventaja frente a la interfaz de una aplicación de escritorio ya que actualmente los controles web cuentan con funcionalidades y cercanías al usuario muy amplias. Esta tiene como ventaja la disponibilidad de la aplicación a través de dispositivos que tengan un navegador web. Además, son multiplataforma ya que se pueden usar desde cualquier sistema operativo (Velázquez Álvarez, 2012).

1.4.2 Metodología de desarrollo de software

Las metodologías de desarrollo de software coleccionan un conjunto de pasos y procedimientos que se deben seguir para organizar, controlar y planear el proceso de desarrollo de un software. Surgen ante la necesidad de trabajar mediante el uso de procedimientos, técnicas, herramientas y documentos durante el desarrollo de software (Pressman, 2010). Estas se clasifican en ágiles y prescriptivas o tradicionales. Entre las metodologías tradicionales se encuentran: Proceso de Desarrollo Unificado (RUP, del inglés: *Rational Unified Process*) y Marco de trabajo de soluciones de Microsoft (MSF, del inglés: *Microsoft Solutions Framework*). Estas metodologías son más usadas en proyectos complejos y de larga duración, en los cuales se requiere que el equipo de desarrollo tenga experiencia en la aplicación de las mismas (Figueroa Diaz, y otros, 2007).

Por otra parte, las metodologías ágiles hacen menos énfasis en la documentación, puesto que el funcionamiento y el desarrollo del software son más importantes. La regla a seguir es no realizar documentos a menos que sean necesarios de forma inmediata. El cliente pasa a formar parte del equipo de desarrollo (Edeki, 2013). Entre las metodologías ágiles existentes se destacan SCRUM, Crystal, Programación extrema (XP, del inglés: *Xtreme Programming*) y Proceso unificado ágil (AUP, del inglés: *Agile Unified Process*). Para la presente investigación se hará uso de la metodología de desarrollo AUP versión UCI (AUP-UCI), ya que esta es la metodología institucionalizada para los proyectos desarrollados en la UCI y sus centros de producción; la misma es una variación de AUP.

La metodología AUP-UCI propone 3 fases (Sánchez Rodriguez, 2014):

- Inicio: durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo para decidir si se ejecuta o no el proyecto.
- Ejecución: en esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el producto es transferido al ambiente de los usuarios finales o entregado al cliente. Además, en la transición se capacita a los usuarios finales sobre la utilización del software.
- Cierre: en esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Las 7 disciplinas propuestas por AUP-UCI son (Sánchez Rodriguez, 2014):

1. Modelado del negocio.
2. Requisitos.
3. Análisis y diseño.
4. Implementación.
5. Pruebas internas.
6. Pruebas de liberación.
7. Pruebas de aceptación.

AUP-UCI propone tres formas de describir los requisitos: casos de uso del sistema (CUS), historias de usuario (HU) y descripción de requisitos por proceso (DRP), agrupados en cuatro escenarios condicionados por el modelado de negocio, los cuales quedan definidos de la siguiente forma (Sánchez Rodriguez, 2014):

1. Los proyectos que modelen el negocio con casos de uso del negocio (CUN) solo pueden modelar el sistema con CUS.
2. Los proyectos que modelen el negocio con modelo conceptual (MC) solo pueden modelar el sistema con CUS.
3. Los proyectos que modelen el negocio con descripción de proceso de negocio (DPN) solo pueden modelar el sistema con DRP.
4. Los proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Para llevar a cabo el modelo de negocio del presente trabajo se decide utilizar el MC por lo que solo se puede modelar el sistema mediante la descripción de los requisitos a través de los CUS. Según el Programa de Mejora de la Metodología de desarrollo para la Actividad productiva de la UCI, este escenario ha sido recomendado para proyectos donde el objetivo primario es la gestión y presentación de información (Sánchez Rodriguez, 2014).

1.4.3 Herramienta y lenguaje de modelado

- **Visual Paradigm 5.0**

Para el modelado de la aplicación se decide utilizar Visual Paradigm en su versión 5.0, ya que este cuenta con las características necesarias para el diseño y modelado del sistema a implementar. Aumenta la calidad del software, a través de la mejora de la productividad en el desarrollo y mantenimiento del mismo. También permite la portabilidad y estandarización de la documentación, además del uso de las distintas metodologías propias de la Ingeniería de Software (Anacleto, 2006). Visual Paradigm es una herramienta de ingeniería de software asistida por computadoras (CASE, del inglés: *Computer Aided Software Engineering*). La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación (Paradigm, 2015).

- **Lenguaje Unificado de Modelado (UML) 2.0**

Lenguaje Unificado de Modelado (UML, del inglés: *Unified Modeling Language*) es un lenguaje de propósito general que ayuda a especificar, visualizar y documentar modelos de sistemas software, incluido su estructura y diseño, de tal forma que se unifiquen todos sus requerimientos. El objetivo principal de UML es estandarizar el modelado de sistemas software, este proporciona vocabulario y reglas para combinar y construir representaciones, modelos conceptuales y físicos del sistema; permite representar varios modelos, combinando notaciones específicas de cada uno (Expósito, y otros). La misma es de suma importancia ya que permite tener una visión más completa del sistema mediante varios tipos de diagramas. Debido a las características del problema a resolver y la aplicación a

desarrollar, se ha definido este lenguaje a utilizar ya que es de gran comodidad para el trabajo con lenguajes orientados a objetos, el cual será el utilizado en el proceso de implementación de la aplicación.

1.4.4 Marco de trabajo

Un marco de trabajo se puede definir como un conjunto de bibliotecas orientadas a la reutilización de componentes de software para el desarrollo rápido de aplicaciones. Realmente es un patrón que nos da la base de la programación del proyecto, incluyendo una forma de trabajar que es de gran utilidad cuando se está trabajando en grupo (Coexia, 2015).

Django es un *framework* web de alto nivel que permite el desarrollo de sitios web rápidos, seguros y mantenibles. Además, este es gratuito y de código abierto y tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y de pago. El código de Django está escrito usando principios y patrones de diseño para fomentar la creación de código mantenible y reutilizable. En particular, utiliza el principio No te repitas (DRY, del inglés: *Don't Repeat Yourself*) para que no exista una duplicación innecesaria, reduciendo la cantidad de código. Además, este se puede ejecutar en Linux, Windows y Mac OS X ya que es multiplataforma (Kaplan-Moss, 2013). Para el desarrollo de la solución se escoge como marco de trabajo Django en su versión 2.0.3, debido a las características que presenta, descritas anteriormente, por lo que se deciden utilizar los lenguajes descritos a continuación.

1.4.5 Lenguajes de programación

- **Python 3.5.1**

Teniendo en cuenta que el marco de trabajo seleccionado Django está basado en Python, entonces se hace uso del lenguaje de programación Python en su versión 3.5.1 para la programación del lado del servidor en la aplicación. Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel además de un enfoque simple pero efectivo a la programación orientada a objetos. Python puede ser clasificado como un lenguaje interpretado, multiplataforma, de tipado dinámico y multiparadigma (Van Rossum, 2017). A diferencia de la mayoría de los lenguajes de programación, Python nos provee de reglas de estilos, a fin de poder escribir código fuente más legible y de manera estandarizada (Van Rossum, 2017).

- **HTML 5.0**

Para la programación del lado del cliente se hace uso del Lenguaje de marcas de hipertexto (HTML, del inglés: *Hypertext Markup Language*) 5.0 es la última versión de HTML. Se trata de una nueva versión con nuevos elementos, atributos y comportamientos. Contiene un conjunto amplio de tecnologías que permite a los sitios web y a las aplicaciones ser más diversas y de gran alcance. Proporciona una mayor optimización de la velocidad y un mejor uso del hardware. Permite a las páginas web almacenar datos localmente en el lado del cliente y operar sin conexión de manera eficiente (Gauchat, 2012).

- **CSS 3.0**

Hojas de estilo en cascada (CSS, del inglés *Cascading Style Sheets*) es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML. El CSS sirve para definir la estética de un sitio web en un documento externo y eso mismo permite que modificando ese documento podamos cambiar la estética entera de un sitio web. Con la utilización de CSS se obtiene un mayor control de la presentación del sitio al poder tener todo el código CSS reunido en uno, lo que facilita su modificación (Gauchat, 2012).

- **Bootstrap 3.3.7**

Bootstrap es un *framework* CSS de código abierto. Se caracteriza por ser una excelente herramienta para crear interfaces de usuarios totalmente adaptables a cualquier tipo de dispositivo y pantalla, independientemente de su tamaño. Además es compatible con la mayoría de navegadores web del mercado como: Google Chrome, Safari, Mozilla Firefox, Internet Explorer y Opera (Pavón Mestras, 2011). También incluye numerosos componentes CSS listos para utilizar y que cubren las necesidades más habituales de los diseños actuales para la web (Thornton, y otros, 2015). Dentro de estos componentes se encuentran: los menús desplegables, grupo de botones, grupos de campos de formulario, imágenes en miniatura, mensajes de alerta, listas de elementos y paneles los cuales serán útiles para la interfaz de la propuesta de solución. Por las características que presenta, las cuales son descritas anteriormente se utiliza este marco de trabajo para el diseño de la interfaz de la solución.

- **JavaScript**

JavaScript en un lenguaje de programación interpretado, o sea, no requiere compilación. Es utilizado especialmente en páginas web embebido en el código HTML (Gauchat, 2012). Es uno de los más potentes e importantes lenguajes de programación en la actualidad por tres enfoques claros: es útil, práctico y está disponible en cualquier navegador web (Sánchez, 2001). JavaScript del lado del

frontend, agrega mayor interactividad a la web. Es un lenguaje multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientado a objetos y es imperativo (Gauchat, 2012).

1.4.6 Sistema gestor de base de datos

Se conoce como Sistema Gestor de Bases de Datos (SGBD) al conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos (Bertino, y otros, 1995). Por las características de Django, la gestión de la base de datos es independiente del gestor, en este caso se utilizó PostgresSQL en su versión 9.4. PostgreSQL es un potente SGBD de código abierto que soporta el esquema objeto relacional. Cuenta con una arquitectura probada que se ha ganado una sólida reputación por su fiabilidad e integridad de datos (Gerrero Martínez, 2010). Este funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema. Además, es ideal para tecnologías web, es fácil de administrar, es multiplataforma y posee capacidades de recuperación de datos (Gerrero Martínez, 2010).

1.4.7 Entorno de desarrollo integrado

Un entorno de desarrollo integrado (IDE, del inglés: *Integrated Development Environment*), es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica (GUI). En algunos lenguajes, un IDE puede funcionar como un sistema en tiempo de ejecución, en donde se permite utilizar el lenguaje de programación en forma interactiva, sin necesidad de trabajo orientado a archivos de texto (Fergarciac, 2013).

PyCharm es un IDE que forma parte de la suite de herramientas de programación ofrecidas por JetBrains. Tiene un editor inteligente, que permite completar código con algunos atajos de teclado. Asimismo, permite navegar a través del código, saltando entre las clases y métodos creados, haciendo el flujo de trabajo mucho más dinámico. Una de las características notables de PyCharm es la posibilidad que tiene de refactorizar el código, que, en términos generales, significa modificar el código sin comprometer la ejecución del mismo. Es multiplataforma, hay binarios para: Windows, Linux y Mac OS X. Proporciona análisis de código, depuración gráfica, integración con VCS (Sistema de Control de Versiones), DVCS (Sistema de Control de Versiones Distribuidos) y soporte para el desarrollo web con Django (Turnquist, 2011). Se decide escoger este IDE, ya que además de las ventajas y facilidades que presenta (mencionadas anteriormente), soporta el lenguaje de programación Python, el cual fue seleccionado para llevar a cabo el desarrollo del sistema.

Conclusiones del capítulo

A partir del análisis de la información presentada en este capítulo se arriba a las siguientes conclusiones:

- El análisis de los conceptos fundamentales asociados al control de los datos permitió identificar la seguridad y la disponibilidad como aspectos fundamentales en sistemas de administración para la gestión de la información científico-técnica.
- El estudio comparativo de soluciones similares para la administración de sistemas resaltó la gestión de roles y permisos, la gestión de la información y el uso de tecnologías libres como características más comunes, mientras que la obtención de reportes estadísticos, la gestión de notificaciones y el desarrollo de interfaces amigables para el usuario destacan por ser las principales limitaciones en estos sistemas.
- El análisis de las características que distinguen a los sistemas de administración estudiados y su relación con la gestión de la información científico-técnica en el contexto universitario permitió reorientarlas al ambiente de innovación de la UCI y los requerimientos de la plataforma NOX.
- La identificación de las características propuestas para la solución, el análisis de las tecnologías que distinguen a soluciones similares para la administración de sistemas y los requerimientos de las plataformas NOX, contribuyó a la selección de las herramientas y tecnologías más adecuadas para desarrollar la propuesta de solución, destacando como metodología de desarrollo AUP-UCI, como marco de trabajo Django y como herramienta de modelado Visual Paradigm usando como lenguaje UML.

Capítulo 2: Análisis, diseño e implementación de la solución propuesta

El presente capítulo tiene como objetivo describir el proceso de construcción de la solución y hacer énfasis en las principales funcionalidades que tendrá el sistema siguiendo la metodología de desarrollo AUP-UCI en su escenario número dos, CUS. Se realiza la descripción del modelo conceptual para a partir de este, definir los pasos con el fin de comenzar con el modelado y diseño del Sistema de Administración de la plataforma NOX para la UCI (NOX-Admin). Además, se identifican los requisitos funcionales y no funcionales con los que debe cumplir la aplicación, la descripción de casos de uso del sistema, así como el diseño de las clases de la propuesta de solución.

2.1 Propuesta de solución

Para darle solución a la problemática se pudiera extender la plataforma NOX adicionándole un módulo de administración o un componente para el control de los datos. Esto traería consigo la descentralización de la información y dificultaría el control de la misma debido a la cantidad de funcionalidades que tendría la aplicación. A su vez, una persona que posea roles administrativos al acceder al sistema se cargarían demasiado las interfaces del mismo al este tener que mostrar las opciones y servicios que brinda NOX, además de las funciones administrativas. Por lo tanto, para el desarrollo de la presente investigación se propone como solución la creación de un sistema de administración que solo sea accedido por los usuarios autorizados, contribuyendo así a la seguridad y disponibilidad de la información de modo que esta se encuentre centralizada y se facilite su control; además de esta forma sería más fácil su posterior extensión si se requiere. Este sistema debe contar con un:

- ✓ Módulo para la gestión de perfiles de NOX donde se definan los roles y permisos de cada usuario.
- ✓ Módulo para la gestión de investigaciones donde se administre toda la información referente a los grupos líneas y proyectos de investigación.
- ✓ Módulo estadístico donde se puedan visualizar las entidades investigativas (líneas, grupos y proyectos de investigación) de preferencia para los investigadores, así como llevar un control del registro de usuarios en la plataforma.
- ✓ Módulo para la gestión de eventos y revistas que estarán disponibles para los investigadores autenticados en NOX.
- ✓ Sistema de notificaciones y alertas asociado, el cual informará al administrador y demás roles de todas las peticiones y acciones de los usuarios de la plataforma.

2.2 Modelo conceptual

Un modelo conceptual, es una representación visual en forma de diagrama de las clases conceptuales u objetos del mundo real que son significativos en un dominio de interés; no se trata de un conjunto de diagramas que describen clases u objetos de *software* con responsabilidades (Larman, 1999). La meta de análisis del dominio es clara: encontrar o crear aquellas clases o patrones de análisis que sean aplicables en lo general, de modo que puedan volverse a usar (Pressman, 2010).

2.2.1 Diagrama del modelo del dominio

La Figura 1 expuesta a continuación muestra el diseño del diagrama del modelo de dominio. El mismo representa la relación entre los conceptos fundamentales del entorno donde se desarrollará la propuesta de solución.

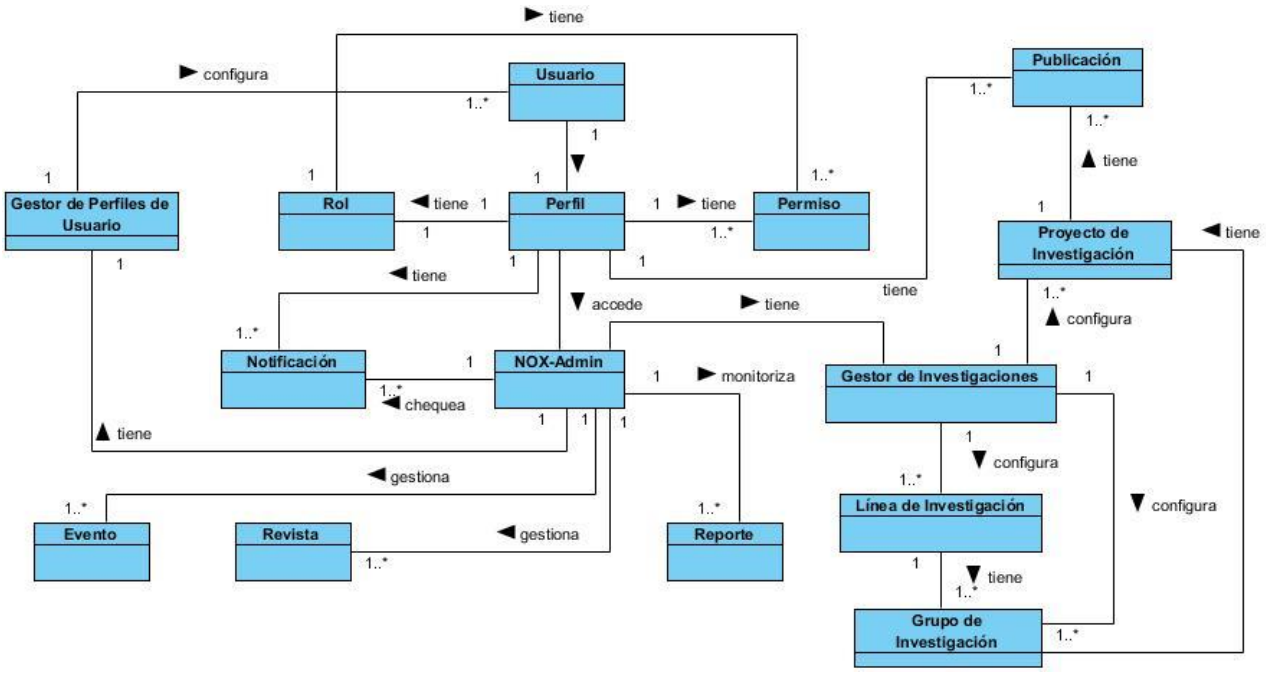


Figura 1 Diagrama del modelo del dominio

Descripción del modelo de dominio

El modelo de dominio representa cómo un usuario interactúa con el sistema NOX-Admin, donde se gestionan los proyectos, las líneas, grupos de investigación, revistas, eventos y perfiles de usuario. Este usuario posee determinados permisos en dependencia del rol que se le otorga. El monitoreo de las investigaciones, los permisos sobre el sistema, así como la gestión de usuarios son controlados por el administrador. Un usuario cuyo rol sea jefe de investigaciones tiene permisos de gestión sobre todo lo referente al trabajo investigativo como las líneas, los grupos y los proyectos. Así como el editor de

eventos y revistas tiene permisos de gestión sobre las revistas y eventos del sistema. Todos los usuarios poseen notificaciones las cuales son chequeadas al acceder al sistema.

Descripción de los conceptos representados en el diagrama

Para lograr una mejor comprensión del dominio del negocio se hace necesaria la descripción de los siguientes conceptos:

- **Usuario**: persona registrada en el sistema.
- **Perfil**: conjunto de datos adicionales asociados a un usuario el cual contiene información referente al investigador en cuestión.
- **NOX-Admin**: sistema de administración de la plataforma NOX, el cual solo será accesible para los administradores del sistema, los jefes de investigaciones y los editores de eventos y revistas.
- **Rol:** agrupación de responsabilidades que pueden desempeñar los usuarios.
- **Permiso:** reglas que definen las acciones que pueden realizar los usuarios sobre el sistema.
- **Gestor de Perfiles de Usuario**: módulo de configuración que permite gestionar los perfiles de usuario registrados en el sistema; así como otorgarles roles y permisos.
- **Gestor de Investigaciones:** módulo de configuración que permite gestionar el contenido referente a las investigaciones: líneas de investigación, grupos de investigación y proyectos de investigación.
- **Notificación:** tipo de contenido que representa las notificaciones de cada usuario.
- **Reportes:** conjunto de gráficos que permiten llevar a cabo un monitoreo del acontecer de NOX.
- **Línea**: entidad investigativa que contiene información científico-técnica, además de grupos y proyectos.
- **Grupo**: entidad investigativa que contiene información científico-técnica, además de proyectos.
- **Proyecto**: entidad investigativa que contiene información científico-técnica, además de miembros (perfiles) y publicaciones.
- **Revista**: conjunto de datos que contienen información con la cual se realiza una búsqueda y se obtienen revistas reales de la web.
- **Evento**: suceso o actividad programada y organizada para los usuarios del sistema.
- **Publicación**: documento de contenido científico elaborado por un investigador, este puede pertenecer al propio investigador o a un proyecto.

Una vez analizado el modelo de dominio y definidos los conceptos identificados en el mismo, se realiza el modelado del sistema NOX-Admin, partiendo de la especificación de los requisitos que el mismo debe cumplir.

2.3 Requisitos del software

La ingeniería de requisitos es el proceso de desarrollar una especificación de software. Las especificaciones pretenden comunicar las necesidades del sistema del cliente a los desarrolladores del sistema (Somerville, 2011). También ayuda a los ingenieros de software a entender mejor el problema en cuya solución trabajan. Incluye el conjunto de tareas que conducen a comprender cuál será el impacto del software sobre el negocio, qué es lo que el cliente quiere y cómo interactuarán los usuarios finales con el software (Pressman, 2010). El cliente intenta replantear un sistema confuso, a nivel de descripción de datos, funciones y comportamiento, en detalles concretos. El desarrollador actúa como interrogador y como negociador. El análisis de requisitos permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimiento), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software (Pressman, 2010). Para la obtención de los requisitos se utilizará como técnica la entrevista con el cliente, la cual facilitará el contacto directo con el mismo y un mejor entendimiento de las funcionalidades que el sistema requiere. Además, se hará uso de la tormenta de ideas, la cual permite el intercambio de ideas con un grupo de profesionales; este debate permitirá concretar los requisitos identificados en la entrevista realizada al cliente.

2.3.1 Requisitos funcionales

Es fundamental conocer los Requisitos Funcionales (RF) para poder identificar qué debe hacer el sistema y entender su funcionamiento, por lo que un RF es una capacidad o condición que el sistema debe cumplir (Pressman, 2010). A continuación, se definen los requisitos funcionales del sistema NOX-Admin, los cuales son agrupados posteriormente en 11 casos de uso del sistema.

Tabla 2 Requisitos funcionales del sistema

Requisitos Funcionales del Sistema	
RF1. Autenticar Usuario	RF27. Ver datos de grupo.
RF2. Mostrar cantidad de perfiles del sistema	RF28. Listar proyectos.
RF3. Mostrar cantidad de líneas del sistema.	RF29. Crear proyecto.
RF4. Mostrar cantidad de grupos del sistema	RF30. Modificar proyecto.
RF5. Mostrar cantidad de proyectos del sistema	RF31. Eliminar proyecto.
RF6. Mostrar cantidad de revistas del sistema	RF32. Ver datos de proyecto.
RF7. Mostrar cantidad de eventos del sistema	RF33. Listar miembros de proyecto.
RF8. Mostrar últimos usuarios registrados	RF34. Listar solicitudes de membrecía a proyecto.
RF9. Mostrar usuarios con más relaciones de amistad.	RF35. Asignar miembro a proyecto.
RF10. Mostrar líneas con más seguidores	RF36. Eliminar miembro de proyecto
RF11. Mostrar grupos con más seguidores	RF37. Listar revistas.
RF12. Mostrar cantidad de usuarios registrados por fecha.	RF38. Crear revista.
RF13. Listar perfiles.	RF39. Modificar revista.
RF14. Crear perfil.	RF40. Eliminar revista.

RF15. Modificar perfil.	RF41. Ver datos de revista.
RF16. Eliminar perfil.	RF42. Listar eventos.
RF17. Ver datos de perfil.	RF43. Crear evento.
RF18. Listar líneas.	RF44. Modificar evento.
RF19. Crear línea.	RF45. Eliminar evento.
RF20. Modificar línea.	RF46. Ver datos de evento.
RF21. Eliminar línea.	RF47. Listar notificaciones.
RF22. Ver datos de línea.	RF48. Responder a petición de membrecía.
RF23. Listar grupos.	RF49. Responder a propuesta de línea.
RF24. Crear grupo.	RF50. Responder a propuesta de grupo.
RF25. Modificar grupo.	RF51. Visualizar nuevo proyecto creado.
RF26. Eliminar grupo.	RF52. Visualizar recomendación.

Con la definición de los requisitos funcionales quedan cubiertas las necesidades del cliente en el momento de la investigación; aunque al sistema en un futuro se le pueden realizar actualizaciones en dependencia de nuevas necesidades por parte de los investigadores una vez el mismo sea utilizado. Tal es el caso de los requisitos funcionales referentes a los reportes estadísticos, ya que según los anteriores requisitos el sistema puede mostrar los grupos y líneas más seguidas por los usuarios, así como mostrar las cantidades de usuarios registrados por fecha, lo cual pudiera ampliarse adicionando nuevos gráficos según requiera el cliente.

2.3.2 Requisitos no funcionales

Los requisitos no funcionales (RNF), son requisitos que imponen restricciones en el diseño o la implementación como restricciones en el diseño o estándares de calidad. Son propiedades o cualidades que el producto debe tener (Pressman, 2010). Estos están vinculados a RF, es decir, una vez que se conozca lo que el sistema debe hacer se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido debe ser (Pressman, 2010). A continuación, se definen los requisitos no funcionales del sistema NOX-Admin.

Requisitos de Software

RNF 1. Servidor

- ✓ El servidor de base de datos debe ser PostgreSQL 9.4 o superior.
- ✓ El servidor de la aplicación debe tener instalado los módulos: python 3.5 y psycopg 2.6.2.
- ✓ El servidor debe tener instalado un intérprete de Python en su versión 3.5 o superior.

RNF 2. PC (del inglés: *Personal Computer*) cliente

- ✓ Las estaciones clientes deberán tener instalado un navegador, preferiblemente Mozilla Firefox 57.0 o superior.

Requisitos de Hardware

RNF 3. Cliente

- ✓ La computadora cliente debe poseer un Procesador Pentium 4 o superior y 512 MB de Memoria de Acceso Aleatorio (RAM, del inglés: *Random Access Memory*) como mínimo.

RNF 4. Servidor

- ✓ La computadora del servidor debe poseer un Procesador Intel Core i3 o superior y una RAM de 4GB o superior.

Seguridad

RNF 5. Restringir el acceso al sistema

- ✓ El sistema debe verificar que el usuario esté autenticado antes de que pueda realizar alguna acción sobre el sistema.
- ✓ Se establecerán roles para acceder al sistema, garantizando que la información almacenada solo podrá ser modificada y/o visualizada por los usuarios autorizados.

Usabilidad

RNF 7. Información ante errores del sistema

- ✓ Los mensajes de error del sistema deben incluir una descripción textual del error.

RNF 8. Títulos sugerentes

- ✓ Las etiquetas de cada funcionalidad y los campos de cada interfaz deben tener títulos asociados a su función de negocio.

RNF 9. Interfaz de fácil manipulación

- ✓ El sistema debe contar con una interfaz amigable para el usuario, con un diseño sencillo que permita el balance adecuado entre funcionalidad y simplicidad; logrando así que no resulte engorrosa la utilización del sistema para el usuario final.

2.4 Casos de uso del sistema

Un Caso de Uso (CU) especifica una secuencia de acciones que el sistema puede llevar a cabo. El diagrama de CUS se utiliza para ilustrar los requisitos del sistema. Además, es una representación gráfica de los procesos y su interacción con los actores (Jacobson, y otros, 2000).

2.4.1 Definición de los actores

Un actor del negocio es cualquier individuo, grupo, entidad, organización, máquina o sistema de información externos; con los que el negocio interactúa. Lo que se modela como actor es el rol que se juega cuando se interactúa con el negocio para beneficiarse de sus resultados (Pressman, 2010). A continuación, se describen los actores de la aplicación a desarrollar.

Tabla 3 Actores del Sistema

Actores	Descripción
Administrador	Usuario con todos los permisos sobre la aplicación. Se encarga de gestionar los usuarios y sus roles, así como modificar la información mostrada en el sistema.
Jefe de investigaciones	Usuarios con permisos de gestión sobre las líneas, los grupos que pertenecen a ellas, así como de los proyectos que pertenecen a los grupos
Editor de eventos y revistas	Usuarios con permisos de gestión sobre las revistas y eventos del sistema

2.4.2 Definición de casos de uso del sistema

Los RF definidos anteriormente en el epígrafe 2.3.1 se agrupan en 11 CU, los cuales se muestran a continuación:

- Autenticar usuario.
- Mostrar estadísticas.
- Obtener reportes.
- Administrar perfiles.
- Administrar líneas.
- Administrar grupos.
- Administrar proyectos.
- Gestionar miembros.
- Gestionar revistas.
- Gestionar eventos.
- Chequear notificaciones.

La Figura 2 muestra el diagrama de CUS de NOX-Admin, el cual ayuda a comprender los procesos del sistema y su interacción con los actores.

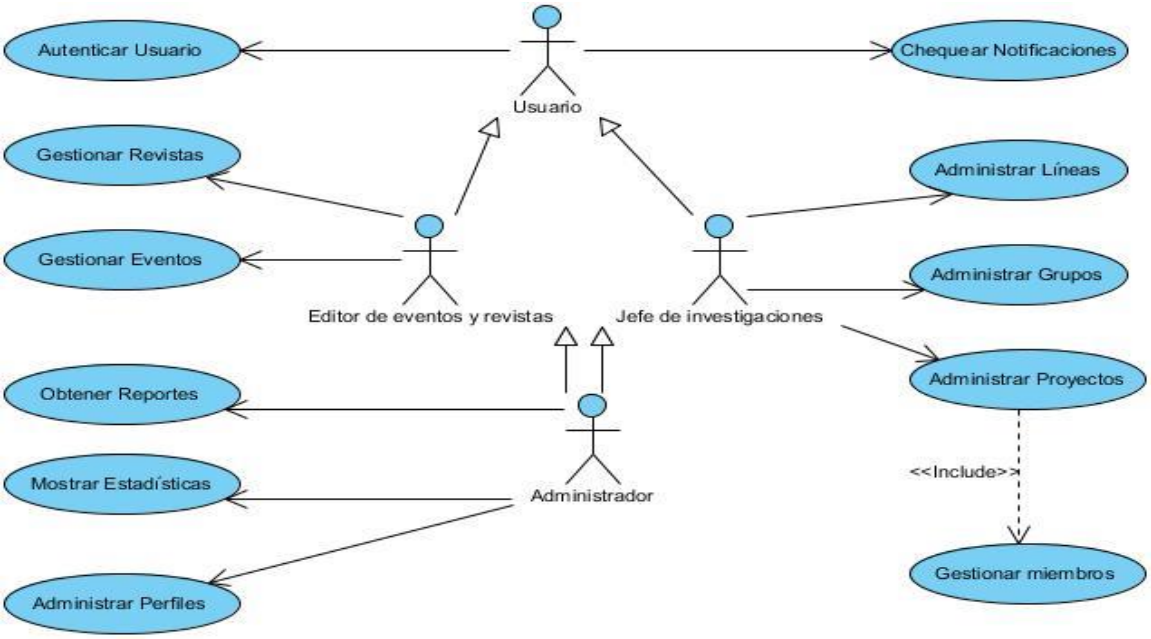


Figura 2 Diagrama de casos de uso del sistema

2.4.3 Descripción de casos de uso del sistema

Una vez modelado el diagrama de CUS se describen cada uno de los casos de uso que integran dicho diagrama. La Tabla 4 muestra la descripción del CU “Administrar Proyectos”, las demás descripciones se encuentran en el expediente de proyecto.

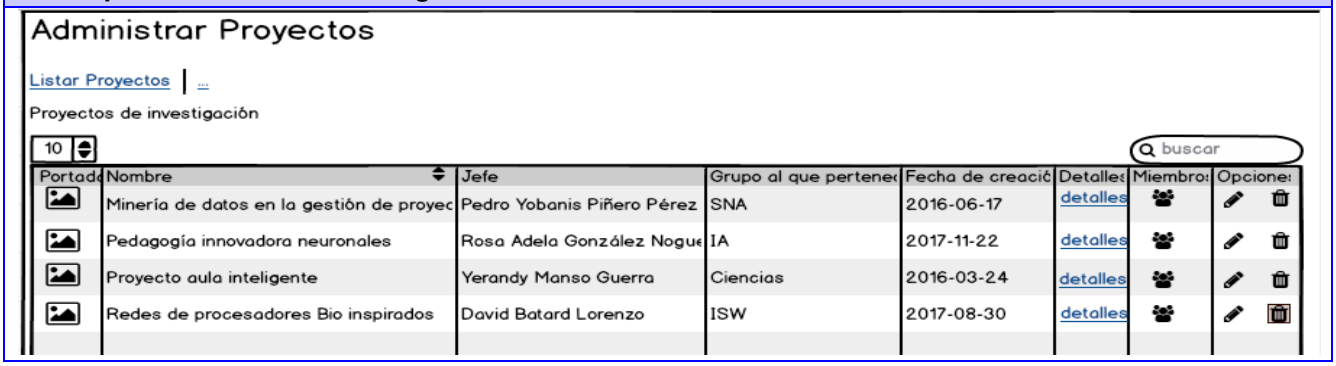
Tabla 4 Descripción del CUS "Administrar proyectos"

Objetivo	Permite listar las proyectos, crear un nuevo proyecto, mostrar proyecto, modificar proyecto, eliminar proyecto y ver detalles del mismo.
Actores	Jefe de investigaciones (Inicia).
Resumen	El caso de uso inicia cuando el jefe de investigaciones selecciona la opción ‘’Administrar Proyectos” del menú lateral izquierdo. La aplicación muestra el listado de proyectos del sistema y brinda la opción de insertar uno nuevo, de editar, eliminar ver detalles, listar sus miembros, listar solicitudes de membrecía, asignar miembro y eliminar miembro.
Complejidad	Media
Prioridad	Alta
Precondiciones	El usuario se ha autenticado en el sistema. El usuario posee el rol de ‘Administrador’ o ‘Jefe de Investigaciones’.
Postcondiciones	Se crea, edita, elimina o muestra un proyecto de investigación.
Referencia	RF28. Crear proyecto. RF29. Modificar proyecto. RF30. Eliminar proyecto. RF31. Ver datos de proyecto.
Flujo de eventos	
Flujo básico “Listar proyectos”	
	Actor / **Sistema**

1.	Selecciona la opción “Administrar Proyectos” del menú lateral izquierdo.	
2.		Muestra el listado de proyectos de investigación existentes en el sistema Permite realizar varias acciones: - Crear proyecto: Ver Sección 1: Crear proyecto - Editar proyecto: Ver Sección 2: proyecto - Eliminar proyecto: Ver Sección 3: Eliminar proyecto - Ver detalles de proyecto: Ver Sección 4: Mostrar proyecto
3.		
Sección 1: “Crear proyecto”		
Flujo básico Crear proyecto		
	Actor	**Sistema**
1.	Selecciona la opción “Agregar proyecto” de la página principal de Administrar proyectos de Investigación.	
2.		Muestra una vista con un formulario de registro con los siguientes campos: - Foto de Portada - Título - Jefe de Proyecto - Descripción - Contenido - Grupo al que pertenece - Aceptado
3.	Introduce los datos correspondientes y da clic en el botón “Crear proyecto”.	
4.		Crea un nuevo proyecto, almacena en la base de datos la información de dicho proyecto y la muestra en el listado de proyectos.
5.		Termina el caso de uso .
Flujos alternos		
3a Deja campos obligatorios vacíos		
	Actor	**Sistema**
3ª		No se inserta el proyecto y señala los campos que aún estén vacíos, para que el usuario los llene.
3b Inserta datos inválidos		
	Actor	**Sistema**
3 b		No inserta el proyecto y señala los campos inválidos.
3c El usuario selecciona la opción cancelar		
	Actor	Sistema
3 c		Cancela la operación.
Sección 2: “Modificar proyecto”		

Flujo básico		
	Actor	**Sistema**
1	Selecciona la opción “Modificar” del proyecto que desea modificar, en la página principal de Administrar proyectos.	
2		Muestra una vista con un formulario con los siguientes campos para editar: - Foto de Portada - Título - Jefe de Proyecto - Descripción - Contenido - Grupo al que pertenece - Activado
3	Modifica los datos que desee y da clic en el botón “Modificar proyecto”.	
4		Modifica el proyecto y guarda la información modificada en la base de datos.
5		Termina el caso de uso
Flujos alternos		
3a. Deja campos obligatorios vacíos		
	Actor	**Sistema**
3ª		No se inserta el proyecto y señala los campos que aún estén vacíos, para que el usuario los llene.
3b. Inserta datos inválidos		
	Actor	**Sistema**
3 b		No inserta el proyecto y señala los campos inválidos.
4a. El usuario selecciona la opción “Cancelar”		
	Actor	**Sistema**
4ª		Cancela la operación.
Sección 3: ‘’Eliminar proyecto”		
Flujo básico: Eliminar proyecto		
	Actor	**Sistema**
1	Selecciona la opción “Eliminar” del proyecto que desea eliminar, en la página principal de Administrar proyectos.	
2		Muestra un mensaje para que el usuario confirme la eliminación del proyecto seleccionado.
3	Selecciona la opción “Eliminar”.	
4		Elimina el proyecto seleccionado de la base de datos.
		Termina el caso de uso.
Flujos alternos		
3ª. El usuario selecciona la opción “Cancelar”		
	Actor	**Sistema**
3ª.		Cancela la operación.
Sección 4: “Ver detalles de proyecto”		

Flujo básico: Ver detalles de proyecto		
	Actor	**Sistema**
1	Selecciona la opción “Ver proyecto” del proyecto que desea visualizar, en la página principal de Administrar proyectos.	
2		Muestra una vista con todos los datos del proyecto de investigación.
3		Termina el caso de uso.
Relaciones	**CU incluidos**	Gestionar miembros: Ver CU Gestionar miembros.
Prototipo elemental de interfaz gráfica de usuario		



2.5 Patrón arquitectónico

La arquitectura utilizada para el modelado del sistema, está basada en el Modelo Vista Plantilla (MVT del inglés: *Model View Template*), según define Django como *framework* de desarrollo. Esto posibilita la organización del sistema en paquetes individuales, sin dependencias entre el modelo y las plantillas (Kaplan-Moss, 2013). A continuación, se presenta una breve descripción de esta arquitectura. Además, la misma se evidencia mediante el diagrama de paquetes del sistema, el cual se especifica más adelante.

Modelo Vista Plantilla

El patrón de diseño MVT, es una especificación del patrón Modelo Vista Controlador (MVC), donde (Alchin, 2013):

- M (Modelo) es la capa de acceso a la base de datos. Esta capa contiene toda la información sobre los datos; cómo acceder a estos, cómo validarlos, cuál es el comportamiento que tiene, y las relaciones entre los datos.
- T (Plantilla) es la capa de presentación. Esta capa contiene las decisiones relacionadas a la presentación; cómo algunas cosas son mostradas sobre una página web u otro tipo de documento.
- V (Vista) es la capa de la lógica de negocios. Esta capa contiene la lógica que accede al modelo y la delega a la plantilla apropiada. Es el puente entre el modelo y las plantillas.

2.6 Diagrama de Paquetes

El diagrama de paquetes se diseña con el objetivo de mostrar la organización que presentan los paquetes con los que se trabaja en el desarrollo de la aplicación (Pressman, 2010), poniendo en evidencia la utilización de la arquitectura mencionada anteriormente. Este diagrama contiene un total de cuatro paquetes. A continuación, se presenta como quedó conformado el diseño del diagrama de paquetes de la aplicación.

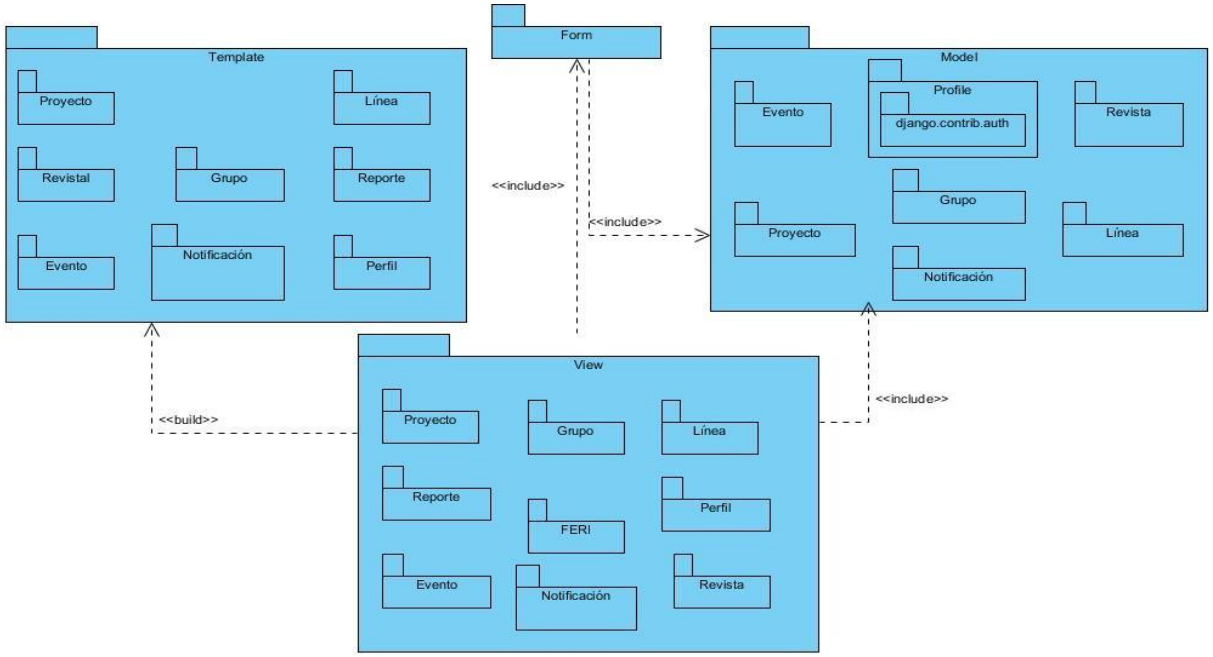


Figura 3 Diagrama de paquetes del sistema

Descripción del diagrama de Paquetes del Sistema

En el diagrama se representan los paquetes Modelo, Vista y Plantilla, siguiendo la arquitectura MVT, descrita en el epígrafe 2.5. En el mismo se evidencia que el paquete Vista, funciona como intermediario entre los paquetes Plantilla y Modelo; este contiene la lógica del negocio y todos los métodos a ejecutar en la aplicación. A su vez interactúa con el paquete *Form* para introducir datos al modelo mediante la clase *django.forms* que provee el *framework* Django y valerse del mismo en la construcción de la Plantilla. Por otra parte, el Modelo representa la capa de acceso a datos y se utiliza para crear las tablas en la base de datos, así como el trabajo con las mismas, permitiendo realizar consultas y gestionar los datos existentes. El paquete Plantilla contiene las plantillas que verá el usuario, siendo esta la capa de presentación.

Paquetes representados en el diagrama

- **Vista**: este paquete contiene la lógica del negocio, contiene los métodos de la aplicación.
- **Plantilla**: es la capa de presentación de la aplicación, contiene las plantillas que se construirán con la vista.
- **Modelo**: este paquete es la capa de acceso a datos, define la estructura de las tablas y permite el trabajo con las mismas.
- **Form**: contiene los formularios de algunos de los modelos.
- **Django.contrib.auth**: representa el paquete que tiene el modelo de usuarios y grupos del sistema.
- **Perfil**: este paquete se encarga de la autenticación y la gestión de perfiles de usuario.
- **Proyecto**: en este paquete se definen las funcionalidades relacionadas con los proyectos de investigación que el sistema permitirá gestionar.
- **Grupo**: en este paquete se definen las funcionalidades relacionadas con los grupos de investigación que el sistema permitirá gestionar.
- **Línea**: en este paquete se definen las funcionalidades relacionadas con las líneas de investigación que el sistema permitirá gestionar.
- **Revistas**: en este paquete se definen las funcionalidades relacionadas con las revistas que el sistema permitirá gestionar.
- **Eventos**: en este paquete se definen las funcionalidades relacionadas con los eventos que el sistema permitirá gestionar.
- **Notificaciones**: en este paquete se definen las funcionalidades relacionadas con las notificaciones del sistema.
- **FERI**: en este paquete se definen los servicios que brindará el sistema.

2.7 Diagrama de Clases del Diseño

El diagrama de clases del diseño constituye la representación de las clases del sistema, la estructura y organización de las mismas (Pressman, 2010). A continuación, se muestra el diagrama de clases del diseño para el caso de uso Administrar Proyectos, el cual muestra las relaciones existentes entre las clases principales. Los diagramas de clases del diseño de los restantes casos de uso se encuentran en el expediente de proyecto.

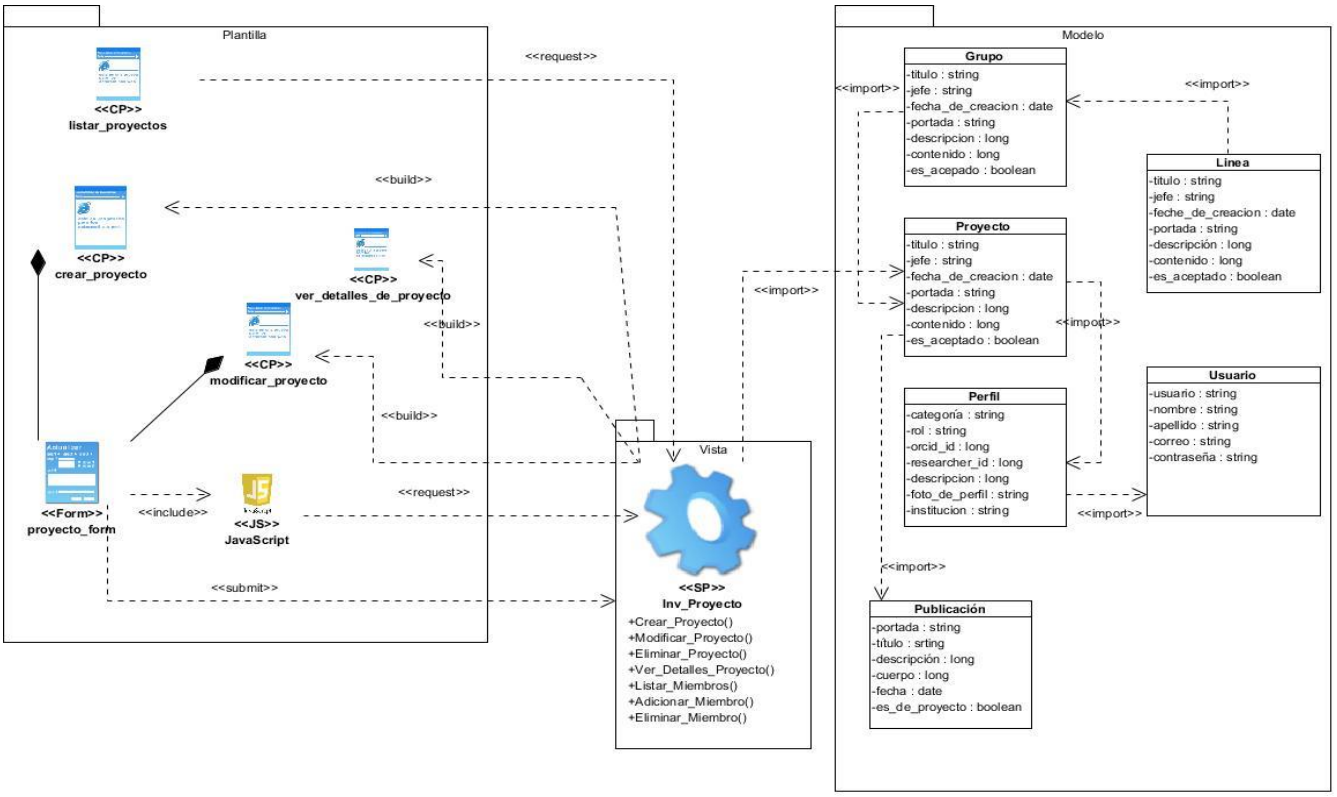


Figura 4 Diagrama de clases del diseño para el CUS "Administrar Proyecto"

Descripción del diagrama de clases del diseño del caso de uso Administrar Proyecto

El caso de uso Administrar Proyectos comienza cuando el usuario selecciona la opción administrar proyectos del menú lateral izquierdo. Esto envía una petición a la página servidor (SP, del inglés: *server page*) Inv_Proyecto, que se encarga de consultar el modelo Proyecto y obtener todos los proyectos de investigación del sistema, los cuales se muestran mediante la página cliente (CP, del inglés: *client page*) listar_proyectos. Si el usuario selecciona la opción adicionar proyecto, se ejecuta la función de la SP Crear_Proyecto, la cual se encarga de construir la CP crear_proyecto. Esta contiene un formulario (proyecto_form) que realiza validaciones usando JavaScript, si los datos del formulario son correctos estos son enviados al SP, el cual se encarga de guardarlos en la base de datos haciendo uso del modelo Proyecto. Por otro lado, si se selecciona la opción modificar de un proyecto específico se envía una petición a la SP, la cual accede a la base de datos mediante el modelo y devuelve un formulario con los atributos de dicho proyecto, así como los de sus clases relacionadas. Si estos son modificados se actualizan en la base de datos, de la misma manera que en crear proyecto. Si se selecciona la opción eliminar de un proyecto de la lista, en CP listar_proyectos se muestra una ventana modal en donde el usuario confirma la eliminación, esta una vez que es confirmada manda esta confirmación al SP, el cual ejecuta la funcionalidad Eliminar_Proyecto, que mediante el modelo, elimina el proyecto de la base de datos. Por último, si se selecciona la opción ver detalles de un proyecto en CP listar_proyectos, se envía una petición a la SP, la cual mediante la funcionalidad Ver_Detalles_Proyecto

consulta el modelo y obtiene todos los datos necesarios para construir la CP ver_detalles, donde se muestran todos los atributos del proyecto seleccionado, así como los de sus clases relacionadas.

2.8 Modelo de datos

El modelo de datos se utiliza para describir la estructura lógica y física de la información persistente gestionada por el sistema (Tecnologías-Información, 2018). A partir de las clases que se representan se puede diseñar el modelo de datos, el cual permite relacionar de forma estructural las tablas que persisten en una base de datos, atendiendo a su tipo y las condiciones que deben cumplir para reflejar la realidad deseada.

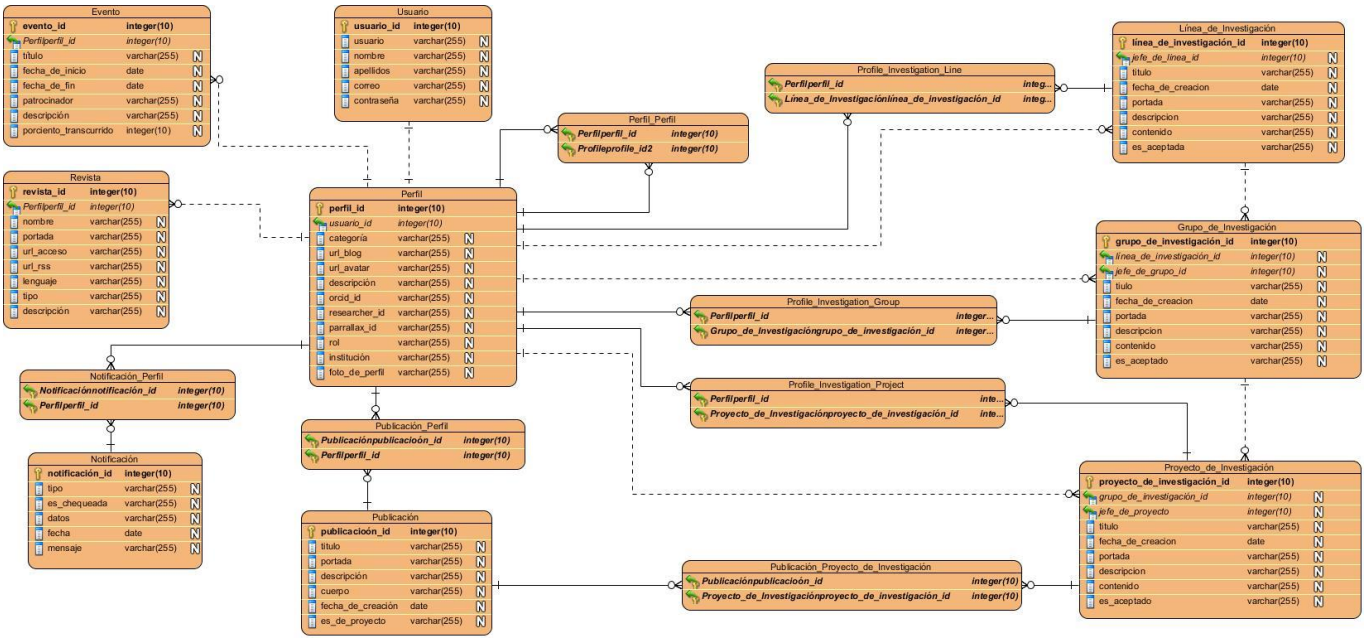


Figura 5 Modelo de datos

El modelo de datos obtenido cuenta con un total de 16 tablas. Se encuentra en tercera forma normal ya que no existen atributos atómicos, las tablas poseen llave primaria y no existen dependencias parciales entre los atributos no primos y la llave primaria, lo que conlleva a la eliminación de anomalías en la actualización de la información.

2.9 Patrones de diseño de software

Los patrones de diseño son descripciones de clases y objetos relacionados que están particularizados para resolver un problema de diseño general en un determinado contexto. Estos a su vez identifican las clases, instancias, roles, colaboraciones y la distribución de responsabilidades (Gamma, 2008). Entre los patrones de diseño más utilizados se encuentran los Patrones Generales de Software para

Asignación de Responsabilidades (GRASP, del inglés: *General responsibility assignment software patterns*) y los patrones Pandilla de los Cuatro (GoF, del inglés: *Gang of four*).

2.9.1 Patrones GRASP

Los patrones de asignación de responsabilidades describen los principios fundamentales del diseño de objetos para la asignación de responsabilidades (Gamma, 2008). Teniendo en cuenta el patrón arquitectónico seleccionado, los patrones de diseño que se aplican durante el desarrollo del sistema NOX-Admin son los expuestos a continuación.

Bajo acoplamiento

El acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con que las conoce y con que recurre a ellas. Una clase con bajo acoplamiento no depende de muchas otras. El bajo acoplamiento soporta el diseño de clases más independientes, que reducen el impacto de los cambios, que son más reutilizables y más fáciles de entender (Larman, 1999).

La clase *User* (Usuario) de Django no depende de otras clases en el sistema. Espera sólo valores nativos del lenguaje Python, lo que evita conocer las implementaciones de otras clases.

Alta cohesión

En la perspectiva del diseño orientado a objetos, la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas, que no realicen un trabajo enorme y que además colaboran con otras clases para llevar a cabo las tareas. Una clase con baja cohesión hace muchas cosas no afines o un trabajo excesivo. Una clase con mucha cohesión es útil porque es bastante fácil darle mantenimiento, entenderla y reutilizarla, además a menudo genera un bajo acoplamiento (Larman, 1999).

En la Figura 6, se ilustra como la clase “Proyecto” delega la responsabilidad de visualizar proyectos a la clase *Project_List*, evidenciando el uso del patrón Alta Cohesión en el sistema.

```python
# Listar Proyectos
class Project_List(Base_ListView):
    model = Project
    template_name = 'dashboard/projects_t/project_list.html'

    @method_decorator(permission_required(['Administrador', 'Jefe de Investigaciones']))
    def dispatch(self, *args, **kwargs):
        return super(Project_List, self).dispatch(*args, **kwargs)

    def get_context_data(self, object_list=None, **kwargs):
        user = self.request.user
        profile = get_profile_by_userid(user)
        context = super(Project_List, self).get_context_data(**kwargs)
        context['context'] = self.context(profile)
        return context
```

Figura 6 Ejemplo del patrón GRASP: alta cohesión

Controlador

Este patrón se evidencia en el archivo views.py contenido dentro de cada paquete de la aplicación, el cual funciona como un controlador que atiende los distintos eventos del sistema. Esta clase recibe información proveniente del modelo y decide qué datos enviar a la plantilla para su posterior organización e interacción con el usuario, además de decidir qué funcionalidad se va a ejecutar para dar respuesta a determinada petición (Larman, 1999).

La clase *Project_Delete* (ver Figura 7) elimina un objeto de tipo Proyecto de la base de datos al haberse ejecutado una acción invocada por la interfaz gráfica. Sirve de mediador entre el modelo y las plantillas.

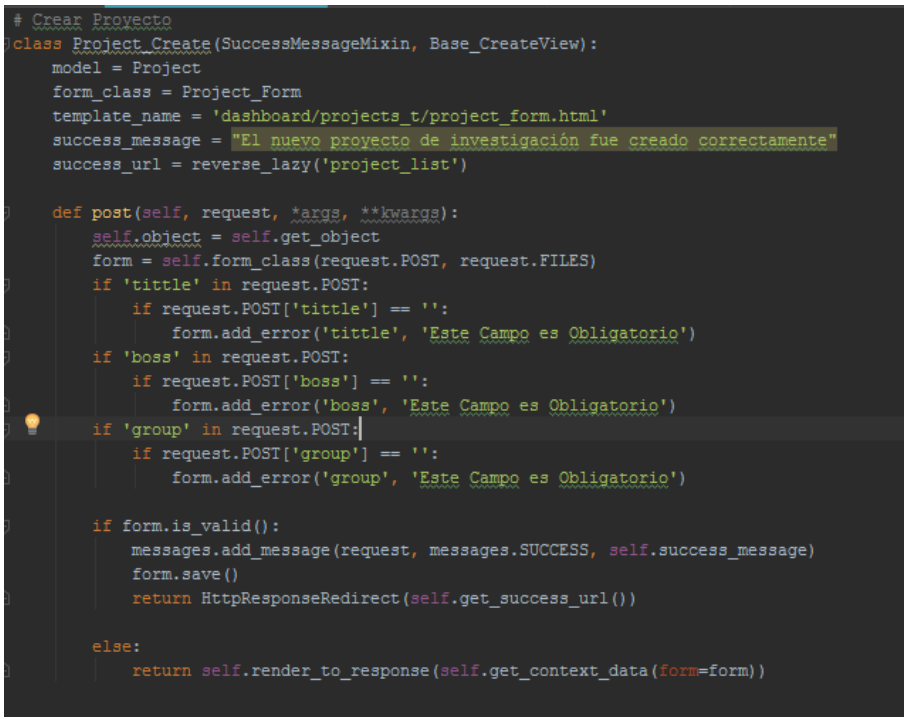
```python
# Eliminar Proyecto
class Project_Delete(DeleteView):
    model = Project
    template_name = 'dashboard/projects_t/project_delete.html'
    success_url = reverse_lazy('project_list')

    @method_decorator(permission_required(['Administrador', 'Jefe de Investigaciones']))
    def dispatch(self, *args, **kwargs):
        return super(Project_Delete, self).dispatch(*args, **kwargs)
```

Figura 7 Ejemplo del patrón GRASP: controlador

Experto

Consiste en asignar una responsabilidad al experto en información, es decir, la clase que cuenta con la información necesaria para cumplir la responsabilidad. Si esto se hace de forma adecuada, los sistemas tienden a ser más fáciles de entender, mantener y ampliar, y se presenta la oportunidad de reutilizar los componentes en futuras aplicaciones (Larman, 1999).

La clase *Project_Create* (ver Figura 8), es la encargada de manejar objetos de tipo Proyecto. El método *post* () adiciona un nuevo objeto de este tipo en la base de datos.



```python
# Crear Proyecto
class Project_Create(SuccessMessageMixin, Base_CreateView):
    model = Project
    form_class = Project_Form
    template_name = 'dashboard/projects_t/project_form.html'
    success_message = "El nuevo proyecto de investigación fue creado correctamente"
    success_url = reverse_lazy('project_list')

    def post(self, request, *args, **kwargs):
        self.object = self.get_object
        form = self.form_class(request.POST, request.FILES)
        if 'tittle' in request.POST:
            if request.POST['tittle'] == '':
                form.add_error('tittle', 'Este Campo es Obligatorio')
        if 'boss' in request.POST:
            if request.POST['boss'] == '':
                form.add_error('boss', 'Este Campo es Obligatorio')
        if 'group' in request.POST:
            if request.POST['group'] == '':
                form.add_error('group', 'Este Campo es Obligatorio')

        if form.is_valid():
            messages.add_message(request, messages.SUCCESS, self.success_message)
            form.save()
            return HttpResponseRedirect(self.get_success_url())

        else:
            return self.render_to_response(self.get_context_data(form=form))
```

Figura 8 Ejemplo del patrón GRASP: experto

2.9.2 Patrones GoF

Los patrones GoF son agrupados en tres grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. Los de creación abstraen el proceso de creación de instancias. Los estructurales se ocupan de cómo clases y objetos son utilizados para componer estructuras de mayor tamaño y los de comportamiento atañen a los algoritmos y a la asignación de responsabilidades entre objetos (Gamma, 2008). En el desarrollo del sistema NOX-Admin se hace uso de los patrones siguientes:

Decorador

Este patrón posibilita añadir responsabilidades adicionales a un objeto de forma dinámica. El sistema cuenta con la clase *django.contrib.auth.decorators* que contiene todas las funcionalidades adicionales o decoradores que realizarán determinada acción en correspondencia con el objeto ejecutado y sus entradas. Un ejemplo del uso de este patrón, lo constituye la clase Project_List de la clase views.py, que tiene la responsabilidad de listar todos los proyectos y cuenta con el decorador “login_required” que verifica si el usuario está autenticado en el sistema. A continuación, en la Figura 9 se muestra un ejemplo del uso de dicho patrón.

```
url(r'^list/$', login_required(Project_List.as_view()), name='project_list'),
url(r'^create/$', login_required(Project_Create.as_view()), name='project_create'),
```

Figura 9 Ejemplo de uso del decorador login_required

Observador

Define una dependencia “uno a muchos” entre objetos, para que cuando uno de ellos cambie su estado, todos los que dependan de él sean avisados y puedan actualizarse convenientemente. Este patrón se evidencia en la clase Grupo ya que cuando se realiza una modificación se actualizan los proyectos asignados al mismo. A continuación, en la Figura 10 se muestra un ejemplo del uso de dicho patrón.

```
class Project(models.Model):
    group = models.ForeignKey(Group, on_delete=models.CASCADE, blank=True)
```

Figura 10 Ejemplo de uso del patrón observador

Conclusiones del capítulo

A partir de los resultados presentados en este capítulo y teniendo en cuenta los objetivos específicos de la investigación, quedan reflejadas las bases ingenieriles y arquitectónicas para realizar la implementación del sistema NOX-Admin, resaltando que:

- ✓ La especificación de los requisitos funcionales y no funcionales contribuyó a una mejor organización del sistema a partir de las características y tecnologías previamente seleccionadas.
- ✓ La descripción de CUS y los diagramas de clases de diseño a partir del lenguaje de modelado UML orientaron efectivamente la implementación del sistema, su documentación y comportamiento acorde a los requisitos.
- ✓ El uso del patrón arquitectónico MVT a partir del marco de trabajo Django y la aplicación de patrones de diseño GRASP y GoF, evidencian la utilización de buenas prácticas durante el desarrollo del sistema, lo cual repercute positivamente en la calidad final del mismo.

Capítulo 3: Implementación y pruebas

En el presente capítulo se muestran los resultados alcanzados en la implementación del sistema, así como los estándares de codificación empleados y algunas interfaces de la aplicación. Se evidencia la organización del sistema mediante el diagrama de componentes el cual representa la vista estática del mismo, y mediante el diagrama de despliegue se representa cómo se distribuirán los servicios que brindará el sistema. Por último, se describirán las pruebas realizadas al sistema y los resultados obtenidos una vez aplicadas las mismas; además de la validación de la investigación.

3.1 Modelo de implementación

Entre las disciplinas propuestas por la metodología de desarrollo seleccionada, AUP-UCI, se encuentra la de Implementación. En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema. El objetivo de esta disciplina es lograr la implementación de las clases y subsistemas que fueron encontrados durante el diseño, así como definir la organización del código. (Sánchez Rodriguez, 2014).

El modelo de implementación está conformado por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Somerville, 2011). El modelo de implementación muestra, a través del diagrama de despliegue, la estructura del sistema en ejecución y a través de los diagramas de componentes, las dependencias entre las partes de código del sistema.

3.1.1 Diagrama de componentes

El término de componente se define como una parte modular, desplegable y reemplazable de un sistema que encapsula implementación y expone un conjunto de interfaces (Pressman, 2010). Los diagramas de componentes describen los elementos físicos del sistema y sus relaciones. Los componentes representan todos los tipos de elementos de software que entran en la fabricación de aplicaciones informáticas. Pueden ser simples archivos, paquetes y bibliotecas cargadas dinámicamente (Jacobson, y otros, 2000). La Figura 11 muestra el diagrama de componentes para el caso de uso “Administrar Proyectos”.

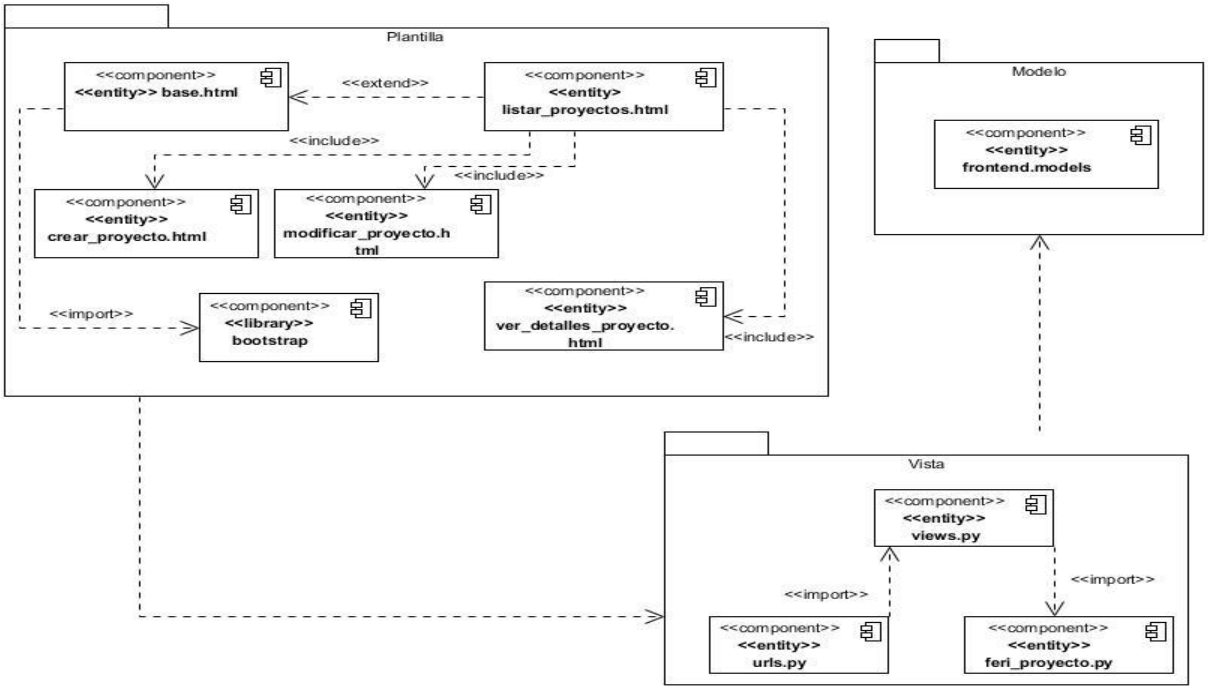


Figura 11 Diagrama de componentes para el CUS "Administrar Proyectos"

Descripción del diagrama de componentes para el CUS Administrar Proyectos

En el diagrama de componentes mostrado anteriormente se evidencia que el componente urls.py contiene los métodos del view.py que se ejecutan según una url específica. views.py utiliza *frontend.models* para la administración de proyectos a nivel de base de datos. Además, views.py para ejecutar sus métodos se apoya en feri_proyecto.py, esta contiene métodos útiles para administrar proyectos como obtener_publicaciones (id_project), el cual devuelve todas las publicaciones de un proyecto específico pasado por parámetro. A su vez, views.py es la encargada de construir listar_proyectos.html. Esta importa a base.html para su completa construcción y visualización por el usuario e incluye a crear_proyecto.html, modificar_proyecto.html y ver_detalles_proyecto.html. El componente base.html contiene lo general de las plantillas, o sea, el contenido que no varía de unas a otras, por lo que importa la librería de *bootstrap* utilizada para la visualización del sistema.

Componentes representados en el diagrama

- **base.html**: plantilla que contiene la estructura básica de todas las plantillas.
- **listar_proyecto.html**: plantilla que contiene el listado de proyectos de la red de investigadores.
- **crear_proyecto.html**: plantilla que contiene el formulario necesario para la creación de un proyecto.
- **modificar_proyecto.html**: plantilla que contiene el formulario necesario para la modificación de un proyecto.

- **ver_detalles_proyecto.html**: plantilla que contiene los detalles de un proyecto, así como el listado de sus publicaciones.
- **bootstrap**: librería CSS
- **views.py**: clase que contiene los métodos para la administración de proyectos.
- **urls.py**: clase que contiene las url de administrar proyecto y los métodos que ejecutan las mismas.
- **feri_proyecto.py**: clase auxiliar que contiene métodos útiles para la administración de proyectos.
- **frontend.models**: contiene los modelos de Proyecto, Perfil y Publicación.

Una vez especificado el diagrama de componentes, se define el diagrama de despliegue con el objetivo de visualizar como se desplegará el sistema desarrollado.

3.1.2 Diagrama de despliegue

Un diagrama de despliegue permite indicar cómo se ubicarán las funcionalidades y los subsistemas dentro del entorno computacional físico que soportará el software (Pressman, 2010). La vista de despliegue muestra la configuración de los nodos de procesamiento en tiempo de ejecución, los vínculos de comunicación entre ellos y las instancias de los componentes. El propósito del modelo de despliegue es capturar la configuración de los elementos de procesamiento y las conexiones entre estos elementos en el sistema (Pressman, 2010). A continuación se muestra en la Figura 12 el diagrama de despliegue diseñado para la aplicación, en el cual se evidencia cómo se distribuyen los servicios que brindará el sistema teniendo en cuenta los requisitos no funcionales de software identificados en epígrafes anteriores, destacándose que la comunicación entre los nodos PC_Cliente y Servidor Web NOX-Admin será mediante el protocolo HTTPS, y la comunicación entre los nodos Servidor Web NOX-Admin y Servidor de Base de Datos será mediante el protocolo TCP/IP.

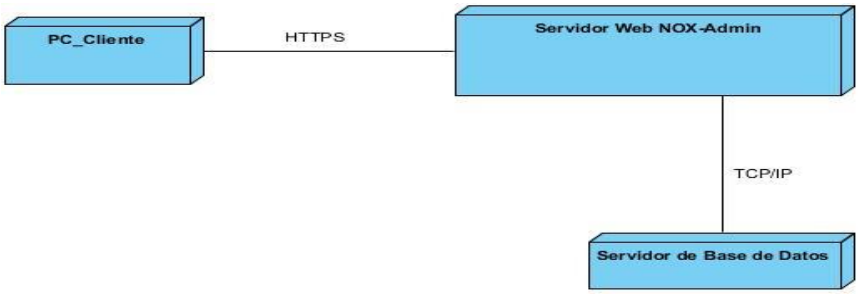


Figura 12 Diagrama de despliegue

PC Cliente: computadora en la cual la aplicación se ejecutará a través de un navegador, en este caso Mozilla Firefox 57.0 o superior.
Servidor Web NOX-Admin: radica la lógica de negocio de la aplicación, utilizando el lenguaje de programación del lado del servidor Python en su versión 3.5.
Servidor de Base de datos: servidor de Base de Datos PostgreSQL 9.4, donde se encuentra la base de datos que utiliza el sistema, este puede estar instalado en la misma computadora donde se encuentra el servidor Web.

3.2 Código Fuente

Para obtener una versión funcional de la aplicación se deben implementar los componentes que se han definido, como resultado se obtienen archivos que contienen el código fuente de la aplicación. El código fuente de un programa es un conjunto de líneas de texto que son las instrucciones que debe seguir la computadora para ejecutar dicho programa. Por tanto, en el código fuente de un programa está escrito su funcionamiento. Estas instrucciones son escritas en un lenguaje de programación que consiste en un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Martínez Mengual, 2016 pág. 43).

3.2.1 Estándares de Codificación

Un estándar de codificación completo comprende todos los aspectos de la generación de código. Si bien los programadores deben implementar un estándar de forma prudente, éste debe tender siempre a lo práctico. Un código fuente completo debe reflejar un estilo armonioso, como si un único programador hubiera escrito todo el código de una sola vez (Martínez Mengual, 2016 pág. 54).
Los estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código (ERP, 2008). Para facilitar el entendimiento del código y establecer un modelo a seguir, se establecieron los estándares de codificación mostrados a continuación para el lenguaje de programación Python en el *framework* de desarrollo Django.

Estructura

- Las líneas podrán tener ochenta caracteres o menos.
- Las funciones de alto nivel y definiciones de clases se separan con dos líneas como se muestra en la Figura 13.

```python
def member_delete(request, id_miembro, id_project):
    user = User.objects.get(id=id_miembro)
    print(user.first_name)
    project = Project.objects.get(id=id_project)
    if request.method == 'POST':
        project.member.remove(user)
        return redirect('project_member_list', project.id)
    return render(request, 'dashboard/projects_t/project_member_delete.html', {'user': user})


class Project_List(ListView):
    model = Project
    template_name = 'dashboard/projects_t/project_list.html'
```

Figura 13 Ejemplo de estándar de codificación: separación entre declaraciones de funciones y definiciones de clases

- Se añaden comentarios descriptivos junto a cada declaración de variables, si es necesario como se muestra en la Figura 14.

```python
class Project_Members(ListView):
    def get(self, request, id_project):
        project = Project.objects.get(id=id_project) #Almacena el proyecto según el id
        miembros = get_profile_members(project)  # Almacena los miembros del proyecto
        cantidad = len(miembros)  # Almacena la cantidad de miembros del proyecto
        nombre = project.tittle  # Almacena el nombre del proyecto
        id_project = project.id  # Almacena el id del proyecto
        return render(request, 'dashboard/projects_t/project_member_list.html',
                      {'miembros': miembros, 'cantidad': cantidad, 'nombre': nombre, 'id_project': id_project})
```

Figura 14 Ejemplo de estándar de codificación: comentarios descriptivos

Importaciones

- Las importaciones siempre estarán colocadas al comienzo del archivo y deberán estar agrupadas por (ver Figura 15):
  1. Importaciones de biblioteca estándar.
  2. Importaciones terceras relacionadas.
  3. Importaciones locales de la aplicación.

```python
from django.shortcuts import render, redirect
from django.urls import reverse_lazy
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.contrib.messages.views import SuccessMessageMixin
from django.views.generic import ListView, CreateView, UpdateView, DeleteView
from apps.dashboard.Inv_Project.forms import Project_Form
from Red_Investigacion.FERI.collections import *
```

Figura 15 Ejemplo de estándar de codificación: orden de las importaciones

Nombres de clases y funciones

- Se utiliza para el nombramiento de funciones letras minúsculas separando las palabras con guiones bajos, como se muestra en la Figura 16.

```
def count_journals():
    return Register_Journal.objects.count()
```

Figura 16 Ejemplo de estándar de codificación: nombramiento de funciones

- Las clases están representadas por palabras que comienzan con mayúsculas, si contiene más de una palabra estas deberán estar separadas por guiones bajos, como se muestra en la Figura 17.

```
class Project_Delete(DeleteView):
    model = Project
    template_name = 'dashboard/projects_t/project_delete.html'
    success_url = reverse_lazy('project_list')
```

Figura 17 Ejemplo de estándar de codificación: nombramiento de clases

3.3 Resultados de la implementación

El sistema implementado, mediante las interfaces que presenta permite realizar una serie de funciones que cumplen con los requisitos especificados en el Capítulo 2. A partir de estos se realiza un análisis del cual cuyos resultados se muestran a continuación.

- ✓ El sistema debe contar con una interfaz amigable para el usuario, con un diseño sencillo que permita el balance adecuado entre funcionalidad y simplicidad; logrando así que no resulte engorrosa la utilización del sistema para el usuario final. Este requisito se refleja en la Figura 18, la cual representa la interfaz de “Administrar Proyectos”, donde se visualizan algunos íconos, botones e hipervínculos que permiten realizar diferentes acciones sobre los perfiles. Estos deben ser sugerentes y entendibles para el usuario. Del mismo modo, las etiquetas de cada funcionalidad y los campos de cada interfaz deben tener títulos asociados a su función de negocio.

Figura 18 Interfaz del CUS “Administrar Proyectos"

- ✓ Además, los mensajes de error del sistema deben incluir una descripción textual del error como se muestra en la Figura 19, la cual representa un error cuando se introducen campos inválidos en el formulario para crear un nuevo perfil.

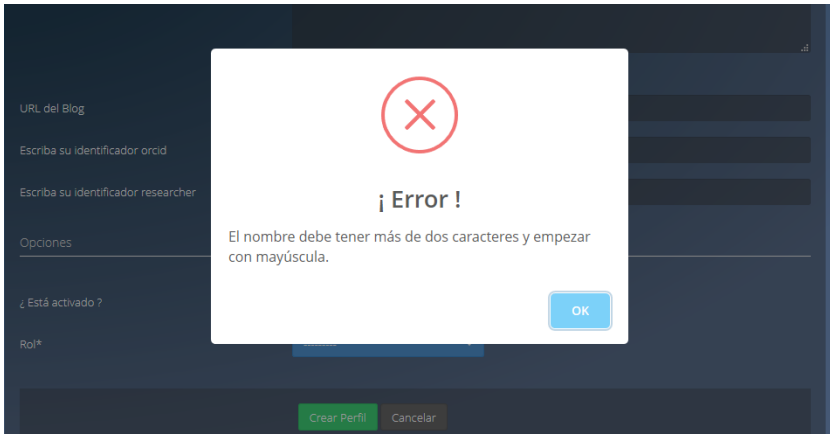


Figura 19 Error del sistema cuando se insertan campos inválidos

También en la Figura 20 se muestra un error cuando un usuario que no posee permisos para acceder al sistema intenta hacerlo.

Figura 20 Error del sistema cuando intenta acceder un usuario no autorizado

- ✓ El sistema verifica que el usuario esté autenticado antes de que pueda realizar alguna acción sobre el mismo. La Figura 21 muestra la interfaz de autenticación en el sistema para los usuarios.

Figura 21 Interfaz de autenticación de NOX-Admin

- ✓ Asimismo, se establecen roles para acceder al sistema, garantizando que la información almacenada solo pueda ser modificada y/o visualizada por los usuarios autorizados. La Figura 22 muestra cómo se asignan los roles a los usuarios.

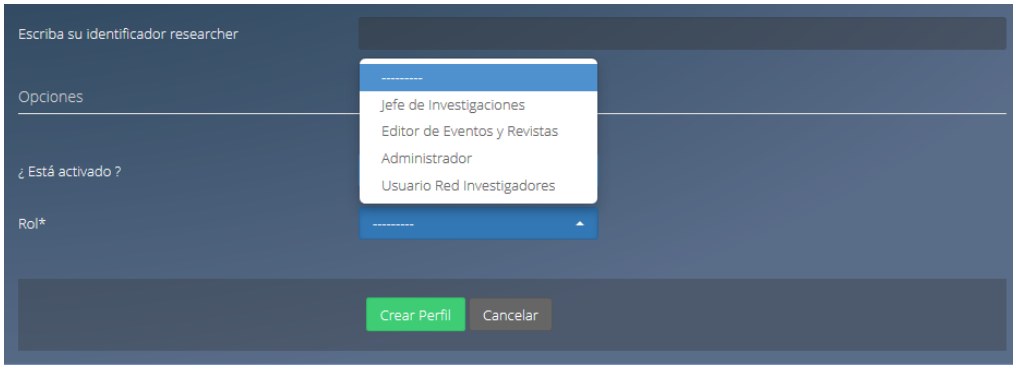


Figura 22 Interfaz para definir roles de usuarios

3.5 Pruebas de software

Una vez terminada la implementación del producto que se requiere es necesario realizarle pruebas con el objetivo de detectar errores en la aplicación y la documentación (Suárez, y otros). Las pruebas son un conjunto de actividades que se planean con anticipación y se realizan de manera sistemática, por lo que se debe definir una estrategia para las pruebas de software que se realizarán, la cual se basa en un conjunto de pasos y procedimientos en que se puedan incluir técnicas y métodos específicos del diseño de casos de prueba (Pressman, 2010).

Estrategia de pruebas

Una estrategia de pruebas de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuándo se planean y se llevan a cabo dichos pasos, y cuánto esfuerzo, tiempo y recursos requieren. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados (Pressman, 2010). Según la metodología seleccionada para el diseño e implementación del sistema, las pruebas que se realizan son (Sánchez Rodriguez, 2014):

- ✓ Pruebas internas
- ✓ Pruebas de liberación
- ✓ Pruebas de aceptación

En la presente investigación se realizan pruebas internas y de aceptación. Las pruebas de liberación no proceden, ya que se necesita para la realización de las mismas más tiempo del asignado para el desarrollo del sistema. Estas constituyen servicios externos que se crean con la dirección de calidad y seguridad informática de la universidad, los cuales serán pedidos en un futuro por la dirección del proyecto SNA para la certificación de la calidad del sistema y su puesta en explotación. A continuación, se muestran las pruebas realizadas al sistema y los resultados obtenidos por cada una.

3.5.1 Pruebas unitarias

Las pruebas de unidad o unitarias son el proceso de probar componentes del programa tales como métodos o clases de objetos. Las funciones o los métodos individuales son el tipo más simple de componente. Las pruebas deben llamarse para dichas rutinas con diferentes parámetros de entrada (Somerville, 2011).

Prueba con Unittest

Las pruebas a nivel de unidad se realizan basadas en el ambiente de ejecución Unittest. Este permite la escritura de pruebas en forma de funciones simples con el objetivo de detectar errores de implementación. En la siguiente imagen se puede visualizar los resultados de las pruebas realizadas a

los diferentes módulos, las cuales analizan un total de 824 líneas de código, retornando para el 93% de las mismas, resultados satisfactorios.

Module ↓	statements	missing	excluded	coverage
apps\dashboard\Event\forms.py	8	0	0	100%
apps\dashboard\Event\views.py	80	0	0	100%
apps\dashboard\Inv_Group\forms.py	18	0	0	100%
apps\dashboard\Inv_Group\views.py	85	14	0	84%
apps\dashboard\Inv_Line\forms.py	8	0	0	100%
apps\dashboard\Inv_Line\views.py	85	4	0	94%
apps\dashboard\Inv_Project\views.py	145	9	0	94%
apps\dashboard\Journal\forms.py	8	0	0	100%
apps\dashboard\Journal\views.py	72	14	0	81%
apps\dashboard\Profile\forms.py	16	0	0	100%
apps\dashboard\Profile\views.py	127	11	0	92%
apps\frontend\models.py	172	13	0	93%
Total	**824**	**65**	**0**	**93%**

coverage.py v4.5.3, created at 2019-05-01 18:15

Figura 23 Resultados de las pruebas unitarias con el ambiente de ejecución Unittest

Estas pruebas fueron aplicadas a cada método de considerable importancia una vez finalizada su implementación. Como consecuencia de esta estrategia de aplicación de las pruebas unitarias, no se tiene un control de las mismas en cuanto a cantidad de iteraciones. Cada uno de estos métodos fue probado al implementarse o modificarse. En total se detectaron 19 no conformidades. Todas fueron corregidas en el momento de su detección.

Prueba de caja de blanca

La prueba de caja blanca también se conoce como prueba de caja transparente o de cristal. Esta prueba consiste específicamente en cómo diseñar los casos de prueba atendiendo al comportamiento interno y la estructura del programa, examinándose la lógica interna sin considerar los aspectos de rendimiento. Dentro de la prueba de caja blanca se incluyen las técnicas de pruebas que serán descritas a continuación (Pressman, 2010):

- ✓ Prueba del camino básico: permite obtener una medida de la complejidad lógica de un diseño y usar la misma como guía para la definición de un conjunto de caminos básicos.
- ✓ Prueba de condición: ejercita las condiciones lógicas contenidas en el módulo de un programa. Garantiza la ejecución por lo menos una vez de todos los caminos independientes de cada módulo, programa o método.
- ✓ Prueba de flujo de datos: Se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa. Garantiza que se ejerciten las estructuras internas de datos para asegurar su validez.

- ✓ Prueba de bucles: Se centra exclusivamente en la validez de las construcciones de bucles. Garantiza la ejecución de todos los bucles en sus límites operacionales.

La técnica de caja blanca empleada en la solución desarrollada fue la Prueba del Camino Básico, la cual permite obtener una medida de la complejidad lógica del diseño procedimental y usar esa medida como guía para la definición de un conjunto básico de caminos de ejecución, garantizando con estos que durante la prueba se ejecute por lo menos una vez cada sentencia del programa (Pressman, 2010).

Para realizar esta técnica es necesario calcular antes la complejidad ciclomática del algoritmo o fragmento de código a analizar. Para ello se selecciona el método last_logged_users (), el cual retorna los últimos cinco usuarios registrados en el sistema.

A continuación, se enumeran las sentencias de código del procedimiento realizado sobre el método last_logged_users () (ver Figura 24) y el grafo de flujo asociado al mismo (ver Figura 25).

```
# Devuelve los 5 últimos usuarios logeados en el sistema
def last_logged_users():
    usuarios = User.objects.filter(is_superuser=False)                #1
    fechas = []
    ultimos_users = []
    profiles = []

    for user in usuarios:                                             #2
        fechas.append(user.date_joined.isoformat())                   #3

    ultimas_fechas = sorted(fechas, reverse=True)[:5]                 #4

    for fecha in ultimas_fechas:                                      #5
        ultimos_users.append(User.objects.get(date_joined=fecha))     #6

    for user in ultimos_users:                                        #7
        profiles.append(ProfileJ.objects.get(user=user))              #8

    return profiles                                                   #9
```

Figura 24 Código del método last_logged_users ()

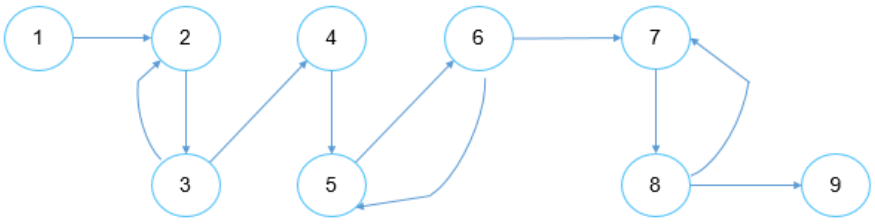


Figura 25 Grafo de flujo asociado al método last_logged_users ()

Fórmulas para calcular la complejidad ciclomática

- ✓ V (G) = (A - N) + 2

Donde “A” es la cantidad de aristas y “N” la cantidad de nodos.

V (G) = (11 - 9) + 2

V (G) = 4

- ✓ V (G) = P + 1

Siendo “P” la cantidad de nodos predicados (son los nodos de los cuales parten dos o más aristas).

V (G) = 3 + 1

V (G) = 4

- ✓ V (G) = R

Donde “R” representa la cantidad de regiones en el grafo.

V (G) = 4

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, por lo que se puede decir que la complejidad ciclomática del código es de cuatro, lo que significa que existen cuatro posibles caminos por donde el flujo puede circular, este valor representa el límite mínimo del número total de casos de pruebas para el procedimiento tratado.

Tabla 5 Caminos básicos del grafo de flujo asociado al método last_logged_users ()

Número	Caminos básicos
1	1-2-3-4-5-6-7-8-9
2	1-2-3-2-3-4-5-6-7-8-9
3	1-2-3-4-5-6-5-6-7-8-9
4	1-2-3-4-5-6-7-8-7-8-9

Posteriormente de haber determinado los caminos básicos se procede a ejecutar los casos de pruebas para cada uno de estos (ver Tablas 6, 7, 8 y 9). Para definir los casos de prueba se tiene en cuenta:

- ✓ Descripción del proceso: Se describe el caso de prueba y de forma general se tratan los aspectos fundamentales de los datos de entrada.
- ✓ Resultados Esperados: Se expone el resultado esperado que debe devolver el procedimiento después de efectuado el caso de prueba.

Tabla 6 Caso de prueba para la trayectoria básica 1 del método de caja blanca

Caso de prueba para la trayectoria básica 1	
Trayectoria	1-2-3-4-5-6-7-8-9
Descripción del proceso.	Selecciona los usuarios registrados en el sistema y los guarda en una lista. Al estar vacía la misma no se puede recorrer y retorna un arreglo vacío.
Resultado esperado.	Se muestra un mensaje diciendo que no hay usuarios en el sistema.

Tabla 7 Caso de prueba para la trayectoria básica 2 del método de caja blanca

Caso de prueba para la trayectoria básica 2	
Trayectoria	1-2-3-2-3-4-5-6-7-8-9

Descripción del proceso.	Selecciona los usuarios registrados en el sistema y los guarda en una lista. Recorre la lista, guarda las fechas de registro y retorna los perfiles de esos usuarios.
Resultado esperado.	Se muestran los 5 últimos usuarios registrados en el sistema.

Tabla 8 Caso de prueba para la trayectoria básica 3 del método de caja blanca

Caso de prueba para la trayectoria básica 3	
Trayectoria	1-2-3-4-5-6-5-6-7-8-9
Descripción del proceso.	Selecciona los usuarios registrados en el sistema y los guarda en una lista. Recorre la lista, al no tener fechas de registro no retorna los perfiles correspondientes a dichos usuarios y retorna un arreglo vacío.
Resultado esperado.	Muestra un mensaje de error diciendo que falta la fecha de registro de los usuarios.

Tabla 9 Caso de prueba para la trayectoria básica 4 del método de caja blanca

Caso de prueba para la trayectoria básica 4	
Trayectoria	1-2-3-4-5-6-7-8-7-8-9
Descripción del proceso.	Selecciona los usuarios registrados en el sistema y los guarda en una lista. Recorre la lista, guarda las fechas de registro al no contar los mismos con perfiles retorna un arreglo vacío.
Resultado esperado.	Muestra un mensaje diciendo que los últimos usuarios registrados en el sistema no poseen perfiles.

3.5.2 Pruebas funcionales

Las pruebas funcionales están basadas en ejecución, revisión y retroalimentación de las funcionalidades implementadas para el software y las especificaciones definidas por el usuario. Con estas pruebas se busca evaluar el sistema mediante modelos de pruebas. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos. Por ello se denominan pruebas funcionales, y el probador se limita a suministrarle datos como entrada y estudiar la salida, sin preocuparse de lo que pueda estar haciendo el módulo por dentro (Pressman, 2010). Las pruebas funcionales que se realizarán a la solución, estarán enfocadas o dirigidas a los CUS para verificar su correcto funcionamiento. En este tipo de pruebas se ejecutan los distintos servicios prestados con datos correctos e incorrectos. En caso de que los datos sean incorrectos se verifica que los mensajes de error sean los deseados y en el caso opuesto que los resultados sean los esperados (Pressman, 2010).

Prueba de Caja Negra

Las pruebas de Caja Negra se aplican a la interfaz del software, examinan algún aspecto funcional de un sistema que tiene poca relación con la estructura lógica interna del software (Pressman, 2010).

Estas pruebas se concentran en los RF del software, tratando de encontrar los siguientes errores (Pressman, 2010):

- ✓ Funciones incorrectas o faltantes.
- ✓ Errores de interfaz.
- ✓ Errores de estructuras de datos o en acceso a bases de datos externas.
- ✓ Errores de comportamiento o desempeño.
- ✓ Errores de inicialización y término.

Los casos de pruebas pretenden demostrar que:

- ✓ Las funciones del software son operativas.
- ✓ La entrada se acepta de forma correcta.
- ✓ Se produce una salida correcta.
- ✓ La integridad de la información externa se mantiene.

Para el diseño de los casos de pruebas de Caja Negra se utilizó la Técnica de Partición Equivalente, la cual divide el dominio de entrada de un programa en clases de datos a partir de las cuales pueden derivarse casos de prueba. La partición equivalente se esfuerza por definir un caso de prueba que descubra ciertas clases de errores, reduciendo así el número total de casos de prueba que deben desarrollarse (Pressman, 2010). A continuación, se presenta el diseño de caso de prueba que se utilizará para comprobar el funcionamiento del sistema.

Descripción de las variables

La Tabla 10 muestra la descripción de las variables que se encuentran asociadas al CU “Administrar proyectos”.

Tabla 10 Descripción de variables para el caso de prueba “Administrar proyectos”

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Foto de portada	Campo de imagen	Sí	Contiene la ruta del directorio donde se encuentra la imagen.
2	Titulo	Cajón de texto	No	Contiene el título del proyecto (debe comenzar con mayúscula).
3	Jefe del proyecto	Lista de selección	No	Almacena el nombre del perfil que es jefe del proyecto.
4	Descripción	Campo de texto	Sí	Contiene la descripción del proyecto.
5	Contenido	Campo de texto	Sí	Almacena el contenido del proyecto en texto HTML.
6	Grupo al que pertenece	Lista de selección	No	Almacena el grupo al que pertenece el proyecto.

7	Aceptado	Lista de selección	No	Estructura que posibilita que el proyecto esté visible o no para los usuarios comunes.

Descripción de Casos de Prueba

Se evaluó y probó la validez de cada una de las entradas al sistema, utilizando datos válidos (V), inválidos (I), y valores que no son necesarios proporcionar (NA). En la Figura 26 se muestra el caso de prueba para el caso de uso “Administrar proyectos”, en el escenario “Crear proyecto”.

Escenario	Descripción	Foto de Portada	Título	Jefe	Descripción	Contenido	Grupo	Aceptado	Respuesta del sistema	Flujo central
EC 1.1 Insertar datos del proyecto correctamente.	El usuario introduce todos los parámetros correctamente y se adiciona el proyecto al sistema.	NA	V	V	NA	NA	V	V	Se inserta un nuevo proyecto al sistema	1. Selecciona la opción “Agregar proyecto” de la página principal de Listar Proyectos. 2. Se llenan y seleccionan los campos. 3. Se selecciona la opción “Crear proyecto”.
			Minería de datos en la gestión de proyectos	Pedro Yobanis Piñero Pérez			ISW	Si		
EC 1.2 Insertar campos inválidos.	El usuario introduce datos inválidos.	NA	I	I	NA	NA	I	I	Muestra un mensaje de error, mecionando los campos erróneos en el formulario	1. Selecciona la opción “Agregar proyecto” de la página principal de Listar Proyectos. 2. Se llenan y seleccionan los campos. 3. Se selecciona la opción “Crear proyecto” 4. El sistema muestra un mensaje de error.
			minería de proyectos							
EC 1.3 Dejar campos vacíos.	El usuario mantiene campos vacíos	NA	I	I	I	I	I	I	Muestra un mensaje de error, diciendo que no se pueden dejar campos obligatorios vacíos y menciona cuáles son esos campos.	1. Selecciona la opción “Agregar proyecto” de la página principal de Listar Proyectos. 2. Se llenan y seleccionan los campos. 3. Se selecciona la opción “Crear proyecto” 4. El sistema muestra un mensaje de error.

Figura 26 Caso de prueba para el CU “Administrar proyectos”, en la sección “Crear proyecto”

Resultados de las pruebas utilizando el método de caja negra

Una vez realizadas las pruebas utilizando el método de caja negra mediante la técnica de partición equivalente usando los casos de pruebas asociados para cada CU, se comprobaron las funcionalidades del sistema y la correcta validación de sus campos. Para una primera iteración se detectan un total de nueve no conformidades de aplicación y una de documentación. Dentro de las de aplicación tres son de validación y seis de ortografía; y la de documentación es de redacción. Para una segunda iteración se detectan una de validación y tres de ortografía, las cuales quedan resueltas para una tercera iteración. La figura 27 muestra un gráfico que representa las iteraciones realizadas.

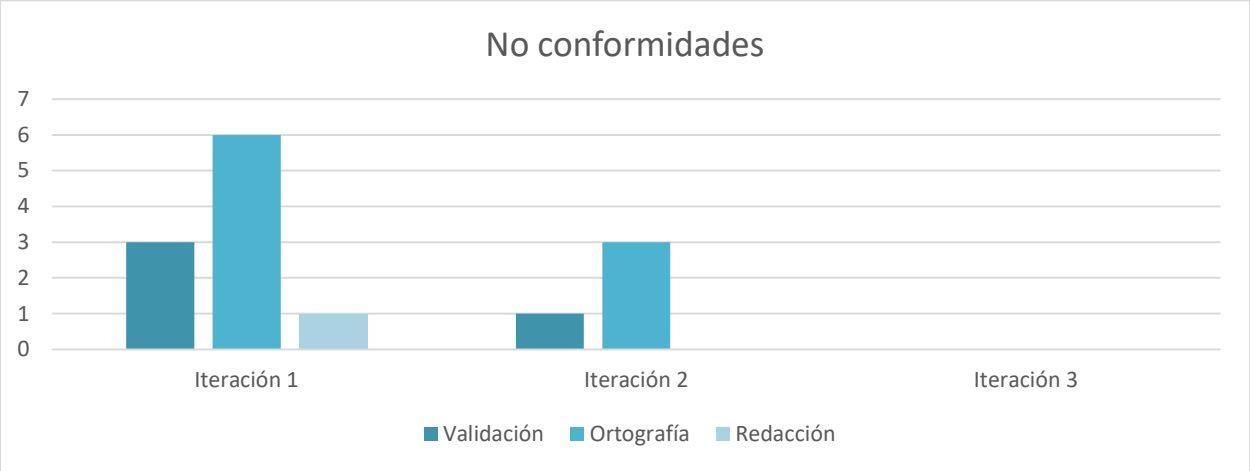


Figura 27 Gráfico de no conformidades aplicando el método de caja negra

3.5.3 Pruebas de seguridad

Las pruebas de seguridad buscan brindar información sobre la confidencialidad, integridad y disponibilidad de los datos, buscando identificar amenazas y riesgos de un sistema. Una vez ejecutadas las pruebas de seguridad es posible medir y cuantificar los riesgos a los cuales se ven expuestas las aplicaciones tanto en la infraestructura interna como externa. Estas pruebas consisten en revisar las aplicaciones en búsqueda de vulnerabilidades (Toledo, 2017).

Pruebas con la herramienta OWASP ZAP

Para validar que el sistema de administración desarrollado resuelve la problemática planteada contribuyendo a la seguridad y la disponibilidad de los datos en la plataforma NOX se realizan pruebas de seguridad basadas en la herramienta OWASP ZAP la cual está diseñada especialmente para monitorizar la seguridad de las aplicaciones web.

Resultados de las pruebas con la herramienta OWASP ZAP

En la ejecución de las pruebas de seguridad mediante OWASP ZAP a cada url se le realizan ataques los cuales consisten en peticiones e inyecciones de Lenguaje de Consulta Estructurada (SQL, del inglés: *Structured Query Language*). La herramienta realiza un total de 1731 peticiones y como resultado de las mismas se obtienen 4 alertas (ver Figura 28), siendo todas de nivel bajo. Además de la validación de todas las entradas a partir de la ruta principal del sistema: “localhost:8000/administrador/” retornando resultados satisfactorios como se muestra a en la Figura 29.

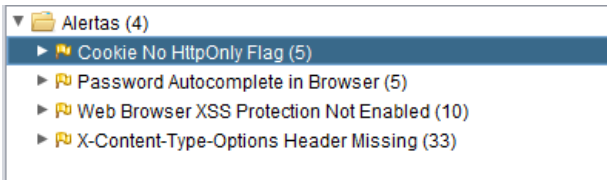


Figura 28 Alertas del sistema a partir de la herramienta OWASP ZAP

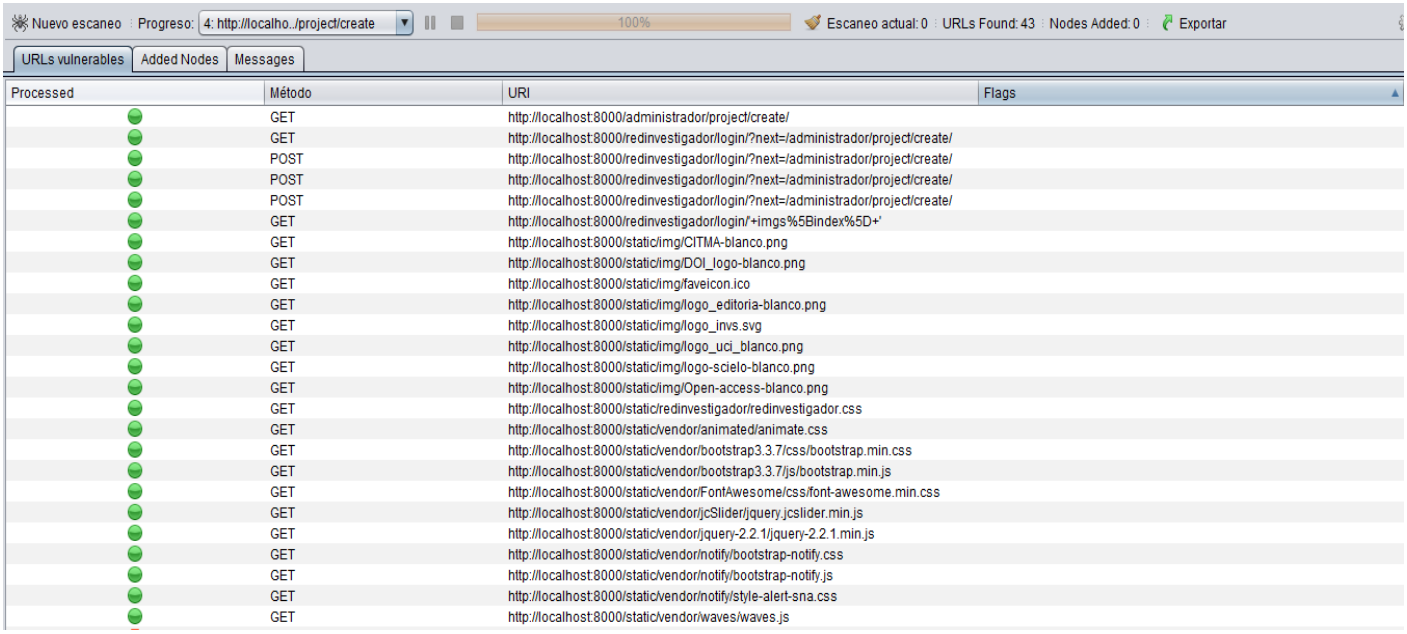


Figura 29 Resultados de las pruebas de seguridad haciendo uso de la herramienta OWASP ZAP

Pruebas con la herramienta Acunetix

Además de las pruebas de seguridad realizadas anteriormente con la herramienta OWASP ZAP se realizan las mismas haciendo uso de la herramienta Acunetix con el fin de detectar errores en el sistema. Para ello la herramienta comprueba diferentes vulnerabilidades con la realización de peticiones las cuales consisten en ataques SQL, de secuencias de comandos entre sitios (XSS, del inglés *Cross-site scripting*), de entidad XML externa (XXE, del inglés: Xml eXternal Entity) e inyecciones de encabezado de host (*Host header injection*). Estas vulnerabilidades se clasifican en:

- ✓ Vulnerabilidades de Severidad Alta
- ✓ Vulnerabilidades de Severidad Mediana
- ✓ Vulnerabilidades de Severidad Baja

Resultados de las pruebas con la herramienta Acunetix

En la ejecución de las pruebas de seguridad la herramienta realiza un total de 3911 peticiones en 26 minutos y 15 segundos (ver Figura 30). La herramienta arroja como resultado un total de 2 vulnerabilidades siendo una de severidad media y otra de severidad baja, las mismas son analizadas y se les da respuesta como se expone a continuación.

- ✓ Vulnerabilidad 1: Modo desarrollador (*debug*) habilitado: El modo de desarrollador en Django se encuentra activado.

Respuesta: Para programar el sistema es necesario tener habilitado el modo desarrollador que permite interactuar al programador con el marco de trabajo de Django, mostrando mensajes de error. Este tipo de modalidad es retirada una vez se despliega la aplicación en un entorno real.

- ✓ Vulnerabilidad 2: Auto-completamiento activado en caja de contraseña: El atributo de auto-completamiento está activado lo que le da la opción al usuario de guardar sus credenciales una vez que las escribe, dando la posibilidad al atacante de obtenerlas de la caché del navegador.

  Respuesta: Detectada esta vulnerabilidad se le introdujo al formulario de inicio de sesión el atributo <INPUT TYPE="password" AUTOCOMPLETE="off">, el cual no permite al navegador guardar las contraseñas de los usuarios que se autentiquen.

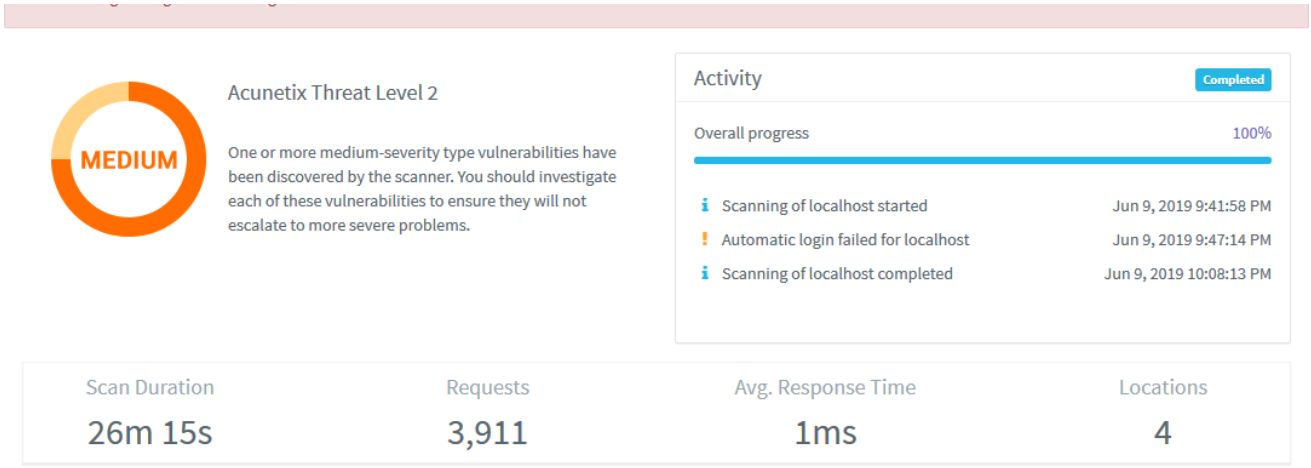


Figura 30 Resultados de las pruebas de seguridad con la herramienta Acunetix

3.5.4 Pruebas de rendimiento

Las pruebas de rendimiento se realizan para determinar lo rápido que realiza una tarea un sistema en condiciones particulares de trabajo. También puede servir para validar y verificar otros atributos de la calidad del sistema, tales como la escalabilidad, fiabilidad y uso de los recursos. Las pruebas de rendimiento son un subconjunto de la ingeniería de pruebas, una práctica informática que se esfuerza por mejorar el rendimiento, englobándose en el diseño y la arquitectura de un sistema, antes incluso del esfuerzo inicial de la codificación (Molyneaux, 2009). Para probar el rendimiento del sistema se realizaron pruebas de carga y estrés.

Pruebas de carga y estrés

Las pruebas de carga son diseñadas para determinar y validar la respuesta de la aplicación cuando es sometida a una carga de usuarios y/o transacciones que se espera en el ambiente de producción. Entre las dimensiones de calidad de esta prueba se encuentra la de rendimiento. Dichas pruebas se esbozan para asegurar que el sistema pueda procesar una carga esperada. Esto normalmente implica planificar un grupo de pruebas en la que la carga se va incrementado regularmente hasta que el rendimiento del

sistema se hace inaceptable. Se ocupan tanto de demostrar que el sistema satisface los requerimientos como de descubrir defectos y problemas en el mismo (Somerville, 2011).

Por su parte, las pruebas de estrés son creadas para encontrar el volumen de datos o de tiempo en que la aplicación comienza a fallar o es incapaz de responder a las diferentes peticiones. La fiabilidad, es una de las dimensiones de calidad de este tipo de pruebas (Somerville, 2011).

Para la realización de este tipo de pruebas se utiliza la herramienta JMeter haciendo uso de una PC con un procesador Intel Core i3-5500u con una frecuencia de 2.0 GHz y una RAM de 4GB.

A continuación, se muestran las variables analizadas:

- Usuarios: cantidad de usuarios.
- Muestra: cantidad de peticiones realizadas para cada url.
- Media: tiempo promedio en milisegundos en el que se obtienen los resultados.
- Mediana: tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.
- Línea 90 %: máximo tiempo utilizado por el 90 % de la muestra.
- Min: tiempo mínimo que demora un hilo en acceder a una página.
- Max: tiempo máximo que demora un hilo en acceder a una página.
- %Error: porciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
- Rend: rendimiento. Se mide en cantidad de solicitudes por segundo.
- Kb/Seg: medición del rendimiento en cantidad de kilobytes por segundo.

Resultados de las pruebas de carga y estrés realizadas mediante la herramienta JMeter

Las pruebas de carga realizadas muestran que el sistema es capaz de responder a 1900 peticiones de 200 usuarios conectados simultáneamente en un tiempo promedio de 1130 milisegundos (1.13 segundos aproximadamente) con 0 % de error. Se realizaron 40900 peticiones iniciadas por 400 usuarios y en este caso el sistema respondió en 2243 milisegundos (2.24 segundos aproximadamente) como tiempo promedio. No fue capaz de responder correctamente el 0.17 % de las peticiones realizadas. Por último, se realizaron pruebas de estrés, con 51200 peticiones iniciadas por 800 usuarios y en este caso el sistema respondió en 2950 milisegundos (2.95 segundos aproximadamente) como tiempo promedio. No fue capaz de responder correctamente el 0.32 % de las peticiones realizadas.

Tabla 11 Resultados de las pruebas de rendimiento con la herramienta JMeter

Usuarios	Muestra	Media	Mediana	Línea 90 %	Min	Max	%Error	Rend	Kb/s
200	1900	1130	1320	1276	2	1288	0	83.4	740.7
400	40900	2243	2609	2046	3	423547	0.17	74.2	662.1
800	52200	2950	5125	7946	4	500956	0.32	66.1	548.9

3.5.5 Pruebas de aceptación

Las pruebas de aceptación son realizadas principalmente por los usuarios con el apoyo del equipo del proyecto. El propósito es confirmar que el sistema está terminado, que desarrolla puntualmente las necesidades de la organización y que es aceptado por los usuarios finales (Carillo, 2011).

El sistema desarrollado fue presentado ante los especialistas principales del proyecto SNA, los cuales luego de haberlo probado están satisfechos con el producto entregado. Prueba de esto lo constituye el acta de aceptación emitida, en la cual consta que el sistema realizado cumple con las condiciones de calidad necesarias y satisface las necesidades plasmadas.

3.6 Valoración de la satisfacción mediante la técnica de Iadov

Para valorar la satisfacción respecto a la seguridad y disponibilidad del sistema se empleó la técnica Iadov que permite el estudio del grado de satisfacción del personal involucrado en un proceso objeto de análisis. La misma, constituye una vía indirecta para el estudio de la satisfacción, ya que los criterios que se utilizan se fundamentan en las relaciones que se establecen entre tres preguntas cerradas que se intercalan dentro de un cuestionario y cuya relación el sujeto desconoce. Estas tres preguntas se relacionan a través de lo que se denomina el cuadro lógico de Iadov (Rodríguez, y otros, 2002).

Primeramente, se aplicó una encuesta de satisfacción a varias personas, siendo las mismas líderes científicos e investigadores destacados con más de 5 años de experiencia, teniendo en cuenta su familiarización con el proceso investigativo de la universidad. En este caso, la técnica está compuesta por cinco preguntas: tres cerradas y dos abiertas, las cuales se reformulan en la investigación para valorar el grado de satisfacción del sistema de administración. Una vez establecidas las preguntas se conforma el “cuadro lógico de Iadov” (ver Tabla 14).

El número resultante de la interrelación de las tres preguntas, indica el nivel de satisfacción de los sujetos, que se expresa en la escala numérica que oscila entre 1 y -1 tal y como se muestra en la Tabla 6. Posteriormente se calcula el índice de satisfacción grupal (ISG) según la ecuación 1, utilizando los diferentes niveles de satisfacción (ver Tabla 13).

Ecuación 1 Cálculo del índice de satisfacción grupal

$$ISG = \frac{A(+1) + B(+0{,}5) + C(0) + D(-0{,}5) + E(-1)}{N}$$

Tabla 12 Niveles de satisfacción expresados en la escala numérica

Nivel de satisfacción	Cantidad
Máximo de satisfacción	+1
Más satisfecho que insatisfecho	+0.5
No definido y contradictorio	0

Más insatisfecho que satisfecho	-0.5
Máxima insatisfacción	-1

Tabla 13 Cuadro lógico de Iadov

¿Te gusta que la solución al problema planteado sea un sistema de administración?	¿El sistema de administración NOX-Admin cumple con todas las funcionalidades requeridas?								
	No			No sé			Si		
	¿Consideras que el sistema contribuye la seguridad y disponibilidad de la información en la plataforma NOX?								
	Si	No sé	No	Si	No sé	No	Si	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta mucho	2	2	3	2	3	3	6	3	6
No me interesa	3	3	3	3	3	3	3	3	3
Me disgusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

Según las categorías empleadas los resultados de la satisfacción se evidencian en la siguiente tabla.

Tabla 14 Resultado de la aplicación de la técnica de Iadov

Nivel de satisfacción	Cantidad	Porciento
Máximo de satisfacción	6	85.71
Más satisfecho que insatisfecho	1	14.29
No definido y contradictorio	0	0.00
Más insatisfecho que satisfecho	0	0.00
Máxima insatisfacción	0	0.00

Sustitución en la ecuación del ISG:

$ISG = 6(+1) +1(+0,5) +0(0) +0(-0,5) +0(-1) / 7$

$ISG = 6,5 / 7$

$ISG = 0,93$

Ubicación del ISG en el eje numérico:

Figura 31 Ubicación del ISG en el eje numérico

Al procesar las respuestas a las encuestas en el cuadro lógico de Iadov, se obtiene un grado de satisfacción grupal de 0,93, lo cual se traduce en una clara satisfacción con el uso del sistema de administración NOX-Admin. En el criterio de los encuestados respecto a que el sistema cumple con todas las funcionalidades requeridas, existe una concordancia de un 100,00%. Respecto a la posibilidad de contribuir a la seguridad y disponibilidad de la información en la plataforma NOX se obtiene una concordancia de un 100%. En cuanto a que es acertada la solución planteada, el 57,14% de los encuestados considera que les gusta la propuesta, el 28,57% responde que no le agrada del todo y el 14,29% lo califica como “No me interesa”.

3.7 Validación de la investigación

A partir del problema a resolver y el análisis de la relación causa-efecto entre la variable dependiente “control de los datos en la plataforma NOX” y las variables independientes “seguridad” y “disponibilidad” de la información, se definen un conjunto de criterios de medida que permiten demostrar cómo a través de la propuesta de solución se logra la relación entre ambas variables. La obtención de estos criterios se realiza teniendo en cuenta las principales deficiencias identificadas en la situación problemática a partir una entrevista realizada con los desarrolladores de NOX y los especialistas principales del proyecto SNA.

- ✓ Existencia de diferentes roles: este criterio incide en la seguridad, ya que al no existir varios roles cualquier usuario puede acceder al sistema de administración sin estar autorizado.
- ✓ Definición de permisos según cada rol: este criterio influye en la seguridad ya que permite al administrador del sistema decidir qué información puede manipular cada usuario según su rol.
- ✓ Aprobación de entidades investigativas: este criterio incide en la disponibilidad ya que las líneas y grupos de investigación necesitan ser aprobadas para su posterior visualización en la plataforma.
- ✓ Generación de reportes estadísticos: permite visualizar los sucesos acontecidos en la plataforma NOX contribuyendo a la disponibilidad de dicha información.
- ✓ Adición de revistas: este criterio incide en la disponibilidad permitiendo la adición y actualización de las revistas que se mostrarán a los investigadores de NOX.

En la siguiente tabla se evalúan cada uno de los criterios de medida definidos. La evaluación se realiza a través de un antes (NOX sin un sistema de administración que contribuya a la seguridad y disponibilidad de la información) y un después (NOX con un sistema de administración que contribuya a la seguridad y disponibilidad de la información), con el propósito de verificar cómo, mediante la solución propuesta, se contribuye a la seguridad y disponibilidad de la información para el control de los datos de la plataforma NOX. Los datos a partir de los cuales se realiza la comparación fueron tomados a partir de una encuesta realizada a los especialistas principales del proyecto SNA y los desarrolladores de la plataforma NOX y la técnica de Iadov, descrita en el anterior epígrafe.

Tabla 15 Validación de las variables de la investigación

Criterio	Antes	Después
Existencia de diferentes roles	Según el 100 % de los encuestados NOX cuenta con un único rol, por lo que cada investigador puede acceder al sistema y realizar las mismas operaciones que los demás.	La seguridad en NOX se beneficia ya que al existir varios roles el acceso al sistema se limita solamente a los usuarios autorizados.
Definición de permisos según cada rol	El 95% de los encuestados alega que los usuarios de NOX poseen determinados permisos los que les permiten realizar una serie de operaciones sobre el sistema, de igual forma existen algunas que no están definidas para ningún usuario ya que estos podrían acceder y modificar información de cualquier tipo.	Con la definición de permisos según cada rol por parte del administrador desde NOX-Admin se contribuye a la seguridad de la información y quedan cubiertas todas las posibles operaciones y requisitos del sistema, definiendo todas las responsabilidades que presenta cada usuario. Donde los usuarios con rol “Jefe de investigaciones” tienen acceso a los módulos de reportes, estadísticas, investigaciones y notificaciones; los usuarios con el rol “Editor de eventos y revistas” tienen acceso a los módulos de estadísticas, eventos, revistas y notificaciones; mientras que los que poseen rol: “Administrador” tienen acceso a todos los módulos del sistema.
Aprobación de entidades investigativas	A partir de la encuesta realizada, el 81% los encuestados plantean que un investigador al proponer una línea o un grupo de investigación tiene que contactar con un especialista del equipo de desarrollo de NOX para que realice la aprobación de la misma desde la base de datos, causando una demora en la disponibilidad de la	Con el sistema de notificaciones desarrollado en NOX-Admin se permite notificar a los usuarios encargados de la aprobación de las entidades investigativas, las cuales desde una interfaz amigable tienen la posibilidad de aprobarlas para su posterior visualización en la plataforma contribuyendo a su disponibilidad.

	información de dicha entidad investigativa.	
Generación de reportes estadísticos	Según el 100% de los encuestados la plataforma NOX no realiza generación de reportes estadísticos para llevar un monitoreo de lo ocurrido en la misma.	Con la generación de reportes se contribuye a la disponibilidad de información estadística del acontecer de la plataforma.
Adición de revistas	Según el 96% de los encuestados NOX contiene una serie de datos con los cuales se realiza una búsqueda de revistas existentes en la web, las cuales posteriormente son visualizadas por los investigadores. Estos datos, en el momento de la encuesta, son adicionados por un especialista del equipo de desarrollo de la plataforma desde la base de datos; lo que propicia la demora de la adición de las mismas y a su disponibilidad en la plataforma.	NOX-Admin permite la adición de los datos de las revistas que los investigadores pueden visualizar por parte de los usuarios encargados para esta tarea a partir de una interfaz amigable, además del recibo de notificaciones por parte de los usuarios con la información de sus revistas de interés.

La comparación realizada en la tabla anterior, a través de los criterios de medida antes definidos, demuestra cómo, con el desarrollo de un sistema de administración para el control de los datos de la plataforma NOX se contribuye a la seguridad y disponibilidad de la información. A pesar del planteamiento anterior, una vez NOX sea utilizado por los investigadores, se puede agregar a NOX-Admin la posibilidad de analizar cómo estos interactúan con la plataforma, que módulos prefieren y que errores presenta la aplicación a medida que esta sea usada, con el fin de mejorar la experiencia del usuario y la usabilidad del sistema, lo cual puede realizarse con técnicas de análisis de datos.

Conclusiones de capítulo

A partir de los resultados obtenidos en el presente capítulo se tiene que:

- ✓ Con la elaboración de los diagramas de componentes y despliegue se logró ilustrar la relación entre los principales componentes del sistema y la organización física del mismo.
- ✓ La implementación del sistema de administración NOX-Admin para la UCI cumple con el estándar de codificación descrito lo que permite hacer más fácil el entendimiento del código y el futuro mantenimiento del sistema.
- ✓ La ejecución de las pruebas unitarias mediante el ambiente de ejecución Unittest y el método de caja blanca permitió detectar errores en la implementación del sistema posibilitando la posterior corrección de los mismos; las pruebas funcionales mediante el método de caja negra comprobaron las funcionalidades del sistema y la correcta validación de sus campos. La

realización de pruebas de seguridad con las herramientas OWASP ZAP y Acunetix permitió identificar las principales vulnerabilidades a las que podía estar expuesto el sistema, para poder darle solución a tiempo; y las pruebas de rendimiento permitieron identificar el tiempo de respuesta del sistema cuando es sometido a una carga de usuarios y transacciones esperadas una vez que el mismo esté en explotación.

- ✓ La realización de las pruebas de aceptación por los especialistas principales del proyecto SNA permitió verificar y validar el cumplimiento de los requisitos previamente identificados, arrojando que la solución propuesta cumple con las condiciones de calidad necesarias.
- ✓ La valoración mediante la técnica de Iadov y la comparación realizada teniendo en cuenta un antes y un después del desarrollo el sistema demostró que los usuarios finales están satisfechos con la solución implementada y que esta satisface las necesidades plasmadas.

Conclusiones Generales

En el desarrollo del presente trabajo de diploma se obtuvo como resultado la creación del sistema de administración NOX-Admin dando cumplimiento al objetivo general, por lo que se arriba a las siguientes conclusiones:

- ✓ El estudio de los principales conceptos, contribuciones y tecnologías relacionadas con la administración de sistemas para la gestión de la información científico-técnica permitió sentar las bases arquitectónicas e ingenieriles para el desarrollo de la solución.
- ✓ Con la implementación del sistema NOX-Admin guiada por la metodología de desarrollo AUP-UCI en correspondencia con los requisitos identificados se obtuvo una solución que favorece el control de los datos en la plataforma NOX contribuyendo así a su seguridad y disponibilidad.
- ✓ La validación de la solución propuesta mediante las pruebas de software internas y de aceptación, además de la valoración de la satisfacción con el uso del sistema mediante la técnica y de Iadov y la comparación a partir de un antes y un después de implementada la solución demostró el correcto funcionamiento del sistema, el cumplimiento de las necesidades plasmadas con la adecuada calidad y su aptitud para ser usado.

Recomendaciones

Según los resultados obtenidos y las experiencias adquiridas en el transcurso del desarrollo del sistema, se recomienda:

- ✓ Realizar las pruebas de liberación pertinentes al sistema por la dirección de calidad y la dirección de seguridad informática de la universidad.
- ✓ Una vez que el sistema esté en explotación analizar la interacción de los investigadores en el sistema utilizando minería de usos y minería web, con el fin de mejorar la calidad y la usabilidad del mismo, así como satisfacer las necesidades de los usuarios de acuerdo a sus principales intereses.
- ✓ Adicionar e implementar nuevos requisitos referentes a la obtención de reportes estadísticos, de acuerdo a las necesidades que puedan surgir por parte de los usuarios con roles administrativos encargados del seguimiento y control de la información científico-técnica en la plataforma.

Referencias

Akademos. 2019. Sistema de Gestión Universitaria. [En línea] 2019. [Citado el: 02 de 02 de 2019.] https://akademos.uci.cu/estructura_investigativa/portada.

Alchin, M. 2013. *SVN Access Manager Documentation Understanding Django, in: Pro Django. Springer.* s.l. : ACCESSMANAGER, 2013.

Anacleto, V. A. 2006. *Reusabilidad Orientada al Negocio.* s.l. : MDA, 2006.

BBVA. 2016. CMS basados en Django: ejemplos y características. [En línea] 2016. [Citado el: 05 de 02 de 2019.] https://bbvaopen4u.com/es/actualidad/cms-basados-en-django-ejemplos-y-caracteristicas.

Berti, Elena y Thompson, Rebeca. *Aplicaciones Web.* s.l. : Google Anaylitics.

Bertino, E. A y Martino, L. A. 1995. *Sistemas de bases de datos orientadas a objetos.* 1995.

Cambronero, Antonio. 2016. *La guía del recién llegado a Wordpress.* 2016.

Carbonell, Daniel Hang. 2012. *Análisis y diseño del Módulo Administración para CRODA 2.0.* 2012.

Carillo, Fernando. 2011. *Pruebas del Software: Niveles de Prueba del Software.* 2011.

Coexia. 2015. Creating solutions, Uso de Frameworks en el Desarrollo Web. [En línea] 2015. https://www.coexia.com/uso-de-frameworks-en-el-desarrollo-web/.

Commerce, U.S. Department of. 2012. National Institute of Standards and Technology. [En línea] 11 de 06 de 2012. [Citado el: 14 de 2 de 2019.] http://csrc.nist.gov/groups/SNS/rbac/.

Dante, Gloria Ponjuán. 2005. *Gestión de información: dimensiones e implementación para el éxito organizacional.* s.l. : Universidad de Buenos Aires : Instituto de Investigaciones Bibliotecología, 2005. Vols. 1514-8327..

Diaz Lazo , Juliet y Pérez Gutiérrez, Adriana. 2011. *Impacto de las Tecnologías de la información y las comunicaciones para reducir la dividión digital de la sociedad hoy en día.* La Habana : s.n., 2011.

Edeki, Charles. 2013. Agile Unified Process. [En línea] 2013.

ERP, Proyecto. 2008. *Normas y estándares de Codificación del ERP.Línea de Arquitectura. .* La Habana : Universidad de las Ciencias Informáticas : s.n., 2008.

Expósito, Christopher, y otros. *Unfied Modeling Lenguaje, Tema 1. Introducción a UML.*

Fergarciac. 2013. Entorno de Desarrollo Integrado. [En línea] 2013. [Citado el: 14 de 2 de 2019.] https://fergarciac.wordpress.com/2013/01/25/entorno-de-desarrollo-integrado-ide/.

Figueroa Diaz, Roberth, Sólis, Camilo y Cabrera, S. 2007. *Metodologías tradicionales vs. metodologías ágiles.* 2007. 10.13140/RG.2.1.2897.3206. .

Franco Espiño, Beatriz y Pérez Alcázar, Ricard. 2015. Redrta.org. *Redrta.org.* [En línea] 2015. http://mgd.redrta.org/directrices-administracion-de-documentos-electronicos/mgd/2015-01-27/161143.html.

Gamma, Erick. 2008. *Patrones de Diseño: elementos de software orientados a objetos reutilizables.* 2008.

Gauchat, Juan Diego. 2012. *El gran libro de HTML 5, CSS y JavaScript.* 2012.

Gerrero Martínez, Rafael. 2010. *PostgreSQL-es.* 2010.

Guglielmetti, Marcos. 2012. *MasterMagazine.* 2012.

Jacobson, y otros. 2000. *El Proceso Unificado del Desarrollo de Software. Madrid : Addyson Wesly.* 2000.

Kaplan-Moss, Jacob. 2013. *El libro de Django.* 2013.

Larman, Craig. 1999. *UML y patrones. Introducción al análisis y diseño orientado a objetos.* s.l. : Prentice Hall, 1999.

Martínez Mengual, Yamil. 2016. *Sistema de Gestión de Información de las Órdenes de Servicios en Copextel UCI.* La Habana, Universidad de Ciencias Informáticas : s.n., 2016.

Martinez Sánchez, Rigoberto. 2018. *Seguridad Informatica En Empresas.* 2018.

Medina Rodríguez, Ania, del Busto Pou, Ismary y Hurtado Machado, Teresa. 2016. *La gestión de la información científica desde el Centro de Documentación Información pedagógica.* s.l. : Universidad Central “Marta Abreu” de Las Villas, 2016.

Molyneaux, Ian. 2009. *The art of application performance testing: Help for programmers and quality assurance. " O'Reilly Media, Inc.".* 2009. 978-0-596-52066-3.

OpenMind. 2019. Redes sociales académicas: qué son y cómo pueden ayudar a la ciencia. [En línea] 2019. [Citado el: 08 de 01 de 2019.] https://www.bbvaopenmind.com/humanidades/comunicacion/redes-sociales-academicas-que-son-y-como-pueden-ayudar-a-la-ciencia/.

Oracle. 2011. Guía de administración del sistema: servicios de seguridad. . [En línea] 2011. [Citado el: 02 de 2 de 2019.] http://docs.oracle.com/cd/E24842_01/html/E23286/rbac1.html. E23286.

Paradigm, Visual. 2015. Visual Paradigm Know-how. Business modeling, software development and enterprise architecture with great products, and knowledge. . [En línea] 2015. [Citado el: 9 de 1 de 2019.] https://www.ecured.cu/Visual_Paradigm.

Pavón Mestras, Juan. 2011. *Bootstrap, aplicaciones web / sistmas web.* s.l. : Dep. Ingeniería del Software e Inteligencia ArtificialFacultad de InformáticaUniversidad Complutense Madrid, 2011.

Pérez Porto , Julián y Gardey, Ana. 2014. Definicion.de: Definición de sistema administrativo. [En línea] 2014. [Citado el: 05 de 04 de 2019.] https://definicion.de/sistema-administrativo/.

Pérez, Tomás Saorín. 2010. 2010.

Pressman, Roger. 2010. *Ingenieria de software : Un enfoque practico.* s.l. : McGraw-Hill, 2010.

Rodríguez, A. L. y Maura, V. G. 2002. *La técnica de Iadov, vol. 8, n.o47.* 2002.

Sánchez Rodriguez, Tamara. 2014. *Metodología de desarrollo para la Actividad productiva de la UCI.* Universidad de las Ciencias Informáticas : s.n., 2014.

Sánchez, Miguel Angel. 2001. *JavaScript.* s.l. : INNOVA, 2001.

Santa Cruz Pacheco, Michel Subert. 2015. *Sistema informático para la gestión de la información de los procesos de las Brigadas Técnicas Juveniles.* 2015. pág. 6, Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Somerville, Ian. 2011. *Ingeniería de Software. 9na.* México: Addison Wesley : Pearson Education, Inc. , 2011.

Suárez Alfonso, A, Cruz Rodríguez, I y Pérez Macías, Y. 2015. *La gestión de la información: herramienta esencial para el desarrollo de habilidades en la comunidad estudiantil universitaria. Revista Universidad y Sociedad.* 2015.

Suárez, Pablo y Fontela, Carlos. *Documentación y pruebas antes del paradigma de objetos.*

Tecnologías-Información. 2018. Modelo de datos. [En línea] 2018. [Citado el: 05 de 02 de 2019.] https://www.tecnologias-informacion.com/modeladodatos.html.

Thornton, Jacob y Eguiluz, javier. 2015. Manual Oficial. [En línea] 2015. https://librosweb.es/libro/bootstrap_3.

Toledo, Federico. 2017. Introducción al testing de seguridad con OWASP ZAP. [En línea] 2017. [Citado el: 04 de 05 de 2019.] www.federico-toledo.com/introduccion-al-testing-de-seguridad/.

Torchbox. 2019. *Wagtail Documentation.* 2019.

Tramullas, Jesús. 2010. *Drupal para bibliotecas y archivos.* 2010.

Turnquist, Greg L. 2011. *Python Testing Cookbook.* 2011.

Van Rossum, Guido. 2017. *El tutorial de Python.* 2017.

Velázquez Álvarez, Dámaso. 2012. *Aplicaciones web Vs Aplicaciones de escritorio.* 2012.