Universidad de las Ciencias Informáticas

Facultad 1

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

Módulo de Administración para el Sistema de Información Primaria de Personas.

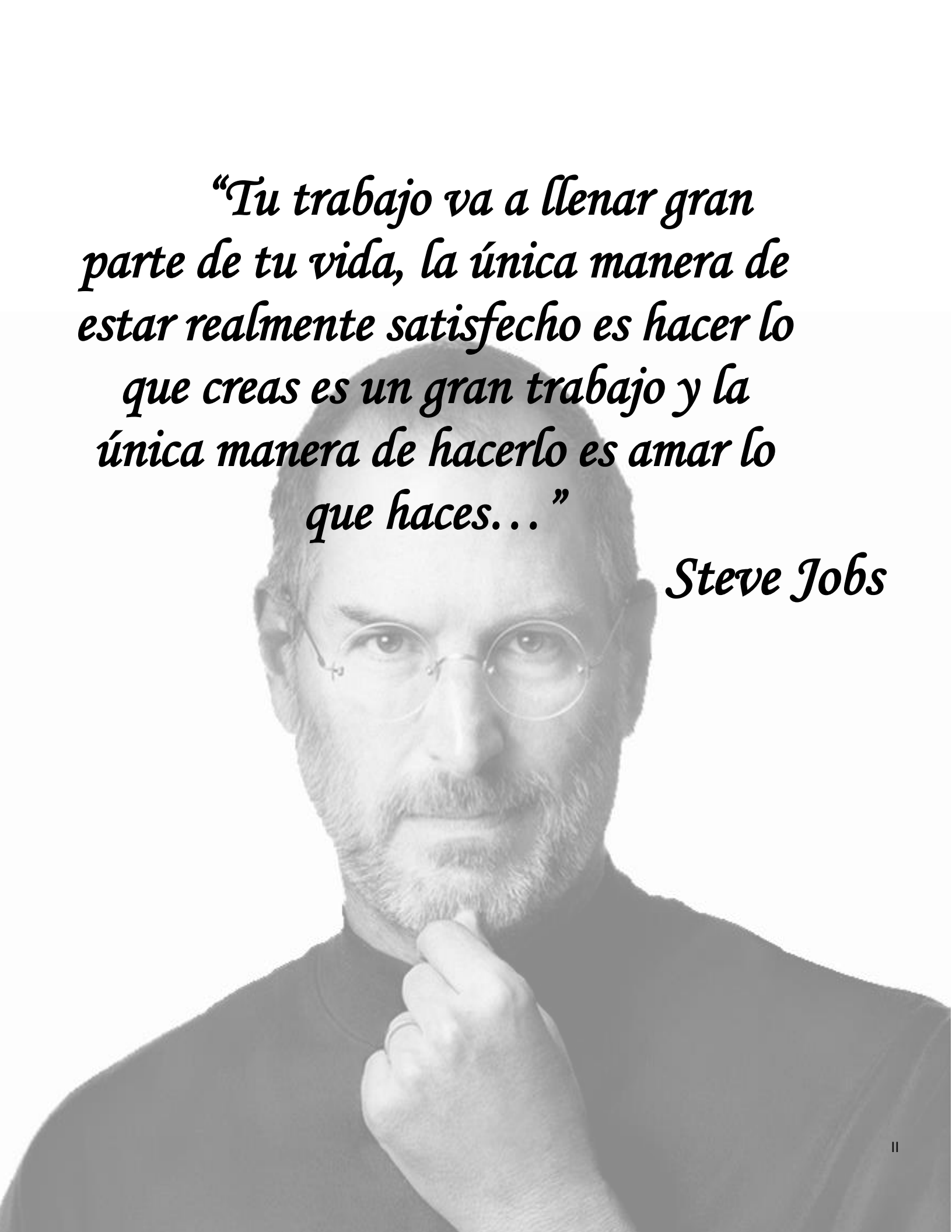
Autor: Evelyn Morales Moreno

Tutor(es): Ing. Asistente Mayleidis López Fernández

Ing. Instructor Osay Gonzáles Fuentes

La Habana Cuba

"Año 60 de la Revolución"



"Tu trabajo va a llenar gran parte de tu vida, la única manera de estar realmente satisfecho es hacer lo que creas es un gran trabajo y la única manera de hacerlo es amar lo que haces…"

Steve Jobs

DECLARACIÓN DE AUTORÍA

Declaro ser la única autora de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales de la misma, con carácter exclusivo.
Para que así conste firmo la presente a los _____ días del mes de _____ del año _____.

_____

Autora
Evelyn Morales Moreno

_____ _____

Firma del Tutor *Firma del Tutor*
Mayleidis López Fernández *Osay González Fuentes*

Datos de contacto:

Ing. Mayleidis López Fernández: Graduada de Ingeniero Industrial Especializado en Organización de Empresas en el Instituto Superior Politécnico "José Antonio Echeverría" (CUJAE) en el 2003. Profesora de Proyecto de Investigación y Desarrollo del Departamento de Ingeniería de Software y Práctica Profesional de la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI). Profesor Asistente.

Correo: mayleidis@uci.cu

Ing. Osay Gonzáles Fuentes: Graduado de Ingeniero en Ciencias Informáticas en la Universidad de las Ciencias Informáticas (UCI) en el 2006. Director del Centro de Identificación y Seguridad Digital (CISED) de la Facultad 1 de la Universidad de las Ciencias Informáticas (UCI).

Correo: ogonzalezf@uci.cu

Agradecimientos:

Le agradezco principalmente a mi familia por ser mi guía y soporte y por apoyarme en todo momento.

A mi mamá y mi papá, que con su ejemplo y sabios consejos me han enseñado a no rendirme ante nada y a luchar por lo que quiero.

A todos mis amigos, los que estuvieron cerca y los que a pesar de la distancia se preocuparon y me ayudaron en lo que estaba a su alcance, especialmente a Anaili, Carlos, David, Cesar y Lily, por soportarme, escucharme y apoyarme en los momentos buenos y malos. A mis amigas Sandra y Claudia por ser mis confidentes y por enseñarme que en la vida las cosas para alcanzarlas hay que luchar. Gracias por ser mis amigos, los quiero mucho.

A Surama, Iraida, Daniela, Sahily, Yasser y Reynaldo por hacerme sentir parte de su familia.

A Mayleidis y Osay por guiarme y apoyarme en este camino tan difícil y por su dedicación.

A mis profesores por todo su tiempo dedicado a enseñar.

Al tribunal y oponente que con sus revisiones y sugerencias hicieron posible el perfeccionamiento de este trabajo de diploma.

Dedicatoria:

A mi mamá y mi papá que son mi razón de ser y la luz de mi vida.

A mi hermano por quererme tanto y confiar en mí.

A Christian por formar parte de mi vida y haber sido mi soporte estos últimos 5 años.

Resumen

Cada vez son más las instituciones que deciden incorporar sistemas de identificación de personas a su flujo de trabajo para tener un mayor control del personal autorizado. Actualmente la Universidad de las Ciencias Informáticas no cuenta con un sistema capaz de recoger y centralizar los datos de las personas y puedan ser consultados por otros sistemas que pertenezcan a una misma empresa. Debido a esto, se le ha dado la tarea al Centro de Identificación y Seguridad Digital CISED de crear en Sistema de Información Primaria de Personas (SIPP). Este está compuesto por cinco módulos (Control de Acceso, Trazas, Recursos Humanos, Captura de Datos y Administración). En este trabajo se desarrolla el Módulo de Administración que gestionará principalmente los permisos de los usuarios del sistema dependiendo de sus roles. Para ello se realizó un estudio de sistemas similares, donde se analizaron los procesos de administración que se realizan en diferentes sistemas, tanto nacionales como internacionales. Se seleccionó la metodología AUP versión UCI para la organización de la investigación, generándose los artefactos correspondientes en cada fase. Fueron descritas las herramientas y tecnologías utilizadas en la solución desarrollada, culminando con las pruebas para la verificación del sistema y para comprobar que se cumple con lo definido por el cliente, teniendo como resultado un módulo que garantiza la integridad de los nomencladores y se supervisa la información del SIPP.

Palabras claves: administración, identificación, información, módulo, tecnología.

Abstract

More and more institutions decide to incorporate people's identification systems to their workflow to have a greater control of the authorized personnel. Currently the Computer Science University does not have a system capable of collecting and centralizing the data of people and it can be consulted by other systems belonging to the same company. Due to this, the CISED Digital Identification and Security Center has been given the task of creating the Primary Person Information System. SIPP is composed by five modules (Access Control, Services, Human Resources, Data Capture and Administration). In this research is developed an Administration Module, that will mainly manage the permissions of the users of the system depending on their roles. For this, a study of similar systems was carried out, where the administration processes executed in different systems, both national and international, were analyzed. The AUP UCI version methodology was selected for the organization of the research, generating the corresponding artifacts in each phase. The tools and technologies used in the developed solution were described, culminating with the tests for the validation of the system and to be capable of verifying that it is fulfilled with what was defined by the client, resulting in a module that will guarantee the integrity of the nomenclators and supervise the SIPP information.

Keywords: administration, identification, information, module, technology.

Índice General

Índice de Tablas

Índice de Figuras

Introducción

Con el desarrollo de las Tecnologías de la Información y las Comunicaciones (TIC) ha aumentado el proceso de informatización de la sociedad mediante el desarrollo de aplicaciones informáticas que cada vez son más flexibles y adaptables a las necesidades de los clientes. La integración de las TIC en los procesos de negocio, representa un objetivo significativo para el desarrollo de las empresas, convirtiéndolo en un reto a nivel mundial.

Cuba, a pesar de ser un país subdesarrollado, no está ajena del avance de la ciencia y las tecnologías. Hoy la informática se percibe como una vía de desarrollo tanto económico como social, en la cual la industria del software deberá jugar un papel muy importante. Además, con el proceso de informatización de la sociedad, se ha facilitado el uso de muchas de las técnicas y métodos que se aplican en las organizaciones hoy en día mediante la utilización de programas y aplicaciones informáticas que evitan que muchos de los procesos que antes se hacían de forma manual, se realicen de una forma más rápida, segura y con un menor margen de error.

Para ayudar en el proceso de informatización de la sociedad cubana, en el año 2002 surge la Universidad de las Ciencias informáticas (UCI) como obra del pensamiento visionario de Fidel Castro. Esta universidad productiva persigue el objetivo de elevar el desarrollo económico y la formación de jóvenes vinculándolos desde sus primeros años a proyectos reales en los distintos centros de producción que existen en la misma. Entre los centros de producción se encuentra el Centro de Identificación y Seguridad Digital (CISED) de la Facultad 1 que se dedica principalmente al desarrollo de soluciones integrales, productos y servicios en el campo de la identificación y la seguridad digital.

Actualmente, los sistemas producidos por la universidad y que han sido comercializados no intercambian entre ellos la información de las personas de una misma empresa lo que conlleva a la duplicidad de datos en múltiples sistemas aumentando las probabilidades de que exista error en la información nominal de las personas. La duplicidad de información aumenta de manera considerable la necesidad de hardware, específicamente el almacenamiento y el procesamiento de los datos. En todos los subsistemas que componen el ecosistema de software de una entidad existen datos nominales de personas que al ser expuestos para su consumo por otras aplicaciones comprometen la seguridad, integridad y confidencialidad de los datos. Cada sistema para autenticarse requiere de un usuario y una contraseña lo cual genera una cantidad de contraseñas y usuarios a conocer por una misma persona, trayendo consigo la necesidad de

recordar varias contraseñas, lo que hace más vulnerable los sistemas. Además, la gestión de usuarios y roles se realiza utilizando diferentes estándares lo que dificulta la comunicación entre los sistemas, hace más lento el proceso.

Una vez planteada la problemática descrita anteriormente se presenta como **problema de investigación:** ¿Cómo lograr la seguridad de los nomencladores y supervisar la información del SIPP?

Quedando definido como **objeto de estudio**: Los procesos de gestión y administración en sistemas de información de personas.

Se propone como **objetivo general** desarrollar un módulo de administración para lograr la integridad de los nomencladores y la supervisión de la información del SIPP, quedando desglosado en los siguientes **objetivos específicos**:

- ✓ Analizar la gestión de la administración en los sistemas de información de personas.
- ✓ Diseñar el Módulo de Administración para el SIPP.
- ✓ Implementar el Módulo de Administración para el SIPP.
- ✓ Realizar pruebas al Módulo de Administración para el SIPP.

Se plantea como **hipótesis de la investigación** que con la implementación del Módulo de Administración del SIPP se logrará la integridad de los nomencladores y se supervisará la información del SIPP. Para la realización de esta investigación se utilizarán algunos métodos científicos los cuales están dialécticamente relacionados:

Métodos teóricos

El **histórico-lógico** con el objetivo de estudiar la evolución de las características de los sistemas administrativos a nivel mundial, con el fin de apoyar el proceso de modelado.

El **analítico-sintético** en la descomposición del problema de la investigación que permitió el estudio y análisis de los diferentes conceptos y definiciones más importantes relacionadas con el proceso de administración de aplicaciones web.

Modelos empíricos

- **Observación:** Para observar de forma exhaustiva el funcionamiento de módulos de administración

de otros sistemas.

- **Análisis documental:** Para extraer la información necesaria de literaturas especializadas, tanto académicas como empresariales, enriqueciendo de forma sustancial el proceso de investigación.
- **Análisis comparativo:** Para detectar diferentes características, ventajas y desventajas en diferentes módulos de administración.

El presente trabajo está estructurado en 3 capítulos:

Capítulo 1: Fundamentación teórica

Se desarrolla la fundamentación teórica de la investigación, donde se describen los principales conceptos, haciendo énfasis en el módulo de administración, así como las principales herramientas y tecnologías para su desarrollo.

Capítulo 2: Propuesta de solución

Se exponen las características del Módulo de Administración. Se describen los requisitos funcionales y no funcionales que ofrecerán una solución al problema de investigación, se realiza una propuesta de solución. Además, se plantea la arquitectura y los patrones de diseño que seguirá el Módulo de Administración.

Capítulo 3: Implementación y prueba

Se expone el diagrama de componentes y la descripción de los mismos. Se describe la implementación de la aplicación y su funcionamiento. También, se muestran las pantallas principales del módulo desarrollado, así como la verificación de la solución a través de las pruebas realizadas a la aplicación.

Capítulo 1: Fundamentación teórica

Para una mejor comprensión del tema tratado es necesario el estudio de los principales conceptos asociados al módulo de administración, así como las principales herramientas y tecnologías para su desarrollo. El estudio de sistemas similares aportará las características a tener en cuenta en la elaboración de la solución.

1.1 Conceptos fundamentales

Módulo de programa

En programación, un módulo es un fragmento de un programa que se desarrolla de forma independiente del resto del programa. Esta independencia hace posible un mecanismo de compilación por separado que limita la complejidad del programa que se está desarrollando. Al compilarse el módulo por separado, la persona que lo desarrolla sólo debe preocuparse de él, prescindiendo en parte de cómo se utiliza este módulo dentro del programa. Quien escriba el resto del programa no debe preocuparse de los detalles del módulo sino sólo de cómo utilizarlo (Gallardo José E.) .

Administración

Según Juan Pablo Amador, la administración “Es el proceso de lograr que las cosas se realicen por medio de la planeación, organización, delegación de funciones, integración de personal, dirección y control de otras personas, creando y manteniendo un ambiente en el cual la persona pueda obrar entusiastamente en conjunto con otras, sacando a relucir su potencial, eficacia y eficiencia y lograr así fines determinados” (Pablo).

Módulo de Administración

Un Módulo de Administración es una estructura organizativa que permite que el sistema funcione de forma ordenada, segura y consistente. Su uso garantiza eficiencia en el resto de los módulos y sus procesos, permitiendo la centralización de algunos procedimientos y configuraciones globales que garantizarían el buen funcionamiento del mismo. Realizan varias operaciones fundamentales:

- Controlar las operaciones como parte de la seguridad.
- Definir las estructuras lógicas para tener una buena organización en el sistema.
- Gestionar las configuraciones globales o específicas de los negocios que integra (Venero Paez, y otros,

2010).

¿Qué tipo de funcionalidades debe realizar un Módulo de Administración?

La mayoría de las aplicaciones por lo general necesitan la gestión o configuración de los usuarios y de las oficinas donde operan sus funcionarios, control de acceso restringido para cada proyecto, la gestión de proyectos, definición de servidores y bases de datos a acceder. Además, las herramientas de colaboración entre desarrolladores, definición de dominios, gestión de sesiones activas para balanceo de carga y mejoramiento del tiempo de respuesta, y otras funcionalidades que son comunes para cualquier sistema. Partiendo de estos conceptos se dice que el módulo administrativo de un sistema informático se encarga de llevar a cabo las funcionalidades antes mencionadas con el objetivo de perfeccionar todo el funcionamiento del mismo (Venero Paez, y otros, 2010).

Como se evidencia anteriormente un elemento importante en la administración de un programa es el término usuario, así como los roles que desempeñan estos en determinado sistema para así garantizar la seguridad de la información.

Usuario de la información

Puede referirse a una persona, grupo o entidad. Se usa para designar a quien utiliza la información o los servicios de información. En el área relativa a la computación se llama usuario al que utiliza un sistema automatizado (un determinado hardware o software) independientemente de que quien lo use sea trabajador o no (Usos y definiciones de los términos relativos a los usuarios o clientes, 2000).

Gestión de usuarios

En un sistema operativo multiusuario se utiliza el concepto de usuario para controlar el acceso al sistema. Se pueden declarar diferentes usuarios y asignar un nivel de acceso diferente, o unos privilegios, para cada uno de ellos. Normalmente antes de trabajar en el sistema es necesario iniciar una sesión, momento en el que la persona que quiere acceder al sistema se identifica como uno de los usuarios del sistema. Aunque de forma intuitiva se puede pensar que cada usuario del sistema es una persona diferente, esto no tiene que ser siempre así. Un usuario del sistema puede ser utilizado por diferentes personas físicas. Independientemente de quien declare ser, si se identifica con la contraseña adecuada, recibirá el mismo trato por parte del sistema informático. Además, es común definir usuarios en el sistema operativo que no

corresponden a ninguna persona. Son usuarios que se utilizan para ejecutar determinados procesos, normalmente servicios, y restringir el acceso a los recursos por parte de dichos procesos.

Así, un usuario se puede corresponder con:

- Una persona
- Un conjunto de personas (todas conocen el identificador y contraseña del usuario)
- Ninguna persona (Carceler, 2016).

Grupo

Los sistemas operativos que trabajan con usuarios, normalmente también permiten declarar grupos. Un grupo es una colección de usuarios y normalmente se utiliza para aplicar un mismo trato (por ejemplo, imponer una limitación en la cantidad de disco utilizada) a cada uno de los usuarios que pertenecen al grupo (Carceler, 2016).

Modelo de control de acceso

El control de acceso es un elemento clave en la seguridad de un sistema y sirve como complemento importante a la definición de la interacción entre los usuarios del sistema y en el caso de sistemas colaborativos a la interacción y coordinación entre los diferentes usuarios y los recursos que utilizan. Un elemento importante en los sistemas colaborativos es el contexto, los modelos de control de acceso deben tener en consideración este elemento para establecer los permisos de acceso, es decir, el modelo autorizará o no el acceso teniendo en cuenta el contexto actual en el que se encuentre el usuario.

La mayor parte de los modelos usados actualmente, se han diseñado, específicamente, para analizar y modelar los aspectos de seguridad por lo que es necesario modelar los aspectos funcionales de forma separada e integrarlos posteriormente. Uno de los modelos más usados es RBAC (Role-Based Access Control o Control de Acceso Basado en Roles). Este modelo usa el concepto de rol en vez de la asignación de privilegios directamente a usuarios. Esto facilita la clasificación de privilegios y permite un control de acceso fino a los recursos (Sánchez, y otros).

Rol de usuarios

Un rol de usuario permite conectar usuarios a compañías; se hace al usuario responsable de la compañía. Se pueden crear varios roles de usuario y luego asignar tareas específicas a los distintos roles. De este modo, se pueden tener diferentes usuarios responsables de distintas tareas para la misma compañía (2017).

Sistema de Información

Un sistema de información queda definido como: “conjunto formal de procesos que, operando sobre una colección de datos estructurada de acuerdo a las necesidades de la empresa, recopila, elabora y distribuyen selectivamente la información necesaria para la operación de dicha empresa y para las actividades de dirección y control correspondientes, apoyando, al menos en parte, los procesos de toma de decisiones necesarios para desempeñar funciones de negocio de la empresa de acuerdo con su estrategia” (Andreu, y otros, 1991).

1.2 Estudio de sistemas similares

Con el objetivo de identificar características y funcionalidades aplicables en la solución se realizó un estudio de sistemas similares tanto en el ámbito nacional como internacional, los cuales se describirán a continuación:

1.2.1 Blackbaud eTapestry

Es un software de administración de donantes y recaudación de fondos para pequeñas, medianas y grandes organizaciones sin fines de lucro. Su funcionalidad principal incluye una base de datos de donantes y un sistema de administración de relaciones, paneles e informes personalizados, herramientas de marketing para correo electrónico y correo directo, herramientas de análisis de datos y calidad de datos, e integración de sitios web para donaciones en línea y procesamiento de pagos. Las opciones de administrador incluyen un informe de usuario que enumera los usuarios de eTapestry para su base de datos y un resumen de su actividad reciente. Al profundizar en un nombre de usuario, puede ver su actividad de los últimos siete días. Si corresponde, tiene la capacidad de desconectar usuarios específicos en la base de datos y cambiar las contraseñas de los usuarios. Las organizaciones que tienen múltiples usuarios accediendo a eTapestry encontrarán que el módulo de administración es una herramienta útil para revisar y administrar el trabajo de sus usuarios dentro de la base de datos. Con este módulo se puede:

- Consultar una lista de usuarios, sus identificadores de inicio de sesión, su última hora de inicio de sesión / cierre de sesión y la última actividad dentro de la base de datos de la organización.
- Recopilar detalles específicos sobre el historial de inicio de sesión de un usuario, incluida la duración del inicio de sesión.
- Crea un informe de actividad del usuario para un usuario específico o para todos los usuarios.
- Profundiza para ver qué cuentas fueron modificadas por un usuario específico y la hora y fecha de modificación.
- Cierra la sesión de un usuario en la base de datos de la organización y cambia su contraseña (Island, 2011).

Después de estudiar este sistema y las opiniones de usuarios que lo usan (tomadas de Vinay Bhagat, 2013-2018) se puede decir que la interfaz de eTapestry es muy cargada y tiene un diseño ordinario, por lo que parece antiguo. Con esta información se puede evitar cometer estos errores a la hora de crear el Módulo de Administración para el SIPP, la interfaz del mismo será sencilla, moderna e intuitiva con el fin de que los usuarios con conocimientos básicos en el manejo de una computadora puedan utilizarla fácilmente.

1.2.2 FHIR v3.0.1

Fast Healthcare Interoperability Resources (FHIR) o Recursos de interoperabilidad de salud rápida es un estándar de interoperabilidad para el intercambio electrónico de información médica. Este posee un módulo administrativo que cubre los datos básicos de un sistema de salud que permite el intercambio de registros electrónicos de salud o historia clínica electrónica y comunicarse con la nube. Este luego se vincula a los otros módulos del sistema. Técnicamente, FHIR está diseñado para la web; los recursos se basan en estructuras XML[1] o JSON[2] simples, con un protocolo RESTful[3] basado en http[4] donde cada recurso tiene una *Uniform Resource Locator o Localizador Uniforme de Recursos* (URL) predecible. Siempre que sea posible, se utilizan estándares abiertos de Internet para la representación de datos (2017).

Este sistema tiene una interfaz sencilla e intuitiva. Además, fue implementado con protocolos antiguos para que se pudiera integrar con sistemas ya creados. Es fácil de desarrollar y de implementar gracias a su

[1] XML: siglas en inglés de eXtensible Markup Language, traducido como “Lenguaje de Marcado Extensible”
[2] JSON: Acrónimo de Java Script Object Notation
[3] RESTful: Representational State Tranfer, Traducido como “Transferencia de Estado Representacional (Protocolo)”

[4] http: *siglas en inglés de Hypertext Transfer Protocol*, traducido como “protocolo de transferencia de hipertextos”

diseño sencillo. Además, es una excelente alternativa para facilitar la adopción de estándares de interoperabilidad. Esto es de gran ayuda para el desarrollo del SIPP, ya que se puede crear un sistema con características similares en cuanto a interfaz y a interoperabilidad, y así obtenga una buena aceptación por parte de los usuarios que lo utilicen.

1.2.3 Sistema de Administración Nacional de la Población de la República de Cuba

"El Sistema de Administración Nacional de la Población de la República de Cuba es una plataforma desarrollada con un enfoque a procesos y se encarga de la gestión de oficinas, usuarios y roles necesarios para la realización de todas las funciones del Sistema Único de Identificación Nacional de la Población de la República de Cuba (SUIN)" (2012).

Para acceder al sistema el administrador debe crear el usuario con los permisos que necesitará en dependencia del módulo al que accederá.

El sistema está desarrollado en plataforma Asp.NET. Está integrado al paquete de Servicios del SUIN y a su vez a la base de datos única del ciudadano. Además, está compuesto por 19 módulos entre los cuales se encuentran:

- Gestión de Usuarios
- Gestión de Roles
- Control de Acceso
- Trazas
- Reportes
- Aplicaciones

1.2.4 Módulo de Administración para el Sistema Autónomo de Identificación, Migración y Extranjería (SAIME)

Es un sistema realizado para la reestructuración, modernización y automatización de la Oficina Nacional de Identificación y Extranjería de Venezuela (ONIDEX). Este sistema cuenta con un Módulo de Administración que tiene como objetivos: el control de las oficinas y los funcionarios que operan con el sistema SAIME

además de un conjunto de configuraciones necesarias para el control centralizado y la eficiencia, seguridad e integridad de los procesos que se llevan a cabo. Este sistema brinda las siguientes funcionalidades:

- Configuración de oficinas móviles.
- Configuración de usuarios de oficinas.
- Configuración de usuarios.
- Configuración de sesiones y puestos de trabajo.
- Configuración de plantillas de roles para los tipos de oficinas.
- Configuración de servidores.
- Generación de actualizaciones.
- Autenticación de los usuarios en el sistema.
- Gestión de usuarios por oficinas (Venero Paez, y otros, 2010).

1.2.5 Módulo de administración CMDBuild (Base de datos de administración de configuración)

MDBuild es una aplicación web de código abierto diseñado para modelar y administrar activos y servicios controlado por el Departamento de Tecnología de la Información y las Comunicaciones (TIC), por lo tanto, maneja las operaciones de flujo de trabajo relacionadas. La gestión de una Base de Datos de Configuración (CMDB) significa mantenerse actualizado y disponible a otros procesos, la base de datos relacionada con los componentes en uso, sus relaciones y sus cambia con el tiempo. Con CMDBuild, el administrador del sistema puede construir y extender su propia CMDB, modelándola según las necesidades de la compañía. El módulo de administración te permite añadir progresivamente nuevas clases de elementos, nuevos atributos y nuevas relaciones. También puedes definir filtros, "vistas" y permisos de acceso limitados a filas y columnas de cada clase. Gracias al motor de flujo de trabajo integrado, puede crear nuevos procesos de flujo de trabajo con editores visuales, e importarlos / ejecutarlos dentro de la aplicación CMDBuild de acuerdo con automatismos configurados. También está disponible un administrador de tareas integrado en la interfaz de usuario del módulo de administración. Esto permite administrar diferentes operaciones (inicio del proceso, recepción y envío de correo electrónico, conector ejecuciones) y controles de datos en la CMDB (eventos síncronos y asincrónicos). Residencia en sus hallazgos, envía notificaciones, inicia flujos de trabajo y ejecuta scripts. Este sistema incluye dos módulos principales:

- El módulo de administración, solo para usuarios administradores, está destinado para el inicio de la configuración básica y las modificaciones consecutivas.
- El módulo de gestión de datos, utilizado por los operadores, le permite gestionar la consulta y actualización de las tarjetas de datos, ejecutar los procesos, imprimir los informes, etc.

Las principales características de este módulo incluyen:

Definición del modelo de datos

- creación de nuevas "clases", es decir, nuevos tipos de objetos en el sistema
- creación y modificación de los "atributos" de una clase (normal y geográfica)
- definición de los "widgets" para colocar en el formulario de gestión de "clase"
- definición de la información de persistencia de los flujos de trabajo
- importación del flujo de trabajo
- impresión del esquema del modelo de datos (completo o solo limitado a la clase solicitada)
- creación de "dominios", es decir, tipos de relaciones y sus posibles atributos
- creación de una lista de tablas ("búsqueda") para gestionar los atributos de los valores predeterminados

Definición de vistas y filtros

- creación de filtros predeterminados que se pueden usar durante la consulta de datos en diferentes clases
- creación de "vistas" usando filtros de datos y usando consultas SQL

Gestión de usuarios y grupos

- creación de usuarios
- creación de grupos de usuarios
- definición de las subvenciones y roles para cada grupo y cada clase o subconjunto de filas y columnas de la clase.

- definición de menú personalizado para diferentes tipos de usuarios.

1.2.5 Valoración de los sistemas estudiados

Una vez analizadas las características de los sistemas estudiados se puede concluir que estos sistemas fueron desarrollados para entidades específicas y responden a necesidades únicas de su negocio. Además, en el caso de los sistemas internacionales no se cuenta con la documentación necesaria, ni se tiene acceso al código fuente por lo que no se pueden utilizar para dar solución a nuestra problemática.

Sin embargo, el estudio realizado permitió que se identificaran algunas características y funcionalidades que presentan los módulos de administración:

- Desarrollo de una aplicación web ya que tienen un camino mucho más sencillo para la compatibilidad multiplataforma que las aplicaciones de software descargables.
- La capacidad de permitir la autenticación de los usuarios en el sistema.
- La capacidad de configurar y gestionar los elementos que serán utilizados en la realización de los procesos de cada uno de los módulos.
- La capacidad de gestionar y otorgar permisos a los usuarios según el rol que desempeña.

1.3 Metodología de desarrollo

Siendo las metodologías para desarrollar software los marcos que permiten la estructuración, planificación y control de un proyecto de software, se requiere que éstos sean concisos, ordenados, claros y específicos. Todo proyecto de software deberá planearse, estructurarse y desarrollarse con habilidad, paciencia y conocimiento. No obstante, son múltiples los problemas que se presentan al realizar dicha labor. El desarrollo de software requiere orden, disciplina y una excelente gestión para que dicha tarea sea eficiente y de calidad. Numerosos han sido los autores de metodologías para desarrollar software. Las han creado ágiles, incrementales, iterativas, por etapas, evolutivas y secuenciales, entre otras. Todos estos marcos de trabajo se han utilizado para estructurar, planificar y controlar proyectos de software. Lo que se propone es un método para desarrollar software que proporciona agilidad en el desarrollo, control de calidad en el producto final, sencillez en su uso, facilidad para comprenderlo y utilizarlo, y que además sea corto y manejable en equipos de trabajo grandes o pequeños (Educación en Ingeniería, 2015).

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

FIGURA 1: TABLA COMPARATIVA DE LAS METODOLOGÍAS DE DESARROLLO (2017).

1.3.1 Algunas metodologías ágiles

Scrum

Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está orientada a proyectos en los que cambian rápidamente los requisitos. El desarrollo de un proyecto se hace por medio de iteraciones, llamadas Sprint, las cuales duran 30 días, luego de lo cual se le presenta al cliente. Cada Sprint incrementa el desarrollo. También se efectúan reuniones constantes de quince minutos dentro del equipo de trabajo hasta que termine el proyecto (Educación en Ingeniería, 2015).

Adaptive Software Development o Desarrollo de Software Adaptativo

Es un método iterativo que tolera cambios y está centrado en componentes de software, no tanto en las tareas. Está compuesta por tres fases: especulación (se inicia el proyecto, se planifican las características del producto), colaboración (se desarrolla) y aprendizaje (se revisa la calidad y se entrega al cliente). Esta última etapa es la de aprendizaje de los errores y se vuelve a iniciar el ciclo hasta culminar (Educación en Ingeniería, 2015).

Extreme Programming o Programación Extrema (XP)

Creada por Kent Beck, es una metodología liviana para desarrollar software. Se caracteriza porque planifica, analiza y diseña en un mismo momento y durante el desarrollo. De esta forma se decide si se va por buen camino, y se evita el retroceso en etapas adelantadas, es decir, se trabaja a partir de prueba y error. El equipo está formado por entre dos y doce personas que trabajan en parejas. Es un método simple que desarrolla sólo lo que requiere, permite la retroalimentación constante, la toma de decisiones difíciles y remediar los errores tan pronto como se detectan; existe comunicación continua entre los clientes y el grupo de trabajo (Educación en Ingeniería, 2015).

Crystal Methodologies o Metodologías Crystal

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo, Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) (Canós, y otros).

Metodología AUP-UCI

A pesar de la variedad de metodologías usadas por los diferentes centros productivos de la UCI, se ha comprobado que muy pocos proyectos las aplican en su totalidad. La diferencia entre estas metodologías no radica únicamente en los productos de trabajos que proponen o en sus roles, sino en su forma de planificar el proyecto y realizar las estimaciones del tiempo.

Para lograr erradicar los problemas detectados, se propone crear un cronograma tipo y un método de estimación para cada una de las metodologías que hoy se usan en los proyectos o converger a una única metodología que cubra las particularidades de cada uno. El esfuerzo en tiempo y en personas de la primera variante es mucho mayor que la posibilidad de converger a los proyectos hacia una sola metodología de desarrollo. Por lo tanto, se decide escoger una metodología para ser adaptada a lo que ya la Universidad

ha estado proponiendo como ciclo de vida de los proyectos, sin alejarse de lo que hasta el momento se ha trabajado e introducir la menor cantidad de cambios posibles.

Al no existir una metodología de software universal, toda esta debe ser adaptada a las características de cada proyecto (recursos, equipos de desarrollo, etc.) exigiéndose así que en el proceso sea configurable. Se decide hacer una variación de la metodología AUP (Agile Unified Process o Proceso Unificado Ágil), de manera tal que se adapte al ciclo de vida definido para el desarrollo del componente propuesto (2015).

De las 4 fases que propone AUP se decide para el ciclo de vida de los proyectos de la UCI la siguiente estructura:

- Inicio: Se llevan a cabo las actividades relacionadas con la planeación.
- Ejecución: Ejecución de las actividades requeridas para el desarrollo del software, incluyendo el ajuste de los planes del proyecto, considerando los requisitos y la arquitectura.
- Cierre: Se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales del cierre del proyecto.

AUP-UCI define cuatro escenarios para modelar el sistema dependiendo de la complejidad y características del proceso de desarrollo. El escenario número cuatro de esta metodología se adapta al desarrollo del componente propuesto. Aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan un negocio muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, debido a que una Historia de Usuario (HU) no debe poseer demasiada información (2015).

Después de analizadas las metodologías de desarrollo anteriores, se decide seleccionar la metodología AUP en una versión desarrollada por la UCI, específicamente el escenario cuatro de historias de usuarios, ya que es una metodología de desarrollo ágil, que permite la adaptación al uso de herramientas independientes y la capacidad de reunir en una única disciplina las etapas de análisis de diseño, requisitos y modelado de negocio.

1.4 Herramientas y tecnologías a utilizar

Producto a que el sistema va a estar integrado por varios módulos que se desarrollarán de manera independiente, y se utilizará la metodología AUP-UCI; como estrategia del centro y a solicitud del cliente se ha decidido el uso de las siguientes herramientas y tecnologías:

1.4.1 Lenguaje de modelado

UML

El lenguaje unificado de modelado o UML (*Unified Modeling Language*) es el sucesor de la oleada de métodos de análisis y diseño orientados a objetos (OOA&D) que surgió a finales de la década de 1980 y principios de la siguiente.

El UML unifica, sobre todo, los métodos de Booch, Rumbaugh y Jacobson, pero su alcance llegará a ser mucho más amplio. En estos momentos el UML está en pleno proceso de estandarización con el OMG (*Object Management' Group* o Grupo de administración de objetos) (Scott, 1999).

UML es un lenguaje de modelado, y no un método. La mayor parte de los métodos consisten, al menos en principio, en un lenguaje y en un proceso para modelar. El lenguaje de modelado es la notación (principalmente gráfica) de que se valen los métodos para expresar los diseños. El proceso es la orientación que nos dan sobre los pasos a seguir para hacer el diseño (Scott, 1999).

1.4.2 Lenguaje de programación

Python 3.5

Python es un lenguaje de programación poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. La elegante sintaxis de Python y su tipado dinámico, junto con su naturaleza interpretada, hacen de éste un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas. El intérprete de Python y la extensa biblioteca estándar están a libre disposición en forma binaria y de código fuente para las principales plataformas desde el sitio web de Python, http://www.python.org/, y puede distribuirse libremente. El mismo sitio contiene también distribuciones y enlaces de muchos módulos libres de Python de terceros, programas y herramientas, y documentación adicional. El intérprete de Python puede extenderse fácilmente con nuevas funcionalidades y tipos de datos

implementados en C o C++ (u otros lenguajes accesibles desde C). Python también puede usarse como un lenguaje de extensiones para aplicaciones personalizables (Rossum, 2009).

1.4.3 Marco de trabajo para el desarrollo:

Django 1.10.3

Django es un marco de trabajo web de Python, de alto nivel que permite el desarrollo rápido de sitios web seguros. Construido por desarrolladores experimentados, Django se encarga de gran parte de las complicaciones del desarrollo web, por lo que puedes concentrarte en escribir tu aplicación sin necesidad de reinventar la rueda. Es gratuito y de código abierto, tiene una comunidad próspera y activa, una gran documentación y muchas opciones de soporte gratuito y pago. Django fue desarrollado inicialmente entre 2003 y 2005 por un equipo web que era responsable de crear y mantener sitios web de periódicos. Después de crear varios sitios, el equipo comenzó a factorizar y reutilizar muchos códigos y patrones de diseño comunes. Este código común se convirtió en un marco de desarrollo web genérico, que fue de código abierto como el proyecto "Django" en julio de 2005. Django ha seguido creciendo y mejorando, desde su primer lanzamiento importante (1.0) en septiembre de 2008 hasta la versión 1.11 (2017) recientemente publicada. Cada lanzamiento ha agregado nuevas funcionalidades y correcciones de errores, que van desde soporte para nuevos tipos de bases de datos, plantillas de motores y almacenamiento en caché, hasta la adición de clases y funciones de vista "genéricas" (que reducen la cantidad de código que los desarrolladores tienen que escribir una serie de tareas de programación) (2017).

1.4.4 IDE (Integrated Develop Environment o Entorno de Desarrollo Integrado) de desarrollo.

Pycharm 2017 2.2

PyCharm presenta un editor de código inteligente que comprende los detalles de Python y proporciona productividad que incluye: formateo automático de códigos, finalización de código, refactorizaciones, importación automática, navegación por código con un clic y más. Además, con la adición de rutinas avanzadas de análisis de código como base, estas características hacen de PyCharm una herramienta útil tanto para los desarrolladores profesionales de Python como para aquellos que recién están comenzando con la tecnología (2017).Y debido a que PyCharm se basa en la plataforma IntelliJ, hereda el soporte de

ese producto para la edición de JavaScript, HTML[5] y CSS[6], entre otras características de las que se beneficiarán los desarrolladores web. Con estas capacidades, se espera que PyCharm se convierta en un IDE de Python líder incluso antes de su próxima versión principal, dijo JetBrains en un comunicado de prensa en PyCharm (2017).

1.4.5 Sistema Gestor de Base de Datos

PostgreSQL 9.4

PostgreSQL es un avanzado sistema de bases de datos relacionales basado en Open Source. Esto quiere decir que el código fuente del programa está disponible a cualquier persona libre de cargos directos, permitiendo a cualquiera colaborar con el desarrollo del proyecto o modificar el sistema para ajustarlo a sus necesidades. PostgreSQL está bajo licencia BSD[7]. Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas del´algebra relacional. Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras (2017).

PosgreSQL usa el modelo cliente/servidor. Una sesión en PostgreSQL consiste en ejecución de los siguientes procesos. El servidor, que maneja archivos de bases de datos, acepta conexiones a las aplicaciones cliente, y realiza acciones en la base de datos. El programa servidor de bases de datos se conoce como postmaster. La aplicación cliente, que necesita realizar operaciones en la base de datos. Las aplicaciones cliente pueden ser de la más diversa naturaleza: pueden ser aplicaciones de texto en una consola, aplicaciones gráficas, un servidor web que accede a la base de datos para mostrar una página, o herramientas especializadas de mantenimiento de bases de datos. Como es habitual en las aplicaciones cliente/servidor, el cliente y el servidor pueden estar en diferentes máquinas. En este caso, estos se comunican sobre una conexión de red TCP/IP[8] (2017).

[5] HTML: siglas en inglés de HyperText Markup Language, traducido como" Lenguaje de marcas de hipertexto"
[6]CSS: siglas en inglés de Cascading Style Sheets, traducido como" Hojas de Estilo en Cascada"
[7] BSD: siglas en inglés de *Berkeley Software Distribution, traducido como "Distribución de software de Berkeley"*
[8] TCP/IP: siglas en ingles de *Transmission Control Protocol*/*Internet Protocol, traducido como "*Protocolo de control de transmisión/*Protocolo* de Internet"

1.4.6 Herramienta de Ingeniería de software

Visual Paradigm 8.0

Visual Paradigm es una herramienta de software diseñada para que los equipos de desarrollo de software modelen el sistema de información comercial y administren los procesos de desarrollo. Visual Paradigm admite lenguajes y estándares de modelado de la industria clave como UML, SysML, SoaML, BPMN, XMI, etc. Ofrece herramientas de software completas que las empresas necesitan para la captura de requisitos, análisis de procesos, diseño de sistemas, diseño de bases de datos, y otros (2017).

1.5 Conclusiones del capítulo

Una vez analizados los conceptos y características expuestas en este capítulo, se puede concluir que:

- El estudio de los conceptos fundamentales permitió una mejor comprensión de los sistemas de información de personas y de sus módulos de administración.
- El análisis de cómo se realizan los procesos en sistemas similares arrojó que ninguno de los sistemas estudiados constituye una solución, sin embargo, aportaron características aplicables al módulo.
- La caracterización de herramientas, metodologías y tecnologías facilitó la comprensión de las mismas.

Capítulo 2: Propuesta de solución

En este capítulo se presenta la propuesta de solución al problema de investigación. Se realiza un modelo de dominio donde se representan las entidades principales y cómo están relacionadas entre sí. Se definen los requerimientos funcionales y no funcionales del módulo, que son descritos a través de las HU. Se define la arquitectura del módulo y se presentan los diagramas de clases, diagramas de secuencia y modelo de datos de la solución propuesta.

2.1 Modelo de dominio

"Un modelo de dominio es una representación de las clases conceptuales del mundo real, no de componentes de software. No se trata de un conjunto de diagramas que describen clases de software, u objetos software con responsabilidades" (Larman, 2003).

2.1.1 Diagrama de clases del modelo del dominio

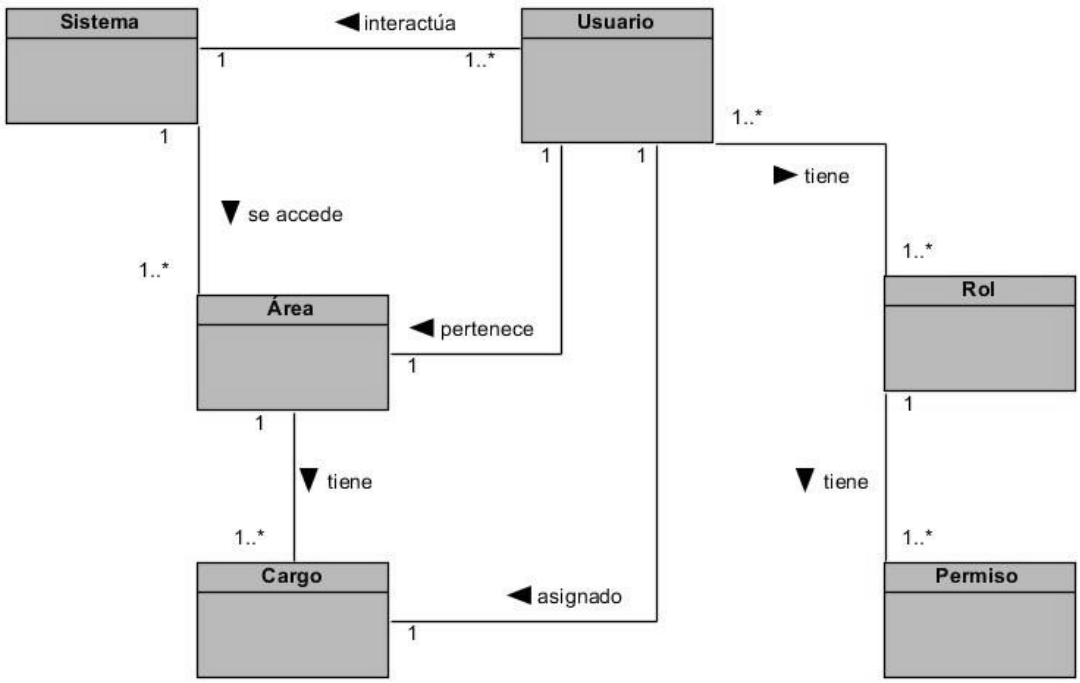


FIGURA 2: MODELO DE DOMINIO DEL MÓDULO DE ADMINISTRACIÓN (ELABORACIÓN PROPIA)

2.1.2 Glosario de conceptos del modelo del dominio

Sistema: Es el Sistema de información Primaria de Personas.

Usuarios: Son las personas que accederán e interactuarán con el sistema.

Roles: Los roles son asociados a los usuarios y definen los permisos que tendrán los mismos en el sistema.

Permisos: Son asignados a los roles con el objetivo de dar acceso a diferentes recursos en dependencia del rol que posea el usuario.

Áreas: Los locales donde se encuentran ubicados los usuarios.

Cargos: Están asociados a los usuarios

2.2 Propuesta de solución:

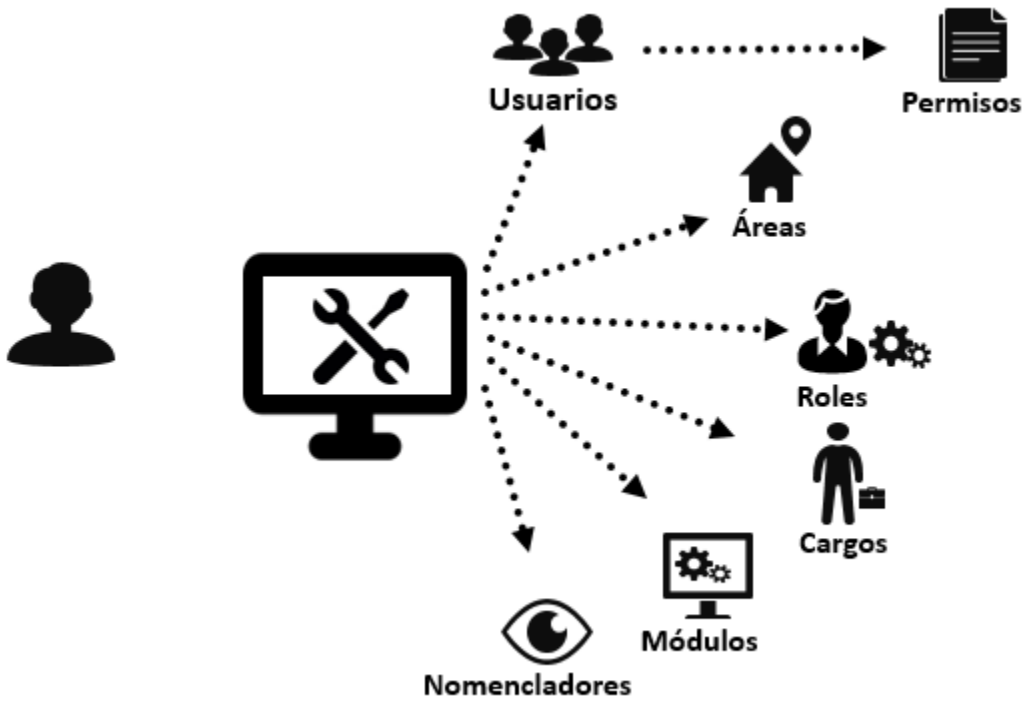


FIGURA 3:DESCRIPCIÓN DE LA PROPUESTA DE SOLUCIÓN. (ELABORACIÓN PROPIA)

Para dar cumplimiento al objetivo planteado se propone desarrollar un módulo de administración que estará integrado al SIPP. El módulo propuesto tendrá como objetivo garantizar la integridad de los nomencladores y supervisar la información de dicho sistema.

Para acceder a este módulo es necesario que se encuentre autenticado en el sistema un usuario con permisos de administrador, una vez dentro, el módulo deberá mostrar las opciones para gestionar las áreas que pertenecen a la entidad, los cargos que poseen los usuarios del sistema, así como sus roles. Además, deberá mostrar las opciones para gestionar los usuarios del sistema.

Para la gestión de cada uno de los elementos mencionados anteriormente el módulo debe permitir realizar las siguientes operaciones:

- ***Listar:*** esta será la primera opción que visualizará el usuario al acceder a una determinada gestión; una vez dentro el módulo debe mostrar un listado con todos los elementos almacenados permitiendo que ellos puedan ser activados, desactivados y modificados. Además, debe brindar la opción de agregar nuevos y mostrar los detalles de cada uno de ellos. Para una mejor usabilidad es necesario que el listado se encuentre paginado y ordenado alfabéticamente permitiendo además filtrar los datos.
- ***Insertar:*** deberá mostrar un formulario al usuario solicitándole los datos necesarios, una vez introducidos los datos el módulo debe validar que estén correctos y en caso contrario se mostrarán los mensajes de errores emitidos; si los datos son correctos entonces los datos serán almacenados en la base de datos y se emitirá un mensaje especificando que la acción se realizó correctamente.
- ***Editar:*** al seleccionar esta opción el módulo deberá mostrar un formulario solicitándole al usuario los datos necesarios, inicialmente dicho formulario se debe mostrar con los datos almacenados para el elemento seleccionado. Una vez introducidos los nuevos datos el módulo debe validar que se encuentren correctos y en caso contrario se mostrarán los mensajes de errores emitidos; si todo se encuentra correcto entonces los datos serán almacenados en la base de datos y se emitirá un mensaje especificando que la acción se realizó correctamente.
- ***Activar/Desactivar:*** Esta acción se encuentra en el listado de elementos, después de acceder a una determinada gestión. Al activar o desactivar un elemento se emitirá un mensaje especificando que la acción se realizó correctamente.

Para gestionar los usuarios es necesaria la gestión de los siguientes nomencladores:

- Color de ojos
- Color de piel
- Municipio
- Provincia
- Sexo

2.3 Especificación de los requisitos de software

2.3.1 Requisitos funcionales(RF)

RF1: Gestionar usuarios.

RF2: Gestionar nomencladores.

RF3: Gestionar roles.

RF4: Gestionar permisos.

RF5: Gestionar áreas.

RF6: Gestionar cargos.

2.3.2 Requisitos no funcionales

Requisitos no funcionales de software para estaciones clientes

RnF1. Sistema Operativo Linux o Windows.

RnF2. Navegador web Mozilla Firefox v17.0 o superior, Google Chrome v20.0 o superior, o versiones actuales de Opera, Internet Explorer y Safari.

Requisitos no funcionales de software para estaciones servidores

RnF3. Sistema operativo Linux o Windows.

RnF4. PostgreSQL 9.4.

RnF5. Python 3.5.

Requisitos no funcionales de hardware para estaciones clientes

RnF6. PC Pentium 4 a 1GHz o superior, mínimo 512 MB de RAM.

Requisitos no funcionales de hardware para estaciones servidores

RnF7. PC Pentium 4 a 2 GHz o superior, mínimo 2 GB de RAM, 20 GB o superior de disco duro.

Requisitos no funcionales de usabilidad

RnF8. Facilidad de uso por parte de los usuarios: el módulo debe presentar una interfaz amigable que permita la fácil interacción con el mismo y llegar de manera rápida y efectiva a la información buscada. Además, debe ser una interfaz de manejo cómodo que posibilite a los usuarios sin experiencias una rápida adaptación.

RnF9. El módulo podrá ser utilizado por cualquier usuario con las siguientes características:

- Conocimientos básicos relativos al uso de una computadora.
- Conocimientos sólidos relativos a los procesos de negocio acorde al rol que desempeñe.

RnF10. El módulo será distribuido en idioma español.

RnF11. Los términos utilizados se establecerán acorde al negocio correspondiente para facilitar la comprensión de la herramienta de trabajo.

RnF12. El módulo poseerá estructura y diseño homogéneos en todas sus pantallas, que facilite la navegación:

- Menús laterales y desplegables que permitan el acceso rápido a la información por parte de los usuarios.
- Restricciones en el diseño y la implementación.

RnF13. Lenguaje de Programación: Python.

RnF14. Plataforma de desarrollo: JetBrains PyCharm 2017 2.2.

RnF15.Marco de trabajo: Django 1.10.

RnF16. Para el acceso a datos se utilizará Django ORM.

RnF17. Para el modelado de UML 2.0 se utilizará Visual Paradigm 8.0.

RnF18. Como Gestor de base de datos se utilizará PostgreSQL 9.4.

Requisitos no funcionalidades de seguridad

RnF19. La seguridad se define a nivel de roles, con el fin de mantener la integridad de los datos en función del acceso de cada uno de ellos, trayendo consigo, además, la protección de la información.

RnF20. Autenticación segura para acceder a la aplicación.

Requisitos no funcionalidades de fiabilidad

RnF21. Si ocurren errores en el módulo no se mostrarán detalles de los mismos al cliente

2.4 Historias de usuario:

TABLA 1: HISTORIA DE USUARIO DEL RF 1(ELABORACIÓN PROPIA)

Número: HU-1	**Nombre de requisito:** Gestionar Usuario
Programador: Evelyn Morales	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo:** 32 horas
Riesgo: Alto	**Tiempo real:** 24 horas
Descripción: Se crea un usuario con contraseña y se define el rol que desempeña. Se tendrá la posibilidad de crear, mostrar, editar, activar y desactivar los usuarios.	
Observaciones: no se podrá dejar ninguno de los campos en blanco Se debe repetir la contraseña obligatoriamente.	
Prototipo de interfaz:	

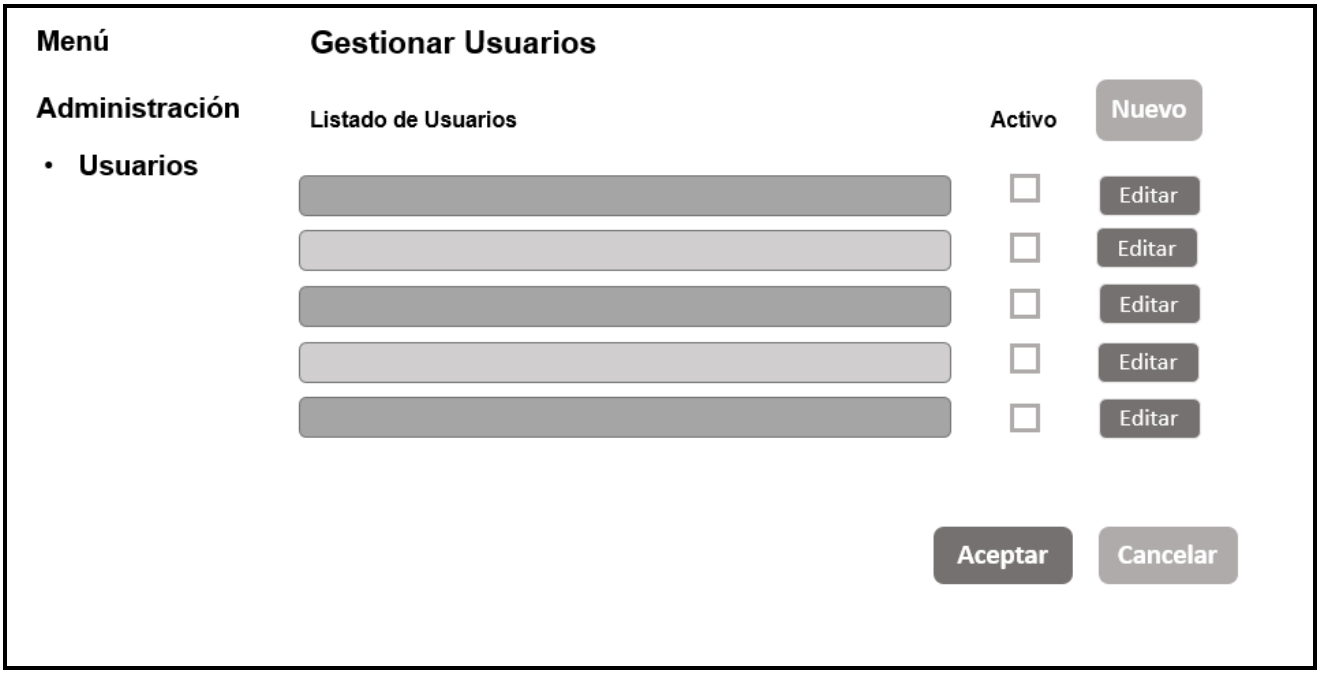


TABLA 2:HISTORIA DE USUARIO DEL RF 2 (ELABORACIÓN PROPIA)

Número: HU-2	**Nombre de requisito:** Gestionar Nomencladores
Programador: Evelyn Morales	**Iteración Asignada:** 2
Prioridad: Media	**Tiempo:** 46 horas
Riesgo: Medio	**Tiempo real:** 38 horas
Descripción: Se define el color de piel, color de ojos, sexo, municipio y provincia de las personas. Se tendrá la posibilidad de crear, mostrar, editar, activar y desactivar cada uno de los nomencladores.	
Observaciones: No se podrá dejar ninguno de los campos en blanco.	
Prototipo de interfaz:	

TABLA 3: HISTORIA DE USUARIO DEL RF 3 (ELABORACIÓN PROPIA)

Número: HU-3	**Nombre de requisito:** Gestionar Roles
Programador: Evelyn Morales	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo:** 24 horas
Riesgo: Medio	**Tiempo real:** 16 horas
Descripción: Se definen los roles de los usuarios. Se tendrá la posibilidad de crear, mostrar, editar, activar y desactivar cada rol.	
Observaciones: Cada usuario debe tener al menos un rol.	
Prototipo de interfaz:	

TABLA 4:HISTORIA DE USUARIO DEL RF 4 (ELABORACIÓN PROPIA)

Número: HU-4	**Nombre de requisito:** Gestionar Permisos
Programador: Evelyn Morales	**Iteración Asignada:** 4
Prioridad: Alta	**Tiempo:** 32 horas
Riesgo: Alto	**Tiempo real:** 23 horas
Descripción: Se definen los permisos de los usuarios según el rol que desempeñe. Se tendrá la posibilidad de crear, mostrar, editar, activar y desactivar cada permiso.	
Observaciones: No se puede dejar el campo en blanco	
Prototipo de interfaz:	

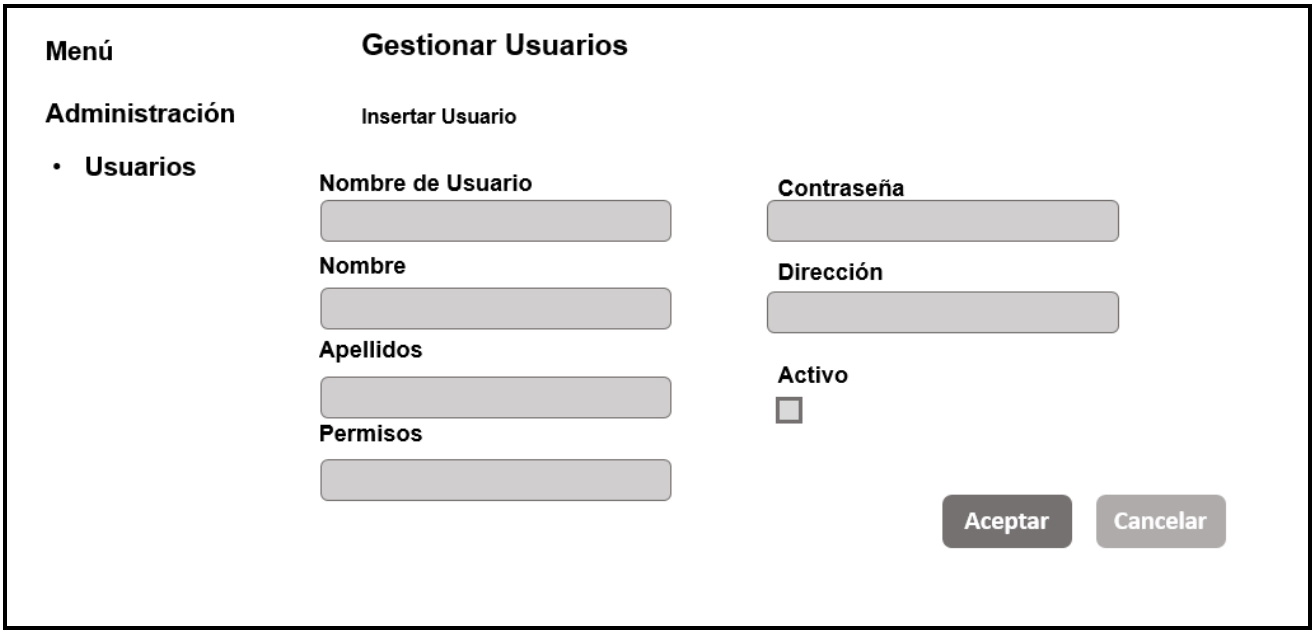


TABLA 5: HISTORIA DE USUARIO DEL RF 5 (ELABORACIÓN PROPIA)

Número: HU-5	**Nombre de requisito:** Gestionar Áreas
Programador: Evelyn Morales	**Iteración Asignada:** 5
Prioridad: Baja	**Tiempo:** 26 horas
Riesgo: Bajo	**Tiempo real:** 20 horas
Descripción: Se definen las áreas a las que pertenece cada usuario, así como su código de área y de quien es hija. Se tendrá la posibilidad de crear, mostrar, editar, activar y desactivar cada área	
Observaciones: Las áreas no podrán estar vacías.	

Prototipo de interfaz:

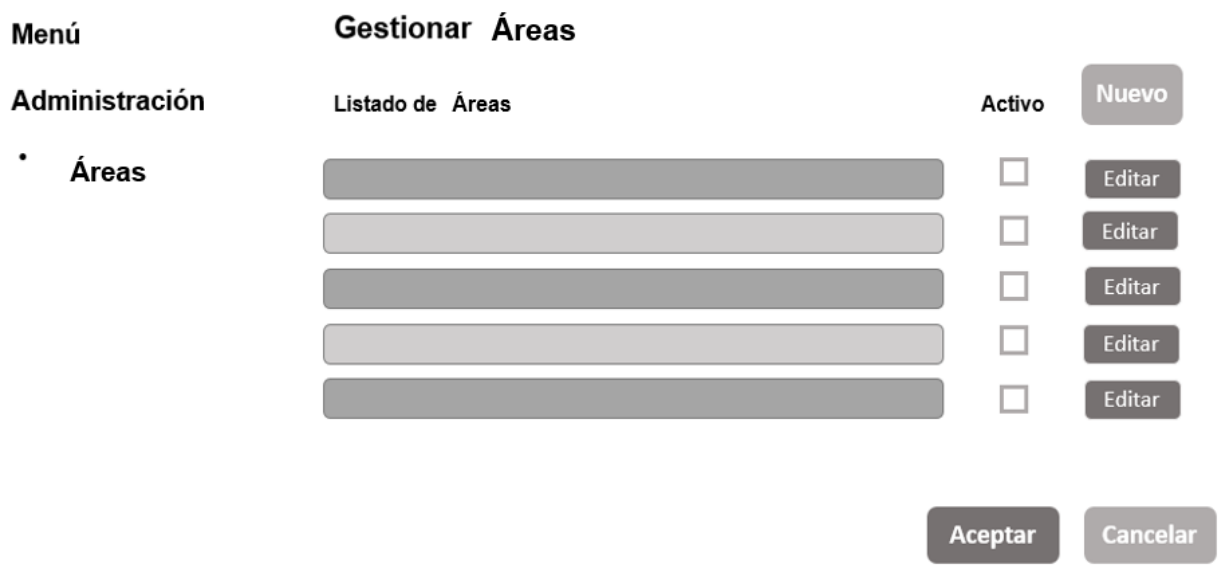
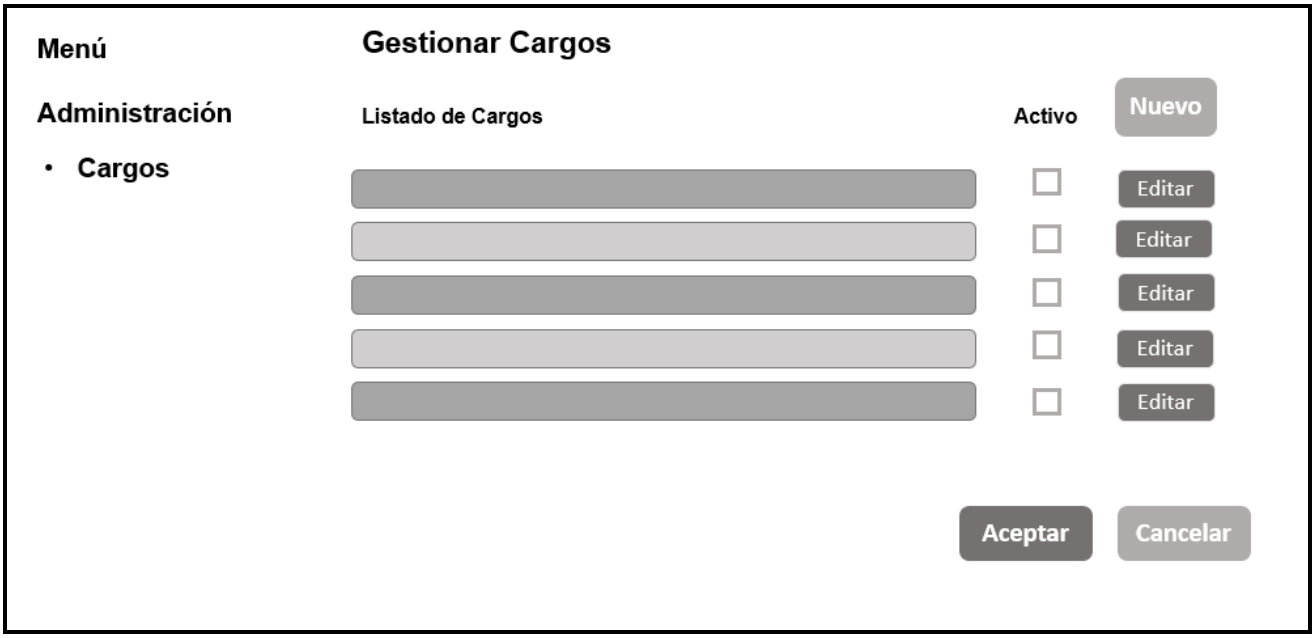


TABLA 6: HISTORIA DE USUARIO DEL RF 6 (ELABORACIÓN PROPIA)

Número: HU-6	**Nombre de requisito:** Gestionar Cargos
Programador: Evelyn Morales	**Iteración Asignada:** 6
Prioridad: Baja	**Tiempo:** 30 horas
Riesgo: Bajo	**Tiempo real:** 24 horas
Descripción: Se definen los cargos de los usuarios. Se tendrá la posibilidad de crear, mostrar, editar y desactivar cada cargo.	
Observaciones: Los usuarios no deben tener más de un cargo	
Prototipo de interfaz:	



2.5 Arquitectura del sistema:

"*La Arquitectura de Software es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema"* (Reynoso, 2004).La arquitectura basada en componentes se utiliza en aplicaciones compuestas de componentes individuales. Esta se enfoca principalmente en la descomposición del diseño en componentes lógicos o funcionales que muestren interfaces de comunicación bien definidas (Protocol reconfiguration using component-based design., 2005) (Fuentes, y otros, 2010).

Patrón Modelo-Vista-Controlador (MVC):

"El patrón MVC es un paradigma que divide las partes que conforman una aplicación en el Modelo, las Vistas y los Controladores, permitiendo la implementación por separado de cada elemento garantizando así la actualización y mantenimiento del software de forma sencilla y en un reducido espacio de tiempo. A partir del uso de Marcos de trabajos basados en el patrón MVC se puede lograr una mejor organización del trabajo y mayor especialización de los desarrolladores y diseñadores" (GONZÁLEZ, Yanette Díaz; Patrón Modelo-Vista-Controlador., 2012) (Patrón Modelo-Vista-Controlador., 2012).

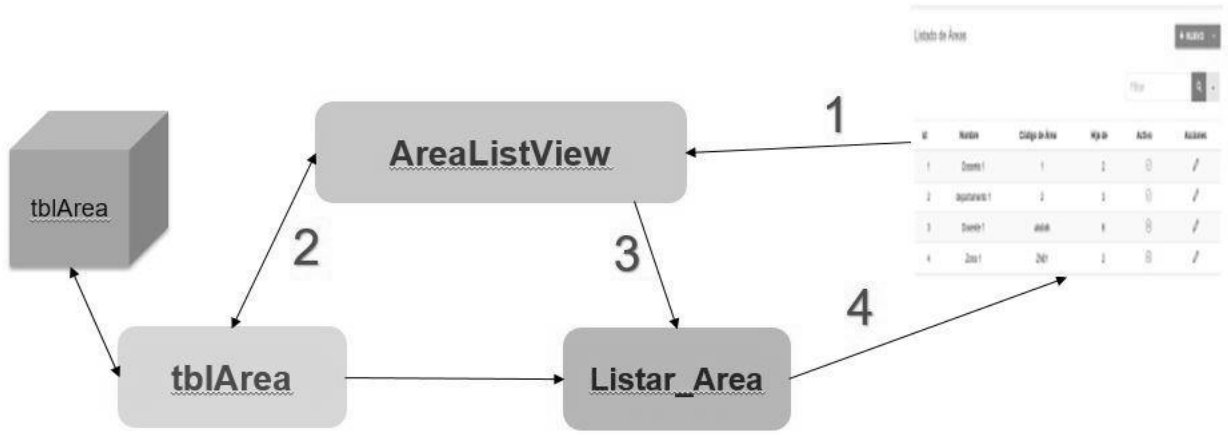


FIGURA 4: ARQUITECTURA MODELO-VISTA-PLANTILLA. CLASE TBLAREA (ELABORACIÓN PROPIA)

El sistema que se diseña sigue este estilo arquitectónico por estar compuesto por componentes individuales (módulos). Además, el usuario interactúa de una manera coherente con el sistema por lo que este patrón facilita la comunicación entre el modelo, la vista y la plantilla involucrados.

En el ejemplo anterior se muestra la interfaz de la aplicación desde el navegador, donde el usuario realiza una petición a la vista AreaListView (1), esta interactúa con el modelo tblArea (2) para obtener la estructura del objeto, que a su vez se comunica con la base de datos para obtener los datos de la tabla tblArea. Luego la vista intercambia la información con la plantilla Listar_Area (3) y esta renderiza los datos y le envía la respuesta al usuario (4).

2.6 Patrones de diseño:

"…un patrón de diseño describe una estructura de diseño que resuelve un problema particular del diseño dentro de un contexto específico y entre "fuerzas" que afectan la manera en la que se aplica y en la que se utiliza dicho patrón" (Pressman, 2010).

Los patrones de diseño ayudan a desarrollar con mayor velocidad, debido a que no es necesario pensar detalladamente sobre cómo resolver un asunto específico. El uso de estos patrones, nos proporciona la facilidad de prevenir problemas y mejorar la legibilidad y claridad del código. Así mismo es importante resaltar que los patrones de diseño nos proporcionan soluciones generales, las cuales no están atadas a un tipo específico de problema (ALPÍZAR, y otros, 2017) (Botero Tabares, 2010).

Entre los patrones a emplear en la construcción del actual sistema, se encuentran los GRASP (General *Responsibility Assignment Software Patterns*, en español, Patrones Generales de Asignación de Responsabilidades de Software), que son una mera codificación de los principios fundamentales de la asignación de responsabilidades a objetos (Larman, 2003), dichos patrones se muestran a continuación:

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, 2003). Este patrón se evidencia en el siguiente fragmento de código:

```
class tblArea(models.Model):
    idEntidad = models.ForeignKey(nEntidad)
    codigoArea=models.CharField(max_length=50,validators=[validate_only_letters_numbers])
    hijade = models.IntegerField(validators=[validate_only_numbers])
    nombre = models.CharField(max_length=150, validators=[validate_only_letters])
    activo = models.BooleanField(default=True)
```

FIGURA 5: FRAGMENTO DE CÓDIGO. CLASE TBLÁREA. (ELABORACIÓN PROPIA)

La clase tbÁrea (Figura 5) es la que conoce la información para la creación de un objeto Área, solo a través de ella se puede acceder a los atributos característicos de un “Área”.

Creador: La creación de objetos es una de las actividades más frecuentes en un sistema orientado a objetos. En consecuencia, conviene contar con un principio general para asignar las responsabilidades concernientes a ella. El propósito fundamental de este patrón es encontrar un creador que se debe conectar con el objeto producido en cualquier evento. El diseño bien asignado, puede soportar un bajo acoplamiento, una mayor claridad, el encapsulamiento y la reutilización (Larman, 2003).

Controlador: Un controlador es un objeto de interfaz no destinada al usuario que se encarga de manejar un evento del sistema. Define además el método de su operación. Una consecuencia importante del patrón controlador es que las operaciones del sistema deberían manejarse en la capa de dominio de los objetos y no en las de interfaz, presentación o aplicación. Esto soporta la reutilización de la lógica para manejar los procesos afines del negocio en aplicaciones futuras (Larman, 2003).

```python
class AreaCreateView(RequiredSecurityMixin, SuccessMessageMixin, CreateView):
    need_login = True
    permission = RequiredSecurityMixin.CREATE
    model = tblArea
    template_name = 'crear.html'
    form_class = CrearAreaForms
    success_url = reverse_lazy('Area_list')
    success_message = "Se registró el Área satisfactoriamente."

    def get_context_data(self, **kwargs):
        context = super(AreaCreateView, self).get_context_data(**kwargs)
        context['cancelar'] = self.success_url
        context['action'] = "Insertar"
        context['info_breadcrum'] = "Lista de Área"
        context['info_panel'] = "Área"
        context['module'] = "Área"

        return context

    def form_invalid(self, form):
        messages.error(self.request, 'Por favor corrija los errores.')
        return super(AreaCreateView, self).form_invalid(form)

    def get_success_url(self):
        register_logs(self.request, self.model, self.object.pk, force_str(self.object), ADDITION)
        return super(AreaCreateView, self).get_success_url()
```

FIGURA 6: FRAGMENTO DE CÓDIGO. CLASE ÁREACREATEVIEW. (ELABORACIÓN PROPIA)

El uso de los patrones Creador y Controlador se evidencian en la clase ÁreaCreateView (Figura 6), esta clase es la encargada de interactuar como mediadora entra la interfaz y el modelo para insertar los datos de una nueva área, así como la creación de una instancia de la clase tblÁrea.

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, 2003).

Este patrón ya viene incluido en el marco de trabajo de desarrollo *Django,* que permite un bajo acoplamiento entre las piezas, lo que evita las dependencias, por ejemplo, a la hora de realizar cambios en las configuraciones de las URL, en la Base de Datos, plantillas HTML, etc., basta solo con realizarlo una sola vez.

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, 2003).

```
class UsuarioCreateView(RequiredSecurityMixin, SuccessMessageMixin, CreateView):
    need_login = True
    permission = RequiredSecurityMixin.CREATE
    model = User
    template_name = 'crear.html'
    form_class = CrearUsuarioForms
    success_url = reverse_lazy('Usuario_list')
    success_message = "Se registró el Usuario %(username)s satisfactoriamente."
```

FIGURA 7: FRAGMENTO DE CÓDIGO. CLASE USUARIOCREATEVIEW. (ELABORACIÓN PROPIA).

```
class CrearUsuarioForms(forms.ModelForm):
    class Meta:
        model = User
        fields =['username','password','first_name','is_superuser','last_name','email', 'is_staff', 'is_active','user_permissions']
        widgets ={
            'username': TextInput(attrs={'class': 'form-control tooltips'}),
            'first_name': TextInput(attrs={'class': 'form-control tooltips'}),
            'last_name': TextInput(attrs={'class': 'form-control tooltips'}),
            'email': EmailInput(attrs={'class': 'form-control tooltips'}),
            'is_superuser': forms.CheckboxInput(attrs={'class': 'form-control tooltips'}),
            'is_active': forms.CheckboxInput(attrs={'class': 'form-control tooltips'}),
            'is_staff': forms.CheckboxInput(attrs={'class': 'form-control tooltips'}),
```

FIGURA 8: FRAGMENTO DE CÓDIGO. CLASE CREARUSUARIOFORMS. (ELABORACIÓN PROPIA).

En la presente solución es evidente la alta cohesión de las clases, como podemos apreciar entre las clases UsuarioCreateView (Figura 7) y CrearUsuarioForms (Figura 8). En este caso la clase UsuarioCreateView realiza una llamada a la clase CrearUsuarioForms para conocer la estructura que debe seguir para la inserción de un nuevo “Usuario”.

2.7 Diagrama de Clases:

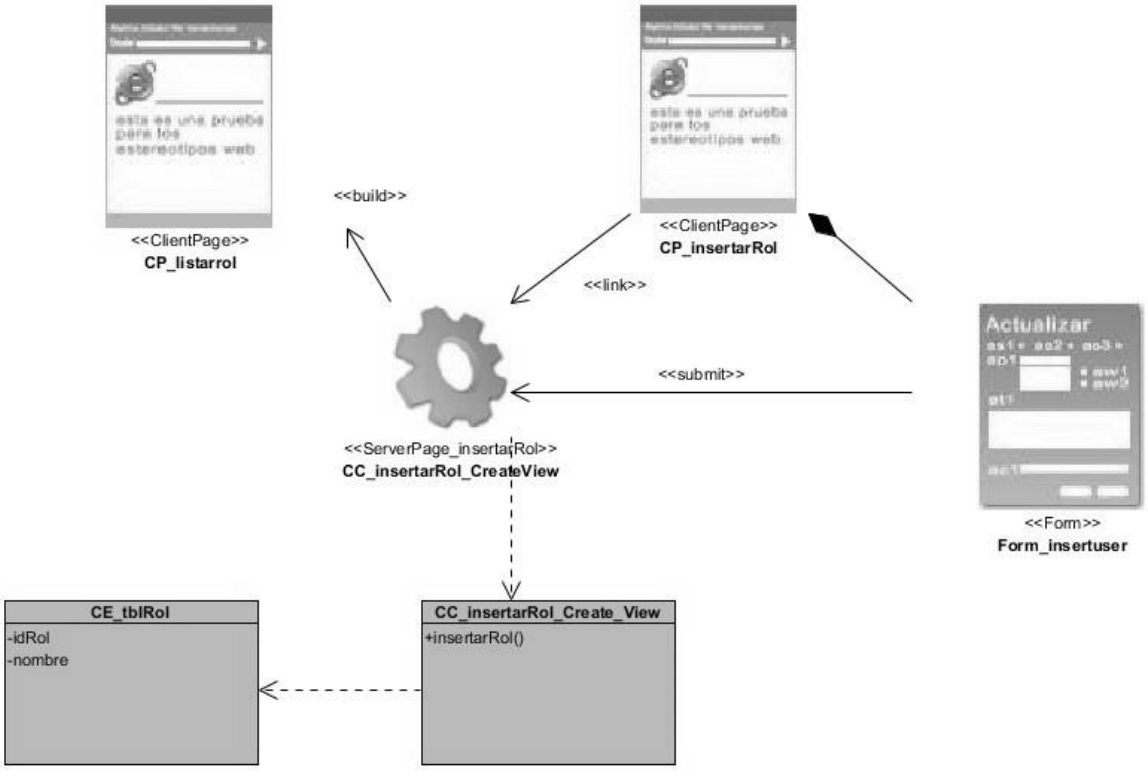


FIGURA 9: DIAGRAMA DE CLASES. (ELABORACIÓN PROPIA).

2.9 Modelo de datos:

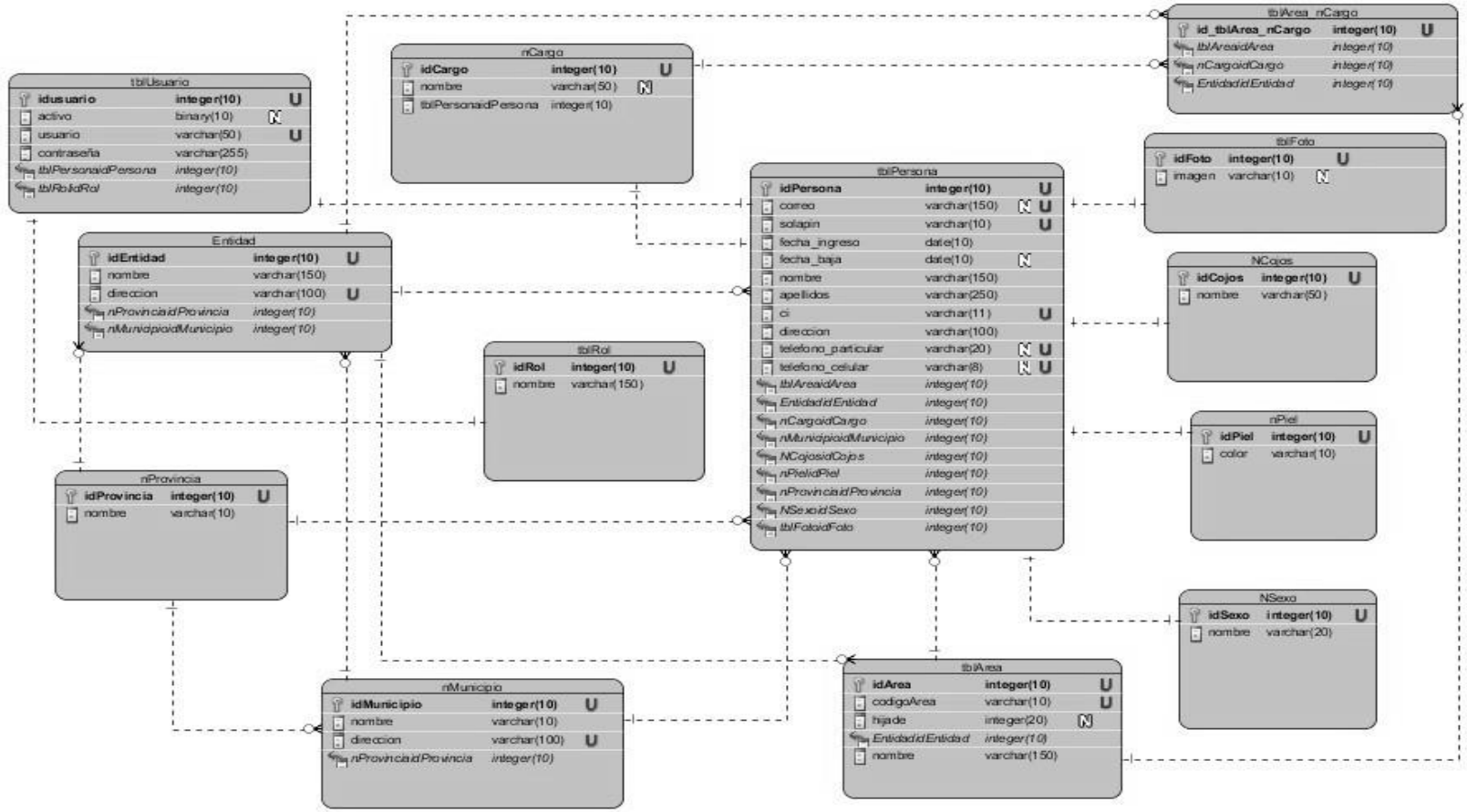


FIGURA 10: DIAGRAMA DE MODELO DE DATOS. (ELABORACIÓN PROPIA)

2.10 Diagrama de despliegue:

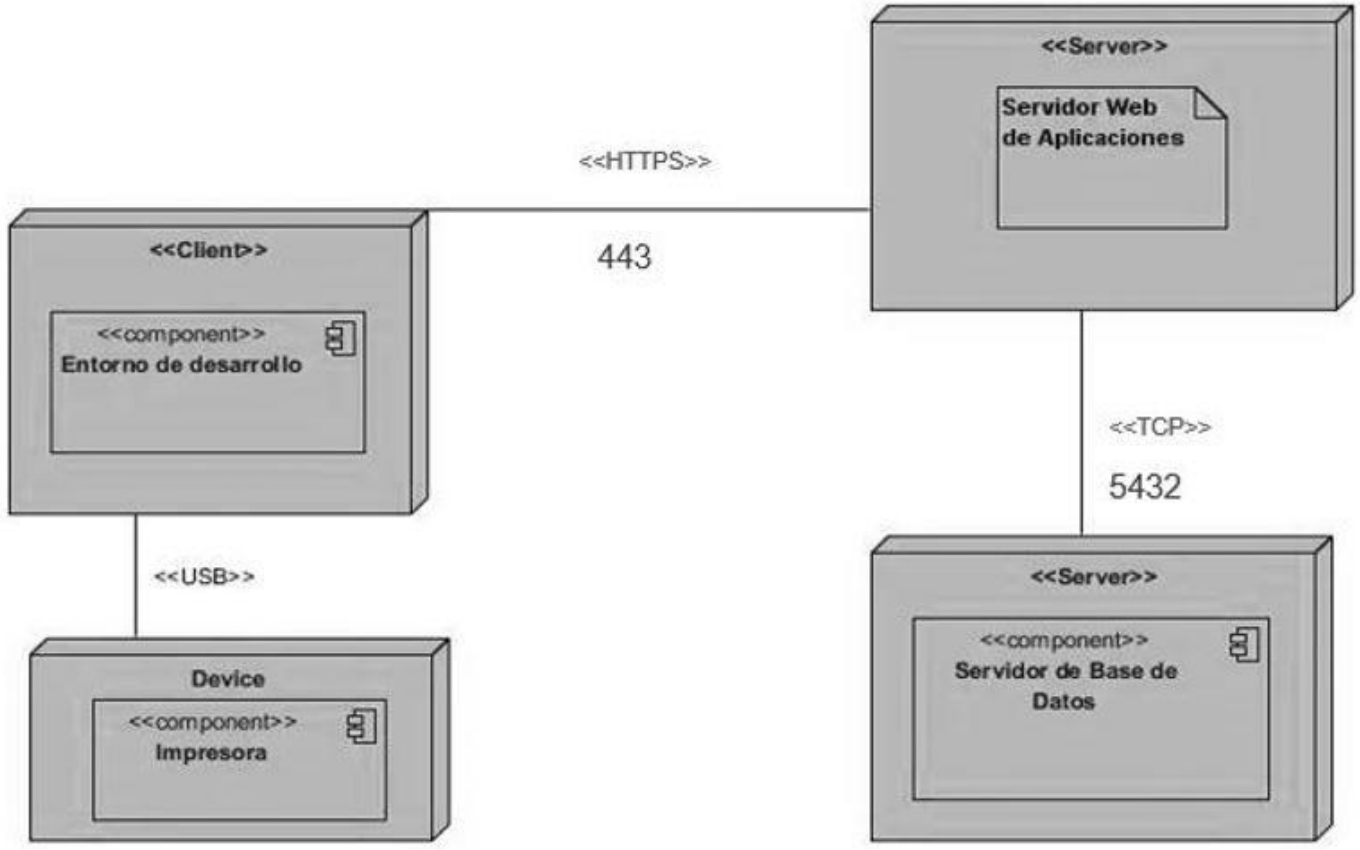


FIGURA 11: DIAGRAMA DE DESPLIEGUE. (ELABORACIÓN PROPIA)

<<HTTPS>>: Hypertext Transfer Protocol Secure o HTTPS (en español protocolo de transferencia segura de hipertexto). Es un protocolo basado en el protocolo HTTP, establece a través del puerto 443 la conexión segura entre el dispositivo de acceso cliente y el servidor de aplicaciones. La conexión es por cable vía modem, Local Area Network (LAN) o red inalámbrica con una velocidad de más de 64 Kbps.

<<TCP>>: estos protocolos establecen la conexión entre el servidor de aplicaciones y el servidor de base de datos. Para el servidor de base de datos de PostgresSQL se define el puerto 5432. La conexión entre el servidor web y el servidor de base de datos permite dar órdenes y obtener información de esta.

2.11 Conclusiones del capítulo:

- ✓ El estudio de los procesos de negocio actuales con respecto a la administración de los sistemas de

información de personas permitió tener una mejor comprensión los mismos.

- ✓ El modelado del negocio posibilitó definir los requisitos funcionales y no funcionales para el Módulo de Administración.
- ✓ El desarrollo de las HU facilitó la administración de los requisitos funcionales del Módulo de Administración.

Capítulo 3: Implementación y prueba:

En este capítulo se muestra cómo ha sido implementada la aplicación basándose en el diagrama de componentes y sus principales funcionalidades. Se describen los procesos de pruebas para identificar y erradicar posibles errores del módulo.

3.1 Implementación

"*La etapa de implementación del software es el proceso de convertir una especificación del sistema en un sistema ejecutable*" (Somerville, 2005). Esta fase comprende la materialización, en forma de código, de todos los artefactos, descripciones y arquitectura propuestos en la etapa de análisis y diseño; con el objetivo de conformar el producto final requerido por el cliente (Larman, 2003).

Durante la implementación del módulo se define como va estar ordenado el código, se integran los resultados en un sistema factible, se implementan los elementos del diseño y se distribuyen físicamente asignando componentes ejecutables a nodos. El objetivo de esta actividad es programar la solución y desarrollar de forma iterativa e incremental un producto completo.

3.2 Estándares de codificación

Los estándares de codificación acordados para el desarrollo de la presente solución son los definidos en el documento PEP 8—Style Guide for Python Code, por Guido van Rossum (Rossum, 2009). A continuación, quedan definidos:

Indentación

- Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes).
- Utilizar una indentación de una tabulación para cada línea con excepción de la primera.
- La indentación se realizará solamente con tabulaciones, no debe utilizarse nunca los cuatro (4) espacios.

Máxima longitud entre líneas:

- Todas las líneas deben estar limitadas a un máximo de setenta y nueve caracteres.
- Dentro de paréntesis, corchetes o llaves se puede utilizar la continuación implícita para cortar las

líneas largas.

- En cualquier circunstancia se puede utilizar el carácter “\” para cortar las líneas largas.

Líneas en blanco:

- Separar las funciones de alto nivel y definiciones de clases con dos líneas en blanco.
- Las definiciones de métodos dentro de una clase deben separarse por una línea en blanco.
- Se puede utilizar líneas en blanco escasamente para separar secciones lógicas.

Codificaciones:

- Utilizar la codificación UTF-8.
- Se pueden incluir cadenas que no correspondan a esta codificación utilizando “\x”, “\u” o “\U”.

Importaciones:

- Las importaciones deben estar en líneas separadas.
- Siempre deben colocarse al comienzo del archivo.

Deben quedar agrupadas de la siguiente forma:

1. Importaciones de la librería estándar.
2. Importaciones terceras relacionadas.
3. Importaciones locales de la aplicación/librerías.
   - Cada grupo de importaciones debe estar separado por una línea en blanco.
   - Evitar utilizar espacios en blanco en las siguientes situaciones:
4. Inmediatamente dentro de paréntesis, corchetes y llaves.
5. Inmediatamente antes de una coma, un punto y coma o dos puntos.
6. Antes del paréntesis que comienza la lista de argumentos en la llamada a una función.
7. Inmediatamente antes de un corchete que empieza una indexación.
8. Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.

Espacios en blancos en expresiones y sentencias:

- Deben rodearse con exactamente un espacio los siguientes operadores binarios:

1. Asignación (=).
2. Asignación de aumentación (+=, -=, etc.).
3. Comparación (==, <, >,>=, <=, !=, <>, in, not in, is, is not).
4. Expresiones lógicas (and, or, not).

- Si se utilizan operadores con prioridad diferente se aconseja rodear con espacios a los operadores de menor prioridad.
- No utilizar espacios alrededor del igual (=) cuando es utilizado para indicar un argumento de una función o un parámetro con un valor por defecto.

Comentarios:

- Los comentarios deben ser oraciones completas.
- Si un comentario es una frase u oración su primera palabra debe comenzar con mayúscula a menos que sea un identificador que comience con minúscula.
- Nunca cambiar las minúsculas y mayúsculas en los identificadores de clases, objetos, funciones, etc.
- Si un comentario es corto el punto final puede omitirse.

Cadenas de documentación:

- Deben quedar documentados todos los módulos, funciones, clases y métodos públicos.
- Para definir una cadena de documentación debe quedar encerrada dentro de (“”).
- Los (“”) que finalizan una cadena de documentación deben quedar en una línea a no ser que la cadena sea de una sola línea.

Convenciones de nombramiento:

- Nunca se deben utilizar como simples caracteres para nombres de variables los caracteres ele minúscula “l”, o mayúscula “O”, ele mayúscula “L” ya que en algunas fuentes son indistinguibles de

los números uno y cero.

- Los módulos deben tener un nombre corto y en minúscula.
- Los nombres de clases deben utilizar la convención “CapWords” (palabras que comienzan con mayúsculas).
- Los nombres de las excepciones deben estar escrito también en la convención “CapWords” utilizando el sufijo “Error”.
- Los nombres de las funciones deben estar escrito en minúscula separando las palabras con un guión bajo “_”.
- Las constantes deben quedar escritas con letras mayúsculas separando las palabras por un guión bajo (_).

3.3 Diagrama de componente:

El principal objetivo del diagrama de componentes es mostrar las relaciones estructurales entre los componentes de un sistema. Este diagrama permite verificar que estos están implementando la funcionalidad requerida de un sistema, asegurando así que el mismo será aceptable. Los diagramas de componentes generalmente están orientados hacia el personal de implementación de un sistema, este presenta una comprensión temprana del sistema general que se está construyendo, proporciona una vista arquitectónica de alto nivel del sistema que será construido, ayuda a los desarrolladores a tomar decisiones sobre asignaciones de tareas y / o mejoras de habilidades necesarias para el sistema y se obtienen una vista temprana del software lógico se ejecutará (Bell, 2004).

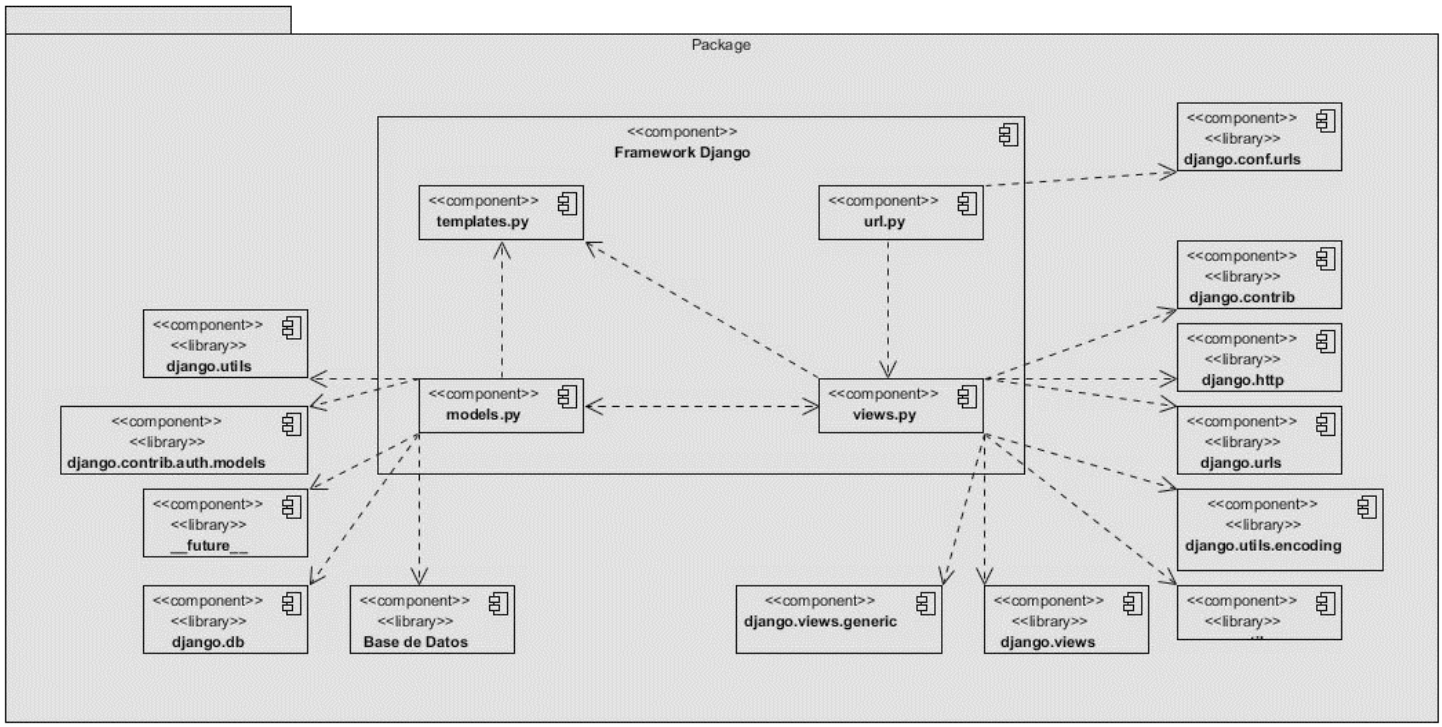


FIGURA 11: DIAGRAMA DE COMPONENTES. (ELABORACIÓN PROPIA).

3.4 Pruebas:

La prueba de una aplicación web es una colección de actividades relacionadas con una sola meta: descubrir errores en el contenido, función, utilidad, navegabilidad, rendimiento, capacidad y seguridad de esa aplicación. Para lograr esto, se aplica una estrategia de prueba que abarca tanto revisiones como pruebas ejecutables (Pressman, 2010).

3.4.1 Funcionales:

Las pruebas funcionales se aplican a un software determinado, con el objetivo de validar que las funcionalidades implementadas se desempeñen de acuerdo a las especificaciones de los requisitos definidos con anterioridad. Al conocer las funciones específicas que se le asignaron a una aplicación para su realización, pueden llevarse a cabo casos de pruebas que demuestren que cada función es completamente operativa mientras que al mismo tiempo se buscan errores en cada función (Pressman, 2010).

Resultados de las pruebas funcionales:

TABLA 7: VARIABLES PARA LOS CASOS DE PRUEBA FUNCIONAL GESTIONAR ÁREA

No.	Nombre del campo	Clasificación	Valor nulo	Descripción	Alias para el caso de prueba
1	Área	Campo de texto	No	Se introduce el nombre del área	NA
2	Código de Área	Campo alfanumérico	No	Se introduce el código del área	CA
3	Hija de	Campo numérico	No	Se introducen los apellidos de la persona que se crea el usuario	HD
4	Activo	CheckBox	Sí	Se selecciona si el área esta activa	AA
5	Entidad	Campo seleccionable	No	Se seleccionan la entidad a la que pertenece el área	EA

TABLA 8: CASO DE PRUEBA FUNCIONAL INSERTAR ÁREA

Descripción General								
Permitirá crear una nueva área en el sistema								
Condiciones de ejecución								
Para crear una nueva área el usuario debe estar logueado en el sistema y poseer los permisos necesarios.								
Escenario	**Descripción**	**NA**	**CA**	**HD**	**AA**	**EA**	**R/ del sistema**	**Flujo central**
ESC 1.1 Introducir datos	El usuario con los permisos correspondie	Docente1	DC01	4	Sí	Universidad	El sistema debe crear un área satisfactoria	En el menú a la izquierda se sigue la siguiente ruta:

incorrectamente	ntes llena todos los campos del formulario correctamente						mente y mostrar una notificación de éxito.	“Administración-Área”. Aparecerá un listado con las áreas existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear una nueva área. Una vez insertados los datos del área se presiona el botón *Aceptar*. Si no desea crear el área presione el botón *Cancelar* para volver al listado de

								áreas existentes. Una vez insertada el área debe mostrarse una lista con todas las áreas incluyendo la insertada.
ESC 1.2 Campos vacíos	El usuario deja uno o más campos vacíos.	Vacío	Vacío	Vacío	Sí	Vacío	El sistema muestra un mensaje de error: ”Este campo es requerido”	En el menú a la izquierda se sigue la siguiente ruta: “Administración-Área”. Aparecerá un listado con las áreas existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondien

								tes para crear una nueva área. Una vez insertados los datos del área se presiona el botón *Aceptar*. Si no desea crear el área presione el botón *Cancelar* para volver al listado de áreas existentes. Una vez insertada el área debe mostrarse una lista con todas las áreas incluyendo la insertada.
ESC 1.3 Ingreso de caracteres inválidos	El usuario introduce caracteres inválidos en los campos.	☺♠❤♣♂♀ ♪♫	DC01	cuatro	No	Universidad	El sistema muestra un mensaje de error: 1- Si es de solo	En el menú a la izquierda se sigue la siguiente ruta:

							números: “El campo debe tener solo números” 2- Si es de texto: “El campo debe tener solo letras” 3- Si es alfanumérico: “El campo debe tener solo letras y/o números”	“Administración-Área”. Aparecerá un listado con las áreas existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear una nueva área. Una vez insertados los datos del área se presiona el botón *Aceptar*. Si no desea crear el área presione el botón *Cancelar* para volver al listado de

								áreas existentes. Una vez insertada el área debe mostrarse una lista con todas las áreas incluyendo la insertada.

Resultados de las pruebas funcionales:

Para la realizar la prueba de los requisitos funcionales se llevaron a cabo 3 iteraciones, donde fueron detectadas un total de 16 No Conformidades (NC). En la primera iteración se detectaron 13 NC, de las cuales se resolvieron 9 y quedaron pendiente 4. En la segunda iteración se detectaron 7 NC, de las cuales 3 eran nuevas y las otras 4 que quedaron pendiente de la iteración anterior, las cuales fueron resueltas en su totalidad, no quedando pendiente ninguna. En la tercera iteración no se detectó ninguna NC, por lo que se puede decir que el sistema funciona correctamente.

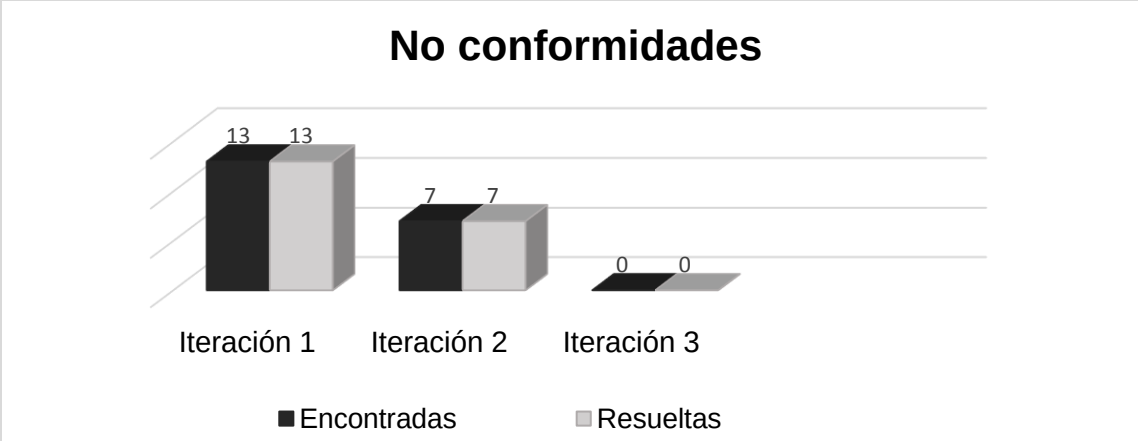


FIGURA 12: COMPORTAMIENTO DE LAS NO CONFORMIDADES POR CADA ITERACIÓN (ELABORACIÓN PROPIA).

Entre las No Conformidades detectadas en las pruebas funcionales se encuentran:

- Los campos de texto admiten números y caracteres extraños.
- Errores ortográficos en los nombres de los campos y en los mensajes.
- Los mensajes no describen el error o la acción que se está realizando sino otros diferentes.
- Los datos son guardados sin validar si tienen errores o no.

3.4.2 Pruebas de Integración:

Las pruebas de integración son una técnica sistemática para construir la estructura del programa, mientras que, al mismo tiempo, se llevan a cabo pruebas para detectar errores asociados con la interfaz. El objetivo es tomar los módulos probados mediante las pruebas de unidad y construir una estructura de programa que se haya dictado por diseño (Pressman, 2010).

Se realizan pruebas de integración al módulo para verificar la compatibilidad y el funcionamiento de las interfaces que comunican los componentes de la propuesta de solución con el SIPP.

Resultados de las pruebas de integración:

Para realizar esta prueba se integraron los cinco módulos y se pudo comprobar que no se ve afectado el funcionamiento de la aplicación. Se pudo controlar el acceso a través de los usuarios, con los permisos pertinentes, creados por el Módulo de Administración. Además, el Modulo de Administración puede supervisar la información a través de los *logs* y trazas del Módulo de Servicios En la siguiente imagen se puede apreciar el uso de los permisos por parte del Módulo de control de acceso.

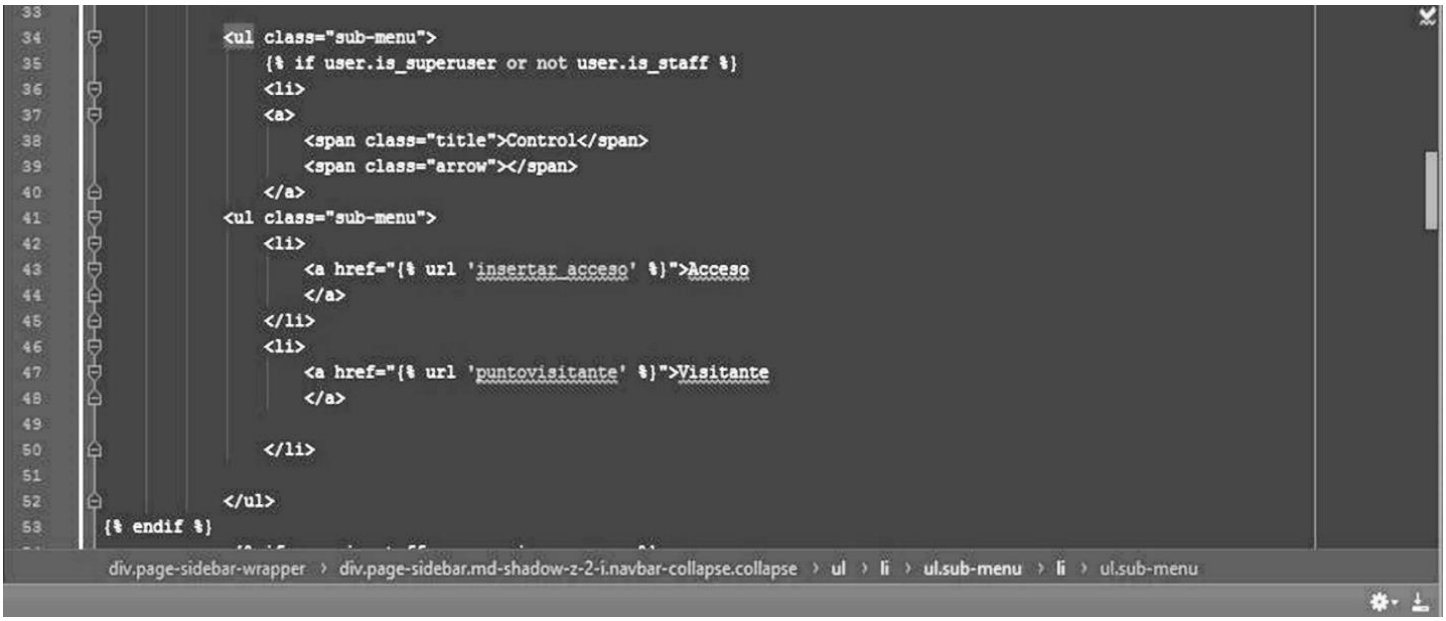


FIGURA 14: FRAGMENTO DE CÓDIGO

3.4.3 Seguridad:

Las pruebas de seguridad se realizan con la intención de explotar vulnerabilidades en el sistema o en su entorno. Estas intentan verificar que los mecanismos de protección que se construyen en un sistema en realidad lo protegerán de cualquier penetración impropia (Pressman, 2010).

La realización de pruebas de seguridad contribuye a la detección temprana de vulnerabilidades y la toma de medidas para la disminución de amenazas de ataque, y con ello proveer sistemas de cómputo más seguros y confiables.

Al módulo desarrollado se le realizaron una serie de pruebas de seguridad mediante el software Acunetix, las cuales se presentan a continuación:

- Ataques de inyección SQL.
- Falsificación de petición.
- Cross-Site Scripting.

Observaciones

La prueba realizada mediante la herramienta Acunetix, permitió detectar en la primera iteración varias vulnerabilidades que se muestran en la Figura 14.

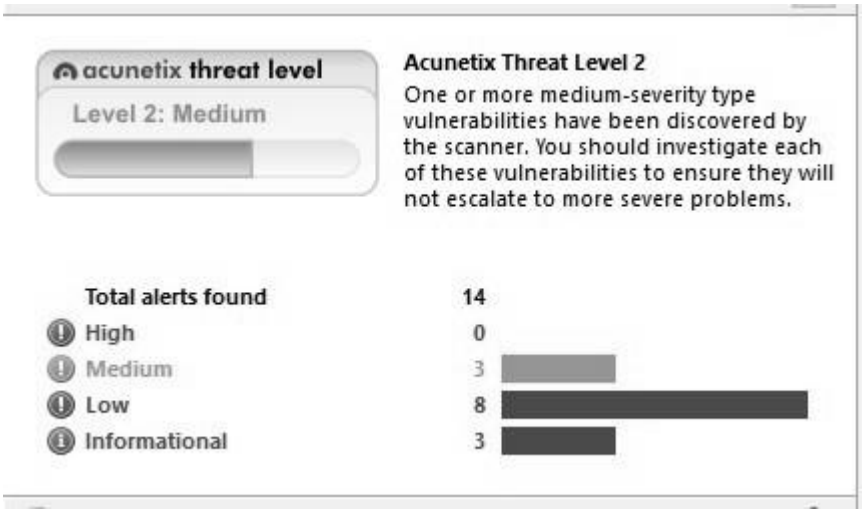


FIGURA 135: RESULTADO DE LA PRUEBA DE SEGURIDAD EN LA PRIMERA ITERACIÓN(ELABORACIÓN PROPIA).

Al realizar la segunda iteración se evidencia una disminución considerable de errores, tal y como se muestra en la Figura 15.

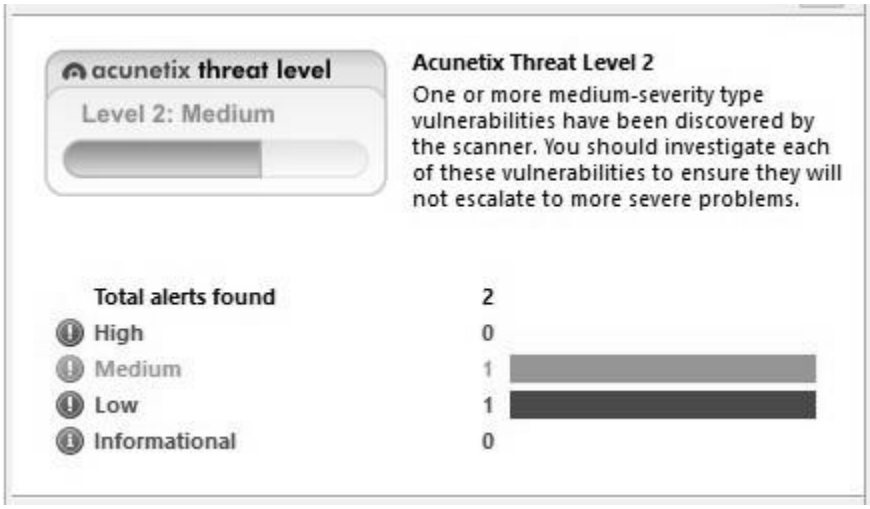


FIGURA 146: RESULTADO DE LA PRUEBA DE SEGURIDAD EN LA SEGUNDA ITERACIÓN(ELABORACIÓN PROPIA).

En la tercera iteración quedan eliminados todos los errores, quedando evidenciado que el sistema no presenta vulnerabilidades, como se muestra en la figura 16.

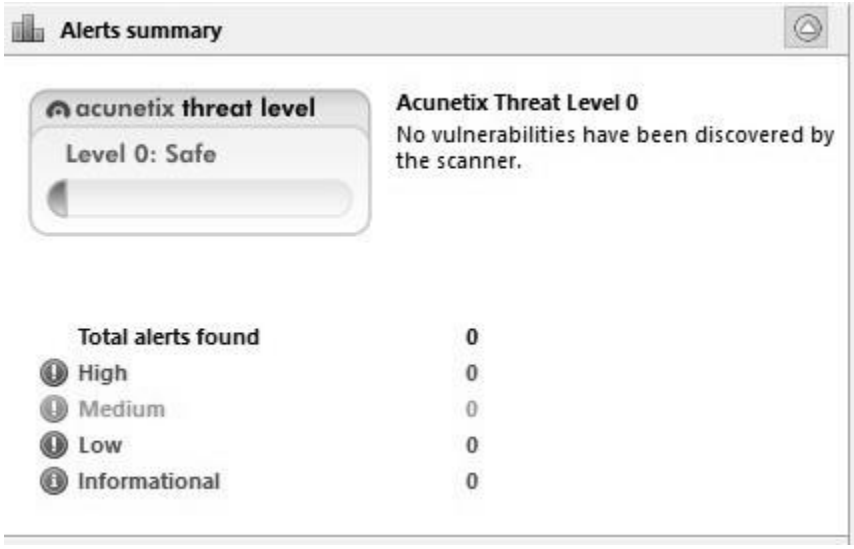


FIGURA 157: RESULTADO DE LA PRUEBA DE SEGURIDAD EN LA TERCERA ITERACIÓN(ELABORACIÓN PROPIA).

3.6 Validación de la Hipótesis

Para realizar la validación de la hipótesis de la investigación se utilizó el método de consulta a expertos, aplicando la variante del método Delphi, quedando definidas las siguientes fases:

1- Definición: A partir del problema de investigación se debe formular el objetivo de la consulta.

2- Conformación del grupo de informantes: Se define el perfil de los expertos y las condiciones para su selección.

3- Ejecución de las rondas de consulta: Se elabora el cuestionario que se aplicará a los expertos.

4- Resultados: Se analiza la información recogida y se elabora el informe de devolución final (REIRE, Revista d'Innovació i Recerca en Educació, 2016).

Definición de las características de selección de expertos:

Para la identificación de posibles expertos se seleccionaron trabajadores de la UCI que tuvieran conocimientos acerca del lenguaje de programación Python y conocimientos sobre tecnologías web. Además de la experiencia laboral y su disposición a participar en la encuesta.

Expertos identificados y seleccionados, evaluación de las características de selección

TABLA 9: EXPERTOS ENTREVISTADOS PARA LA VALIDACIÓN DE LA HIPÓTESIS DE LA INVESTIGACIÓN (ELABORACIÓN PROPIA).

No.	Experto	Entidad	Años de experiencia	Conocimientos Web
1	Osay González Fuentes	CISED	6	Si
2	Raul Manuel Azahares Reyes	CISED	4	Si
3	Denier Naranjo Oliva	CISED	5	Si
4	Daynelis Valdéz Monrabal	CISED	4	Si
5	Osmel Mojeda	CISED	1	Si

Posterior a la selección de los expertos se generaron un grupo de preguntas que permitirá validar el módulo de Administración para el SIPP. Estas preguntas están enfocadas principalmente al funcionamiento del módulo. Los expertos podrán expresar sus valoraciones en las siguientes categorías:

- Muy adecuado (MA).
- Adecuado (A).
- Poco adecuado (PA).
- Inadecuado (I).

Encuesta realizada:

1- El módulo garantiza la integridad de los nomencladores

2- El módulo permite gestionar los usuarios

3- El módulo es capaz de asignar permisos y roles a los usuarios

4- El módulo permite supervisar la información del SIPP

5- Considera que el módulo permite definir las distintas áreas de la entidad

Una vez aplicada la encuesta se pudo llegar al siguiente resultado:

Todos los especialistas verificaron que el módulo garantiza la integridad de los nomencladores creando un usuario que no posee permisos de administrador ni de edición de nomencladores y comprobaron que no podían cambiar la información de los mismos.

También se demostró que se podían gestionar los usuarios a través de este módulo, pudiendo un usuario con permisos de administrador crear, mostrar, editar, activar y desactivar usuarios del sistema.

Otro resultado en el que estuvieron de acuerdo los expertos fue que el módulo es capaz de asignar permisos y roles a los usuarios a través de un usuario con permisos de administrador.

El 100% de los expertos verificó que la información se supervisa en este módulo, pues este es el único de los cinco que tiene acceso a todos los módulos. Este supervisa la información a través del Módulo de Trazas, el cual registra los logs del sistema.

Por último, se comprobó que el Módulo de Administración permite definir las áreas de la entidad a través de un formulario.

Los resultados obtenidos evidencian una buena aceptación por parte de los expertos, calificando el Módulo de Administración de adecuado y muy adecuado.

3.7 Conclusiones del capítulo:

- El estudio de los estándares de codificación definidos para la implementación permitió desarrollar un código reutilizable.
- La realización del diagrama de componentes permitió mostrar las relaciones estructurales entre los componentes del Módulo de Administración.
- La realización de las pruebas funcionales posibilitó verificar que las funcionalidades implementadas en el Módulo de Administración se desempeñan de acuerdo a las especificaciones de los requisitos definidos con anterioridad.
- Las pruebas de seguridad permitieron explotar las vulnerabilidades existentes en el Módulo de Administración lo que propició la mejora de las mismas.

Conclusiones

- El estudio de los referentes teóricos y tendencias para los sistemas de identificación de personas permitió determinar las características que constituyen la base para el diseño de las funcionalidades que se definen en la propuesta de solución
- Se identificaron los patrones de diseño y arquitectura, que por sus características ofrecen mayor soporte a la implementación de los requisitos previamente expresados por el cliente; lo que garantizó una mejor comprensión del negocio y la estructura base para la organización lógica del código fuente.
- Se implementó el módulo teniendo en cuenta los estándares de codificación lo que disminuyó el impacto ante futuras modificaciones en la aplicación
- La realización de las pruebas de software permitió erradicar las insuficiencias detectadas en el módulo desarrollado, proporcionándole mayor seguridad, usabilidad, rendimiento y facilidad de uso de las funcionalidades presentes
- La implementación del Módulo de Administración logró la integridad de los nomencladores y la supervisión de la información del SIPP

Recomendaciones

A los especialistas del centro CISED analizar la posibilidad de adicionar la funcionalidad Gestionar Módulos para el SIPP.

Referencias

2017. Administration Module FHIR. [Online] 2017. https://www.hl7.org/fhir/administration-module.html.

ALPÍZAR, Jean Carlo Murillo, RODRIGUEZ, Carolina Oconitrillo and BOLAÑOS, Leylin Esquivel. 2017. *Patrones de diseño.* San Ramón : Universidad de Costa Rica, 2017.

Andreu, R and Ricart J. E. y Valor, J. 1991. *Estrategia y Sistemas de información Mc Graw-Hill.* Madrid : s.n., 1991.

Bell, Donald. 2004. UML basics: The component diagram. [Online] 2004. [Cited: abril 4, 2018.] http://www.softwareresearch.net/fileadmin/src/docs/teaching/WS13/SE/UML_basics-_The_component_diagram.pdf.

Botero Tabares, Ricardo. 2010. *Patrones Grasp y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación.* Bogotá : Entre Ciencia e Ingeniería, 2010. 8.

Canós, José H. and Letelier, Patricio y Panadés, Carmen M. Metodologías Agilesen el Desarrollo de Software. [Online] http://issi.dsic.upv.es/archives/f-1069167248521/actas.pdf.

Carceler, Victor. 2016. Institut Puig Castellar Santa Coloma de Gramenet. [Online] 2016. https://elpuig.xeill.net/Members/vcarceler/c1/didactica/apuntes/ud4/na1.

Educación en Ingeniería. **Medina, Lucy Nohemy and López, Wilmer Mesías. 2015.** Bogotá : s.n., 2015, Vols. vol. 10, No. 20.pp 98-109.

2017. eweek. [Online] 2017. [Cited: 11 27, 2017.] . http://www.eweek.com/development/jetbrains-strikes-python-developers-with-pycharm-1.0-ide.

Fuentes, Lidia, Troya, José M. and Vallecillo, Antonio. 2010. *Desarrollo de Software Basado en Componentes.* Málaga : Campus Teatinos, 2010.

Gallardo José E., García Carmen M. Apuntes para la asignatura Informática de la facultad de ciencias (Matemáticas) de la Universidad de Málaga. [Online] [Cited: 11 22, 2018.] http://www.lcc.uma.es/~pepeg/modula/temas/tema10.pdf.

GONZÁLEZ, Yanette Díaz; Patrón Modelo-Vista-Controlador. **GONZÁLEZ, Yanette Díaz and ROMERO, Yenisleidy Fernández. 2012.** no 1, La Habana : Revista Telem@ tica,, 2012, Vol. vol 11.

2017. IBM Knowledege Center. [Online] 2017. https://www.ibm.com/support/knowledgecenter/es/SSWGNW_10.1.0/com.ibm.swg.ba.cognos.ctrl_ug.10.1.1.doc/t_user rolesdef.html.

Island, Daniel. 2011. Blackbaud e Tapestry Admin Module. [Online] 2011. https://www.blackbaud.com/files/resources/downloads/Datasheet_eTapestry_AdminModule.pdf.

Larman, Craig. 2003. *Uml y patrones.* Madrid : Pearson educación S.A, 2003.

2012. *Manual de Usuario para el Siastema de Administración Naciona de la República de Cuba.* La Habana : s.n., 2012.

2017. Metodología ágil de desarrollo de software extremo (AMS_XP) y software libre (OSS). [Online] 2017. [Cited: enero 18, 2017.] http://www.monografias.com/trabajos67/metodologia-desarrollo-softwares/metodologia-desarrollo-softwares2.shtml.

2015. Metodología de desarrollo para la actividad preoductiva de la UCI. La Habana : s.n., 2015.

2017. mozilla.org. [Online] 2017. [Cited: 11 27, 2017.] . https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction.

Pablo, Amador Juan. Definición, objetivo, importancia y característica de la administración. [Online] https://www.google.com.cu/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0ahUKEwjmycmvmNPXAhUMyWMKHfouD_8QFggvMAI&url=http%3A%2F%2Ffaculty.ksu.edu.sa%2Fbelaichi%2FClases%2FTRADUCCI%25C3%2593N%2520ADMINISTRATIVA%2520%25D8%25A7%25D9%2584%25D8%25AA%25D8%25B1%2.

Patrón Modelo-Vista-Controlador. **Fernández Romero, Yenisleidy and Díaz González, Yanette. 2012.** 1, La Habana : Telemática, 2012, Vol. 11.

2017. PostgreSQL. [Online] 2017. https://www.postgresql.org/.

Pressman, Roger S. 2010. *Ingeniería del software.* New York : Mc Graw Educación, 2010. ISBN: 978-607-15-0314-5.

Protocol reconfiguration using component-based design. **FOUKALAS, Fotis. 2005.** Heidelberg : Springer, 2005.

REIRE, Revista d'Innovació i Recerca en Educació. **Reguant-Álvarez, Mercedes and Torrado-Fonseca, Mercedes. 2016.** 1, Barcelona : Universidad de Barcelona, Instituto de Ciencias de la Educación, 2016, Vol. 9.

Reynoso, Carlos. 2004. Introducción a la Arquitectura de Software. *Universidad de Buenos Aires.* 2004, Vol. 33.

Rossum, Guido Van. 2009. Python.org. [Online] 9 2009. [Cited: 11 27, 2017.] http://python.org.ar/pyar/Tutorial .

Sánchez, M, et al. Modelo de Control de Acceso en un Sistema. [Online] [Cited: febrero 13, 2018.] https://aipo.es/articulos/4/24.pdf.

Scott, Kendall. 1999. UML Gota a Gota. 1999.

Somerville, Ian. 2005. *Ingeniería del Software.* Madrid : Pearson Educacion S.A, 2005.

Usos y definiciones de los términos relativos a los usuarios o clientes. **Nuñes, Israel A. 2000.** 1-2, Medellín : Interam, 2000, Vol. 23.

Venero Paez, Diana and Valdés Gónzales, Alejandro. 2010. *Módulo de Administración para el Sistema Único de Identificación Nacional de la República de Cuba.* La Habana : s.n., 2010.

Vinay Bhagat. 2013-2018. Trust Radius. [Online] 2013-2018. [Cited: diciembre 15, 2017.] https://www.trustradius.com/products/etapestry/reviews.

2017. Visual Paradigm. [Online] 2017. https://www.visual-paradigm.com/support/faq.jsp.

Anexos

Anexo 1: *Modelo de control de Acceso RBAC*

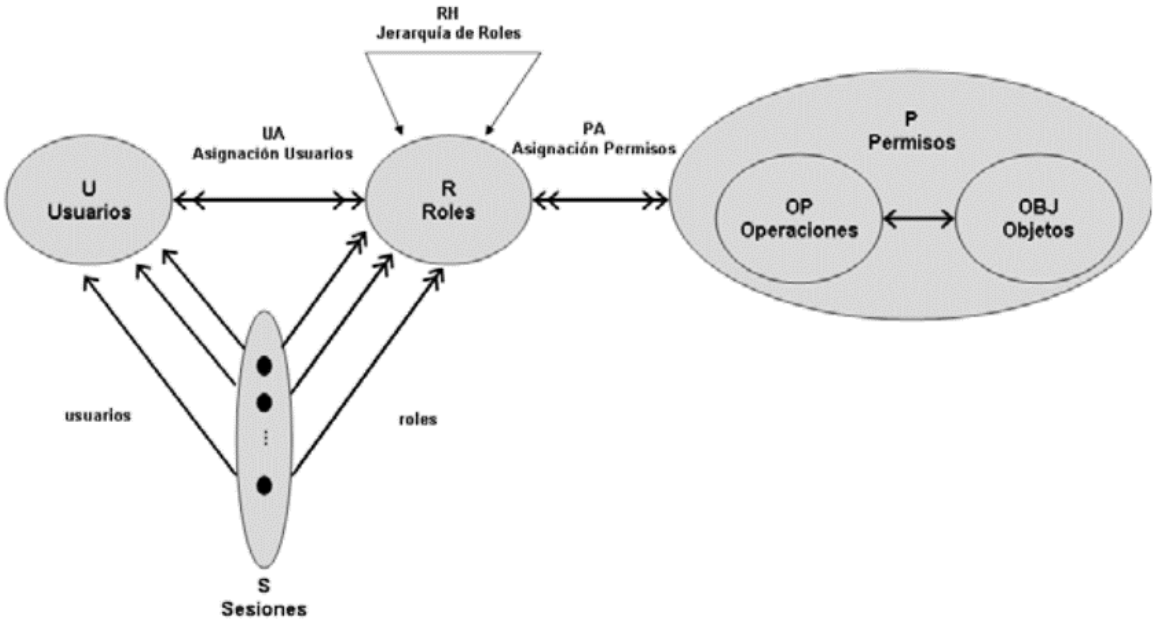


FIGURA 168: DIAGRAMA DEL MODELO DE CONTROL DE ACCESO RBAC (SÁNCHEZ, Y OTROS).

Anexo 2: Casos de pruebas funcionales

TABLA 10: CASO DE PRUEBA FUNCIONAL GESTIONAR CARGO (ELABORACIÓN PROPIA).

No.	Nombre del campo	Clasificación	Valor nulo	Descripción	Alias para el caso de prueba
1	Cargo	Campo de texto	No	Se introduce el nombre del cargo	NC
4	Activo	CheckBox	Sí	Se selecciona si el área esta activa	CA

TABLA 11: CASO DE PRUEBA FUNCIONAL INSERTAR CARGO (ELABORACIÓN PROPIA).

Descripción general:					
Permitirá crear un nuevo cargo en el sistema.					
Condiciones de ejecución:					
Para crear un nuevo cargo el usuario debe estar logueado en el sistema y poseer los permisos necesarios.					
Escenario	**Descripción**	**NC**	**CA**	**R/ del sistema**	**Flujo central**
ESC 1.1	El usuario con permisos correspondientes llena todos los campos del formulario correctamente.	Jefe de proyecto	Si	El sistema debe crear un cargo satisfactoriamente y muestra notificación de éxito.	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo". Aparecerá un listado con los cargos existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo cargo. Una vez insertados los datos del cargo se presiona el botón *Aceptar*. Si no desea crear

					el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez insertado el cargo debe mostrarse una lista con todos los cargos incluyendo el insertado.
ESC 1.2	El usuario deja uno o más campos vacíos.	vacío	Si	El sistema muestra un mensaje de error: *"Este campo es requerido".*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo". Aparecerá un listado con los cargos existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo cargo. Una vez insertados los datos del cargo se presiona el botón *Aceptar*. Si no desea crear el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez insertado el cargo debe mostrarse una lista

					con todos los cargos incluyendo el insertado.
ESC 1.3	El usuario introduce caracteres inválidos en los campos.	☺☻♥ ♠♣♀	Si	El sistema muestra un mensaje de error: *1- Si es de solo números: "El campo debe tener solo números"* *2- Si es de texto: "El campo debe tener solo letras"* 3- *Si es alfanumérico: "El campo debe tener solo letras y/o números"*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo". Aparecerá un listado con los cargos existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo cargo. Una vez insertados los datos del cargo se presiona el botón *Aceptar*. Si no desea crear el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez insertado el cargo debe mostrarse una lista con todos los cargos incluyendo el insertado.

TABLA 12: CASO DE PRUEBA FUNCIONAL EDITAR CARGO (ELABORACIÓN PROPIA).

Descripción general:
Permitirá editar un cargo en el sistema.
Condiciones de ejecución:
Para editar un cargo el usuario debe estar logueado en el sistema y poseer los permisos necesarios.

Escenario	Descripción	NC	CA	R/ del sistema	Flujo central
ESC 2.1	El usuario con permisos correspondientes llena todos los campos del formulario correctamente.	Jefe de proyecto	Si	El sistema debe editar un cargo satisfactoriamente y muestra notificación de éxito.	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo-Editar". Aparecerá un formulario con los datos correspondientes para editar un cargo. Una vez editados los datos del cargo se presiona el botón *Aceptar*. Si no desea editar el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez editado el cargo debe mostrarse una lista con todos los cargos incluyendo el editado.
ESC 2.2	El usuario deja uno o más campos vacíos.	vacío	Si	El sistema muestra un mensaje de error: *"Este campo es requerido".*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo-Editar". Aparecerá un formulario con los datos correspondientes para editar un cargo. Una vez editados los datos del cargo se presiona el botón *Aceptar*. Si no desea editar

					el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez editado el cargo debe mostrarse una lista con todos los cargos incluyendo el editado.
ESC 2.3	El usuario introduce caracteres inválidos en los campos.	☺☻♥ ♠♣♀	Si	El sistema muestra un mensaje de error: *1- Si es de solo números: "El campo debe tener solo números"* *2- Si es de texto: "El campo debe tener solo letras"* 3- *Si es alfanumérico: "El campo debe tener solo letras y/o números"*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo-Editar". Aparecerá un formulario con los datos correspondientes para editar un cargo. Una vez editados los datos del cargo se presiona el botón *Aceptar*. Si no desea editar el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez editado el cargo debe mostrarse una lista con todos los cargos incluyendo el editado.

TABLA 13: CASO DE PRUEBA FUNCIONAL GESTIONAR ROL (ELABORACIÓN PROPIA).

No.	Nombre del campo	Clasificación	Valor nulo	Descripción	Alias para el caso de prueba

1	Rol	Campo de texto	No	Se introduce el nombre del Rol	NR
4	Activo	CheckBox	Sí	Se selecciona si el área esta activa	RA

TABLA 14: CASO DE PRUEBA FUNCIONAL INSERTAR ROL (ELABORACIÓN PROPIA).

Descripción general:					
Permitirá crear un nuevo rol en el sistema.					
Condiciones de ejecución:					
Para crear un nuevo rol el usuario debe estar logueado en el sistema y poseer los permisos necesarios.					
Escenario	**Descripción**	**NR**	**RA**	**R/ del sistema**	**Flujo central**
ESC 1.1	El usuario con permisos correspondientes llena todos los campos del formulario correctamente.	Programador	Si	El sistema debe crear un rol satisfactoriamente y muestra notificación de éxito.	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Rol". Aparecerá un listado con los roles existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo rol. Una vez insertados los datos del rol se presiona el botón *Aceptar*. Si no desea crear el rol presione el botón *Cancelar* para volver al listado de los roles existentes. Una vez insertado el cargo debe mostrarse una lista

					con todos los cargos incluyendo el insertado.
ESC 1.2	El usuario deja uno o más campos vacíos.	vacío	Si	El sistema muestra un mensaje de error: *"Este campo es requerido".*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo". Aparecerá un listado con los cargos existentes y varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo cargo. Una vez insertados los datos del cargo se presiona el botón *Aceptar*. Si no desea crear el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez insertado el cargo debe mostrarse una lista con todos los cargos incluyendo el insertado.
ESC 1.3	El usuario introduce caracteres inválidos en los campos.	☺☻♥ ♠♣♀	Si	El sistema muestra un mensaje de error: 1- *Si es de solo números: "El*	En el menú a la izquierda se sigue la siguiente ruta: "Administración-Cargo". Aparecerá un listado con los cargos existentes y

				campo debe tener solo números” *2- Si es de texto: “El campo debe tener solo letras”* 3- *Si es alfanumérico: “El campo debe tener solo letras y/o números”*	varias opciones, presiona sobre el botón *Nuevo* y aparecerá un formulario con los datos correspondientes para crear un nuevo cargo. Una vez insertados los datos del cargo se presiona el botón *Aceptar*. Si no desea crear el cargo presione el botón *Cancelar* para volver al listado de los cargos existentes. Una vez insertado el cargo debe mostrarse una lista con todos los cargos incluyendo el insertado.

TABLA 15: CASO DE PRUEBA FUNCIONAL EDITAR ROL (ELABORACIÓN PROPIA).

Descripción general:					
Permitirá editar un rol del sistema.					
Condiciones de ejecución:					
Para editar un rol el usuario debe estar logueado en el sistema y poseer los permisos necesarios.					
Escenario	**Descripción**	**NR**	**RA**	**R/ del sistema**	**Flujo central**
ESC 2.1	El usuario con permisos correspondientes llena todos los campos del	Programador	Si	El sistema debe modificar el rol satisfactoriamente y muestra notificación de éxito.	En el menú a la izquierda se sigue la siguiente ruta: “Módulo de Administración-Rol-Editar y aparecerá un formulario con los datos correspondientes para

	formulario correctamente.				editar un rol. Una vez editados los datos del rol se presiona el botón *Aceptar*. Si no desea editar el rol presione el botón *Cancelar* para volver al listado de los roles existentes. Una vez editado el rol debe mostrarse una lista con todos los roles incluyendo el editado.
ESC 2.2	El usuario deja uno o más campos vacíos.	vacío	No	El sistema muestra un mensaje de error: *"Este campo es requerido".*	En el menú a la izquierda se sigue la siguiente ruta: "Módulo de Administración-Rol-Editar y aparecerá un formulario con los datos correspondientes para editar un rol. Una vez editados los datos del rol se presiona el botón *Aceptar*. Si no desea editar el rol presione el botón *Cancelar* para volver al listado de los roles existentes. Una vez editado el rol debe mostrarse una lista con

					todos los roles incluyendo el editado.
ESC 2.3	El usuario introduce caracteres inválidos en los campos.	☺☻♥ ♠♣♀	Si	El sistema muestra un mensaje de error: 1- *Si es de solo números: "El campo debe tener solo números"* 2- *Si es de texto: "El campo debe tener solo letras"* 3- *Si es alfanumérico: "El campo debe tener solo letras y/o números"*	En el menú a la izquierda se sigue la siguiente ruta: "Módulo de Administración-Rol-Editar y aparecerá un formulario con los datos correspondientes para editar un rol. Una vez editados los datos del rol se presiona el botón *Aceptar*. Si no desea editar el rol presione el botón *Cancelar* para volver al listado de los roles existentes. Una vez editado el rol debe mostrarse una lista con todos los roles incluyendo el editado.