UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 1

Sistema para la gestión de la prenómina en el Centro de Ideoinformática

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

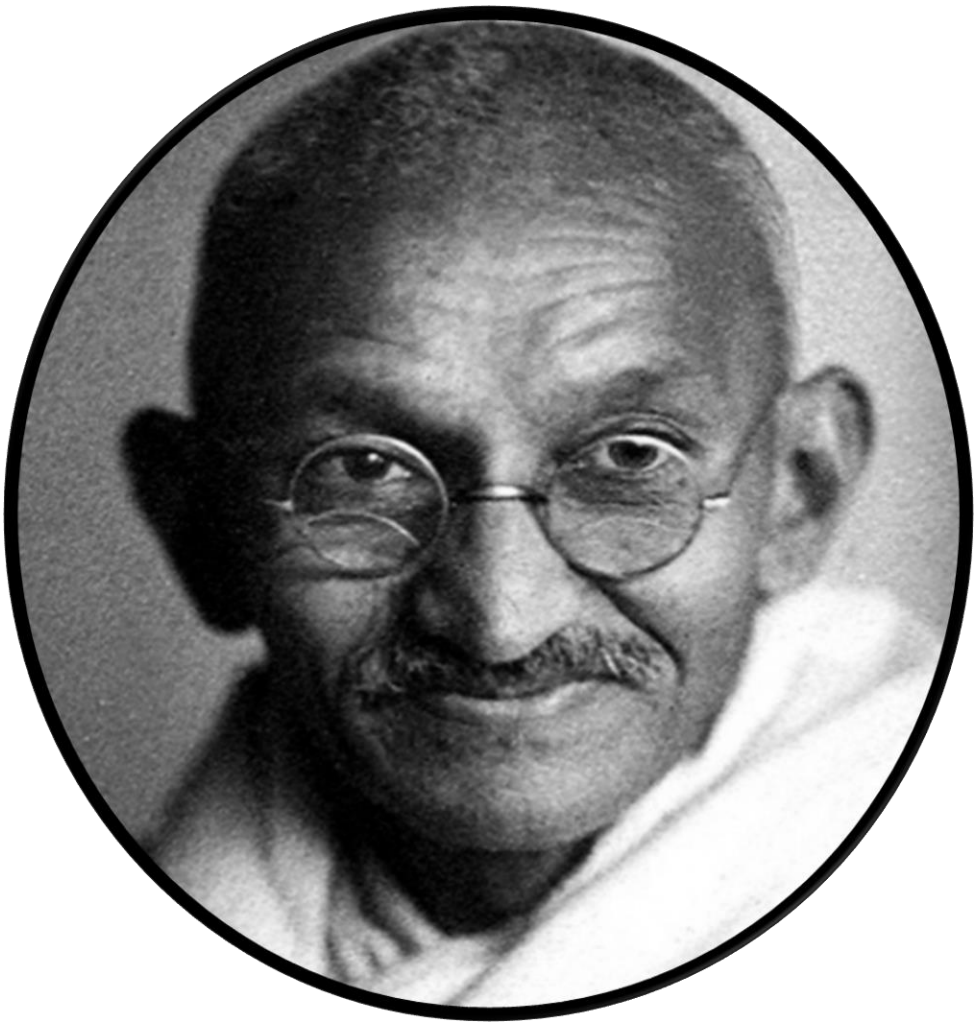
Autor:
Luis Ernesto Fernández Buchillón

Tutores:

MSc. Dunnia Castillo Galán

MSc. Sahilyn Delgado Pimentel

La Habana, junio de 2019



Vive como si fueras a morir mañana. Aprende como si fueras a vivir siempre.

Gandhi

Declaración de autoría

Declaro por este medio que yo Luis Ernesto Fernández Buchillón, con CI:95111839279 soy el autor principal del trabajo titulado Sistema para la gestión de la prenómina en el Centro de Ideoinformática y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Autor

_____

Luis Ernesto Fernández Buchillón

Tutor

_____

Msc. Sahilyn Delgado Pimentel

Tutor

_____

Msc. Dunnia Castillo Galán

Agradecimientos

A mis padres, mis abuelos, mis primos, mis tías y toda mi familia por que sin ellos no estaría aquí. A mis amigos y amigas porque sin ellos no fuera quien soy hoy. A la familia del BF3 por los buenos ratos y lo que hemos compartido juntos. A la mujer de mi vida, Dani, que me enseñó que si de verdad deseas algo con muchas ganas solo tienes que sentirlo y nacerá como una cascada fluyendo como soñaste.

Dedicatoria

*A **mis padres**, **mis abuelos** y **mi familia** que han luchado toda la vida para que este momento llegara. A **mi Dani** que siempre me ha apoyado, ha creído en ese **mí** que solo ella sabe que existe, me ha entendido como nadie y nunca me ha fallado.*

A todos, con todo mi corazón.

*Luis Ernesto Fernández Buchillón (**Mr. J**).*

Resumen

Como parte del proceso de informatización de la sociedad cubana, se crea la Universidad de las Ciencias Informáticas (UCI), la cual cuenta con una extensa población universitaria. Debido a las deficiencias que trae consigo que el control de la asistencia y la confección de la prenómina salarial de los trabajadores se lleve a cabo de manera manual; se desarrolló un estudio donde se determinó que era de vital importancia implementar una herramienta que sirviera de apoyo a los responsables de llevar a cabo dicha tarea. La presente investigación consiste en la implementación de un sistema capaz de controlar los elementos de la asistencia, incidencias laborales y realizar la confección de la prenómina salarial. El sistema permitirá un aumento de la disponibilidad de la información, la confidencialidad y la seguridad de toda la documentación generada en la ejecución de este proceso. Se realizó un estudio de las herramientas, metodologías y lenguajes existentes que permitieran la construcción de un sistema informático afín a las necesidades y condiciones tecnológicas existentes.

Palabras clave: informatización, prenómina, procesos, sistema de gestión.

Índice

Índice de figuras

Índice de tabla

INTRODUCCIÓN

El constante avance de las Tecnologías de la Información y las Comunicaciones (TIC), ha constituido una revolución mundial. En la actualidad, las TIC abarcan todos los ámbitos de la experiencia humana, propiciando un elevado desarrollo en muchas áreas como la economía, la política y la cultura. Su beneficio depende, en su totalidad, de cómo las use una determinada entidad y cuánta importancia se le otorgue a la gestión de la información sobre la misma (Gracía, 2016).

Actualmente se vive en la sociedad de la información y el conocimiento, tendencia evidenciada fundamentalmente, en la relación existente entre el avance de las TIC y la evolución de la humanidad. Donde el desarrollo económico, tecnológico y social de una nación, está cada día más ligado al desarrollo científico-tecnológico y social de la misma. Todos los procesos enmarcados de esta forma contribuyen de manera continua al paradigma de socialización digital a nivel global, en los cuales los elementos conectados entre sí no son solo páginas, sino personas.

Esto trae consigo, que el protagonismo que ha alcanzado la tecnología en la vida cotidiana de las personas esté completamente ligado a las organizaciones e instituciones en general. En toda empresa o entidad donde existe un gran número de personal, uno de los procesos que se suele informatizar es la gestión de los recursos humanos debido a que es mucha la información generada respecto a pagos, asistencias y certificados.

Los Recursos Humanos (RRHH) constituyen el eslabón más débil de las políticas de modernización, debido a que modernizar significa cambiar la cultura de la organización y, por tanto, incidir sobre valores y actitudes de las personas que trabajan en el sector empresarial. En Cuba existe un Sistema Integral de Capital Humano, el cual se define como conjunto de políticas, objetivos, metas, responsabilidades, normativas, funciones, procedimientos y herramientas permiten la integración interna de los procesos de gestión de capital humano y externa con la estrategia de la empresa, a través de competencias laborales de un desempeño laboral superior y el incremento de la productividad del trabajo (Decreto No 281, 2007).

Cuba no es un país ajeno a la gestión de los recursos humanos, por lo que este campo también presenta la necesidad de informatizarse, por eso en el país se han desarrollado aplicaciones informáticas para contribuir al desarrollo de la informatización en este sector en empresas tanto nacionales como del exterior. Una de

las instituciones que ha dado grandes pasos en este sentido es la Universidad de las Ciencias Informáticas (UCI).

Como parte de los centros de desarrollo pertenecientes a la universidad, se encuentra en la Facultad 1 el Centro de Ideoinformática (CIDI) el cual se dedica a la búsqueda de soluciones para el trabajo en internet. Este cuenta con tres departamentos y un aproximado de 72 trabajadores de ambos sexos. En este sentido resulta engorrosa la gestión manual de los recursos humanos a partir de un documento excel para el registro de la documentación ya que en el centro coinciden múltiples casos de incidencias en las claves de ausencias como licencia por maternidad, licencia sin sueldo, vaciones y certificados durante períodos prolongados, por lo que, con el volumen de información para almacenar, se pueden cometer errores como pagos indebidos cuando se realizan las prenóminas o la no correspondencia entre las fechas de entrega y registro de licencias sin sueldo y de maternidad. Por otro lado, resulta complicado para el asistente de control consultar los datos laborales por el cúmulo de información almacenada en formato duro. Todo esto puede provocar problemas o atraso en la prenómina[1] además de verse comprometido el control de la misma.

Considerando la **situación problemática** anteriormente descrita, se plantea como **problema de investigación:** ¿Cómo mejorar la gestión de la prenómina en el Centro de Ideoinformática?

Para la realización de la investigación se define como **objeto de estudio:** El proceso de gestión de prenóminas de los recursos humanos. Se define además como **campo de acción**: La gestión de la prenómina en el Centro de Ideoinformática.

En la presente investigación se plantea como **objetivo general**: Desarrollar un sistema informático utilizando tecnologías *web* para la mejora del proceso de gestión de la prenómina en el Centro de Ideoinformática.

Con el propósito de cumplir gradualmente el objetivo general antes mencionado, el mismo se desglosa en los siguientes **objetivos específicos:**

[1] Prenómina: nos permite emitir reportes y validaciones para verificar que la captura de datos y los cálculos sean correctos antes de continuar con la operación de la nómina; por lo tanto, la **prenómina** se puede revisar tantas veces como sea necesario.

1. Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con el desarrollo de herramientas para la gestión de la prenómina en el Centro de Ideoinformática.
2. Realizar el estudio de sistemas homólogos a nivel nacional y global.
3. Identificar las funcionalidades del sistema para la gestión de la prenómina en el Centro de Ideoinformática.
4. Diseñar el sistema para la gestión de la prenómina en el Centro de Ideoinformática.
5. Implementar las funcionalidades del sistema para la gestión de la prenómina en el Centro de Ideoinformática.
6. Validar las funcionalidades del sistema para la gestión de la prenómina en el Centro de Ideoinformática a través de pruebas de software.

Idea a Defender: Con la creación de un sistema informático utilizando tecnologías *web* se mejorará el proceso de la gestión de la prenómina en el Centro de Ideoinformática.

Para hacer efectivo el cumplimiento de los objetivos planteados, quedan definidas las siguientes **tareas de investigación:**

1. Realización de estudios sobre las formas de gestión de la prenómina en el Centro de Ideoinformática.
2. Realización de estudio de sistemas homólogos a nivel nacional y global.
3. Identificación de las funcionalidades que tendrá el sistema, las técnicas de programación, lenguaje de programación y marco de trabajo para el desarrollo de la aplicación.
4. Diseño del sistema para la gestión de la prenómina en el Centro de Ideoinformática.
5. Implementación del sistema para la gestión de la prenómina en el Centro de Ideoinformática.
6. Validación de las funcionalidades del sistema para la gestión de la prenómina en el Centro de Ideoinformática a través de pruebas de software.

En la etapa de investigación del presente trabajo de diploma se hizo uso de métodos teóricos y empíricos, así como de la técnica de recopilación de información.

Métodos teóricos:

En la presente investigación se emplearon los siguientes métodos teóricos:

- Histórico-lógico: permitió mediante el análisis de la evolución de la gestión de la prenómina, determinar las principales características que debía poseer el sistema que se describe en la presente investigación, así como las herramientas y tecnologías a utilizar.
- Analítico-sintético: permitió realizar el estudio teórico de la investigación facilitando el análisis de documentos y la extracción de los elementos más importantes relacionados con el proceso de desarrollo de sistemas de gestión de prenómina que hicieron posible la elaboración de conclusiones relacionadas con el objeto de estudio.
- Inductivo-deductivo: se aplicó para la determinación de las generalidades y se parte del análisis de casos particulares encontrados, para arribar a razonamientos que permitieron la fundamentación teórica y elaboración del sistema presentado.

Métodos empíricos:

Se emplearon en la presente investigación los siguientes métodos empíricos:

- Entrevista: se evidenció en conversaciones planificadas para obtener información con la directora y la asistente de control lo que permitió realizar el levantamiento de requisitos, así como un mejor entendimiento del funcionamiento de la tecnología a utilizar. Ver anexo 1.

- Observación: fue empleada en los distintos momentos de la investigación para la recogida de la información precisa, real y confiable que ayudó a comprender lo principal de la problemática. Posibilitó obtener conocimiento acerca del funcionamiento de los sistemas existentes en la actualidad para proponer una mejor solución al proceso en cuestión.

- La Modelación: se utilizó con el objetivo de realizar una representación simplificada de los cálculos necesarios, descomponerlos y estudiarlos para conocer nuevas relaciones y cualidades del proceso para mejorar su comprensión, operar y experimentar con ellos, mediante el uso de diagramas y modelos más simples.

Estructura del trabajo:

Capítulo I. Fundamentación teórica del sistema de gestión de prenómina en el centro CIDI. En este capítulo se acentúan las bases para entender el problema a solucionar. Se realiza un estudio de algunas de

las soluciones existentes actualmente en Cuba y en el mundo. Se definen los conceptos fundamentales, técnicas, tendencias, lenguajes, metodologías y herramientas, que se utilizaron en la construcción de la solución.

Capítulo II. Análisis y diseño del sistema para gestión de la prenómina en el centro CIDI. En este capítulo se representa y analiza todo lo relacionado con el modelado del negocio del sistema para la gestión de la prenómina en el centro CIDI. Se explica cómo funciona este, las descripciones completas de los requisitos funcionales y no funcionales con que cuenta el sistema implementado, así como la de los diagramas que representan los requisitos del sistema.

Capítulo III. Implementación y validación del sistema para la gestión de la prenómina en el centro CIDI. Este capítulo muestra todo lo relacionado con el modelo de implementación del sistema a partir del diseño anteriormente realizado. Se describen las pruebas realizadas al sistema desarrollado y los resultados obtenidos, así como la validación de la misma a partir del criterio de expertos en el tema.

CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA PARA EL SISTEMA DE GESTIÓN DE PRENÓMINA EN EL CENTRO CIDI

Introducción

En este capítulo se exponen los elementos relacionados con el marco teórico metodológico que respalda la presente investigación. Se definen los conceptos esenciales asociados al dominio de la investigación. Se realiza el análisis de las soluciones similares existentes en el ámbito nacional e internacional, lo cual permite evidenciar la necesidad de desarrollar el sistema propuesto para dar solución al problema planteado; así como seleccionar el entorno de desarrollo adecuado para su implementación. Se analizan además las características del desarrollo de la propuesta de solución con el fin de seleccionar la metodología de software adecuada para guiar este proceso.

1.1 Conceptos asociados al dominio de la investigación

Para tener una mejor comprensión sobre los temas que se estarán abordando en el presente trabajo de diploma, es necesario comprender las definiciones más importantes que forman parte del problema reflejado en la investigación.

Sistema informático. Puede ser definido como un sistema de información que basa la parte fundamental de su procesamiento, en el empleo de la computación, como cualquier sistema, es un conjunto de funciones interrelacionadas, *hardware*, software y de Recurso Humano. Un sistema informático normal emplea un sistema que usa dispositivos que se usan para programar y almacenar programas y datos (Alegsa, 2018).

Prenómina: Es el proceso que se realiza para conocer los días trabajados, faltas, retardos, días festivos trabajados, horas extras, bono de puntualidad y sueldo bruto a pagar de cada trabajador, así como el cálculo de primas vacacionales y dominicales (Angulo, 2017).

Gestión de la prenómina: consiste en elaborar oportunamente la liquidación de salarios, prestaciones sociales, aportes al Sistema General de Seguridad Social en Salud y para fiscales, permitiendo poder efectuar el pago a los funcionarios vinculados a la planta de personal y demás obligaciones de pagos a terceros contraídas por la entidad, con el fin de garantizar el cumplimiento de las obligaciones laborales en la entidad (Ordoñez, 2017).

Gestión de recursos humanos: la gestión del talento humano ha sido estudiada y definida por diversos autores en las distintas etapas de su desarrollo. García, Sánchez y Zapata (2008:13) asumen la gestión

humana como: “la actividad estratégica de apoyo y soporte a la dirección, compuesta por un conjunto de políticas, planes, programas y actividades, con el objeto de obtener, formar, motivar, retribuir y desarrollar al personal requerido para generar y potencializar, la administración, la cultura organizacional y el capital social, donde se equilibran los diferentes intereses que convergen en la organización para lograr los objetivos de manera efectiva”. La importancia del área se centra en posibilitar la consecución de los intereses personales de los empleados y de los intereses de las organizaciones, por supuesto (García, 2008).

El autor de la presente investigación define el **Sistema de gestión de la prenómina** como una herramienta que permite emitir reportes y validaciones para optimizar recursos, reducir costes y mejorar la productividad en cuanto a la captura de datos y a cálculos correctos. Es un instrumento que reporta datos en tiempo real que permiten tomar decisiones para corregir fallos y prevenir la aparición de gastos innecesarios.

1.1.1 Descripción del proceso de gestión de prenómina en el centro de Ideoinformática

En el Centro de Ideoinformática la gestión de la prenómina se realiza a través de la utilización del catálogo de claves de ausencia ordenadas por código (Anexo 2 y 3). Esta prenómina se conforma en todos los departamentos del centro por separado, ajustando las personas por su horario de trabajo mediante el modelo oficial (Anexo 4) de la prenómina del mismo. Dentro de los casos de horarios atípicos se encuentran los choferes que se les paga por horas y los técnicos generales los cuales se les aplica la cuota adicional de la nocturnidad. Todas las personas tienen una evaluación asignada por el jefe del departamento, la misma está condicionada por el requisito de cobro que a su vez depende de las claves aplicadas a cada persona en el mes.

1.2 Estado del arte asociado a los sistemas homólogos

En la actualidad el desarrollo tecnológico ha posibilitado llevar la automatización a todas las esferas de la sociedad. En contradicción se puede decir que una de las áreas que más ha tardado en automatizarse a nivel mundial ha sido la del Control de Asistencia. Esto se debe a la amplia utilización de sistemas tradicionales de control basados en los antiguos relojes mecánicos que utilizan tarjetas de cartón, o registros de la hora en la cual ingresan o salen de la empresa de forma manual a través de un libro de firma.

No obstante, existen mecanismos que posibilitan que dicho control se lleve a cabo de forma más precisa y controlada. A partir de códigos de barras, tarjetas de proximidad por radio frecuencia (RFID), Tarjetas de

Banda Magnética o Sistemas Biométricos de Huellas Digitales y con la ayuda de un software capaz de controlar la entrada de datos (Silveiro, 2015).

1.2.1 Análisis de soluciones existentes a nivel internacional

GrupoTress: concentra toda la información relacionada con los empleados de la empresa para agilizar la administración, consulta, modificación y organización de la misma. GrupoTress entre otras funcionalidades brinda al cliente la posibilidad de, mediante una tarjeta magnética de uso personal, llevar el control de la asistencia, retardos, horas extras de cada trabajador. Además, es capaz de generar de forma automática la prenómina. Brinda informes de horas trabajadas durante el período de nómina. Manejo de horas de comida y descanso, entre otros. Además proporciona una gran cantidad de reportes con el objetivo de optimizar el control empresarial (Cervantes, 2012).

Infinite Consulting.SA: es una empresa especializada en la instalación de dispositivos biométricos. Cuenta con el software Mantra para la gestión de asistencia que le permite crear los reportes característicos de un control de asistencia. Sus principales beneficios son: permite crear reportes de asistencia dentro de los parámetros de tiempo que el cliente elija. Evitará posibles fraudes de usurpación de identidad muy comunes por el uso de tarjetas. Permite saber en tiempo real quien está o no, así como hacer seguimientos globales o personales para saber el desempeño asistencial de su personal (Mok, 2012).

'Doce' versión 2: es un programa de control de asistencia que incorpora un registro de datos del personal de una empresa o institución. El control de asistencia se efectúa usando el teclado de una computadora en el cual el personal registra un código individual de la misma manera en que se podría hacer con un reloj tarjetero. Permite utilizar una cámara *web* para que tome fotografías de la persona que marca su entrada y salida de manera que, posteriormente, a través de una revisión manual, sea posible identificar algún fraude cometido por personas que marcan a cuenta de otras. Los reportes que el programa puede generar son: distintos modelos de nóminas, historial, reporte de atrasos y faltas, control de marcas en el reloj, asistencia del personal por día, planilla mensual, tiempo no trabajado por atrasos o salidas anticipadas, selección manual, cuadro mensual exportado a MS Excel (Paez, 2016).

Los programas de software **GrupoTress** y **Mantra** brindan muchas funcionalidades con relación al control de la asistencia que pudieran ser de utilidad, pero requieren leer los datos provenientes de terminales colectoras de datos, usando tarjetas de banda magnética el primero y a través de dispositivos biométricos en el caso de Mantra. Deja de ser viable la utilización de ambos productos debido al elevado costo que

representa en la actualidad instalar este tipo de dispositivos por lo que la Universidad de las Ciencias Informáticas no tendría la posibilidad de adquirir esta tecnología, ello unido al hecho de que todos presentan licencia privativa hace imposible su utilización.

Por otra parte, **Doce versión 2** posee características similares a las de los sistemas anteriores a pesar de tener una menor cantidad de prestaciones. Tiene la peculiaridad de que, a diferencia de los anteriormente analizados, este no necesita obligatoriamente contar con lectores de datos tan exigentes para la captura de los mismos. Cuenta con un mecanismo para llevar a cabo el control de asistencia mucho más simple, se puede realizar usando el teclado de una computadora en la cual el personal registra un código individual de la misma manera en que se podría hacer con un reloj tarjetero. Además, este producto posee ciertas funcionalidades que se adaptan a las necesidades de la presente investigación; pero se imposibilita la adecuación a los requerimientos de la situación problemática porque al igual que los dos anteriores todos poseen licencia privativa.

1.2.2 Análisis de los sistemas homólogos para el control de asistencia y gestión de nóminas en Cuba

Fastos (Sistema Integral para la gestión de Recursos Humanos): el sistema de Recursos Humanos, está formado por los módulos Configuración, Personal, Capacitación, Cuadros y Evaluación de Desempeño. Permite controlar las informaciones fundamentales de los empleados de una entidad, también realizar varios procesos y operaciones que son inherentes al área de recursos humanos, tales como:

- Registro de los empleados: se guardan los datos de los empleados, así como informaciones referentes a los reportes de vacaciones, certificados médicos, licencias, resolución.
- Control de la plantilla: permite establecer la estructura organizativa de las plazas de la entidad.
- Control de asistencia: lo cual incorpora el control de claves de asistencias, turnos de trabajos, horarios, tarjeta de asistencia e incidencias de cada empleado.
- Permite acoplar relojes (RTA 600) para actualizar la información de la tarjeta de asistencia de forma automática.
- Informes y modelos: permite obtener un total de 56 informes, por ejemplo, cierre del período, análisis de fondo de tiempo, estadísticos, entre otros (Desoft, 2011).

Rodas XXI: Sistema Integral Económico Administrativo desarrollado por la empresa cubana CITMATEL. es capaz de guardar por trabajador los pagos y retenciones fijas que se realizan a cada uno de ellos por lo que para la confección de las nóminas cada mes sólo es necesario actualizar las incidencias que correspondan y todo el trabajo posterior de cálculo es realizado de forma automática. Este módulo permite el control, planificación y gestión de la actividad de recursos humanos aplicable en todas las entidades, incluye administración de personal y cuadros (Rossum, 2019).

Ambos sistemas poseen funcionalidades que podrían servir ya que cuentan con características similares a las que se desea obtener. Sin embargo, Fastos requiere la utilización de relojes RTA 600 lo que ocasionaría un gasto innecesario para la facultad por lo que no es factible su utilización. Por su parte el Sistema Integral Económico Administrativo Rodas, como su nombre lo indica, posee varios módulos capaces de solucionar la mayoría de los problemas de una empresa. Por su complejidad y la cantidad de módulos que opera se cree que su utilización se ajusta mejor a otros departamentos de la empresa. Se concluye que ambos sistemas presentan especificaciones que no se adecuan con las características particulares de la dirección ni con la existencia de horarios de trabajos atípicos con que se cuenta actualmente, además de que ambos poseen licencia privativa.

Tabla 1. Estudio de los sistemas homólogos. Fuente: Elaboración propia.

Sistemas	**Licencia**	**Código abierto**	**Dominio de aplicación (web)**	**Realiza Prenómina**
GrupoTress	Privativa	No	No	Sí
Infinite Consulting.SA	Privativa	No	No	No
Doce versión 2	Privativa	No	No	Sí
Fastos	Privativa	No	No	No
Rodas XXI	Privativa	No	No	Sí

Después de profundizar en algunos de los sistemas similares en el mundo y en Cuba se percibe que todos son privativos por lo que no es posible obtener sus códigos, 3 si permiten generar la prenómina y ninguno es de dominio *web*, llegando a la conclusión de que cada uno brinda una visión distinta del mismo objetivo a lograr. No sería correcto mezclarlos ya que cada uno cumple su objetivo específico. Para el sistema de gestión de la prenómina de CIDI se estudiaron los casos homólogos presentados, pero se enfocará en su objetivo general. Esto significa que se extraerá el conocimiento necesario de cada uno para lograr una conformación diferente a las otras versiones y poseerá características nuevas y específicas para el centro atendiendo a la existencia de horarios de trabajos atípicos con que se cuenta actualmente y la forma en la que se trabaja en el centro identificada por el horario establecido en la UCI.

1.3 Metodología seleccionada

El éxito del producto a desarrollar depende, en gran parte, de la metodología escogida, ya sea tradicional o ágil, donde los equipos maximicen su potencial y aumenten la calidad del producto con los recursos y tiempos establecidos. La metodología de investigación es una disciplina de conocimiento encargada de elaborar, definir y sistematizar el conjunto de técnicas, métodos y procedimientos que se deben seguir durante el desarrollo de un proceso de investigación para la producción de conocimiento (Figueroa, 2012).

Una metodología está compuesta por:

- Qué tareas se llevan a cabo en cada etapa: actividades elementales en que dividen los procesos.
- Procedimiento: definición de la forma de ejecutar la tarea.
- Técnica: herramienta utilizada para aplicar un procedimiento.
- Herramienta: para realizar una técnica, se puede apoyar en las herramientas software que
- automatizan su aplicación.
- Producto: resultado de cada etapa.

Se definió como metodología a utilizar en la presente investigación **PROCESO UNIFICADO AGIL (AUP-UCI):** teniendo en cuenta que es la metodología adaptada al ciclo de vida de los proyectos productivos de la universidad, es ampliamente usada en el área y es extremadamente flexible al proceso de desarrollo de software. AUP-UCI constituye una variante de AUP (Proceso Unificado Ágil, por sus siglas en inglés). Surge con el objetivo de ser una metodología que se adapte al ciclo de vida definido por la actividad productiva en

la universidad. Se elaboró teniendo en cuenta el Modelo CMMI-DEV v1.3 que constituye una guía para aplicar las mejores prácticas en una entidad desarrolladora, estas prácticas se centran en el desarrollo de productos y servicios de calidad (Rodríguez Sánchez, 2014).

Esta metodología propone tres fases para el desarrollo del software: Inicio, Ejecución y Cierre, así como siete disciplinas: Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas internas, Pruebas de liberación y Pruebas de aceptación. Cuenta además con Once roles: Jefe de proyecto, Planificador, Analista, Arquitecto de información, Desarrollador, Administrador de la configuración, Cliente o proveedor de requisitos, Administrador de calidad, Probador, Arquitecto de software y Administrador de bases de datos (Rodríguez Sánchez, 2014).

AUP-UCI propone además cuatro escenarios a utilizar para modelar el sistema en los proyectos. Se selecciona el escenario 2 que aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

Enmarcando en la investigación el escenario 2 se puede decir que la gestión de la prenómina en el Centro de IdeoInformática es un proceso definido y este se adecua perfectamente con el tipo de metodología seleccionada.

1.4 Herramientas

Las herramientas de desarrollo de software son productos informáticos que dan soporte a una tarea concreta dentro de las actividades de desarrollo de software para facilitar y asegurar la entrega de un sistema con calidad.

1.4.1 Herramientas *CASE*

Las herramientas *CASE* (Ingeniería de Software Asistida por Ordenador por su nombre en inglés *Computer Aided* Software *Engineering*) se definen como un conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software (M.T, 2008).

***Visual Paradigm for UML* 8.0**

Visual Paradigm para *UML* es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado *UML* ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta *UML CASE* también proporciona abundantes tutoriales de *UML*, demostraciones interactivas de *UML* y proyectos *UML* (Larman, 2007).

Para dar solución a la presente investigación se elige Visual Paradigm por sus características que permiten dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación, además de que este software de modelado *UML* ayuda a una más rápida construcción de aplicaciones de calidad y a un menor coste, además de ser un software libre.

1.4.2 Servidor Web NGINX

Un servidor web es un programa que sirve para atender y responder a las diferentes peticiones de los navegadores, proporcionando los recursos que soliciten usando el protocolo HTTP o el protocolo HTTPS (la versión cifrada y autenticada). Un servidor web es un programa que se ejecuta continuamente en un computador, manteniéndose a la espera de peticiones de ejecución que le hará un cliente o un usuario de internet. El servidor web se encarga de contestar a estas peticiones de forma adecuada, entregando como resultado una página web o información de todo tipo, de acuerdo a los comandos solicitados (Sánchez, 2010).

En el desarrollo del sistema propuesto se utilizará como servidor web **NGINX**, este es un servidor http y proxy inverso gratuito, de código abierto y de alto rendimiento, además de ser servidor proxy para IMAP y POP3. Es conocido por su estabilidad, su gran conjunto de características, una configuración sencilla y por consumir pocos recursos (Abalos, 2012).

Algunas de las principales características de NGINX son:

- Capacidad de manejar más de 10 000 conexiones simultáneas con bajo uso de memoria.
- Balanceo de carga.
- Tolerancia a fallos.
- Soporte TLS/SSL.

- Autenticación de acceso.
- Compresión y descompresión gzip.
- Reescritura de URL.
- Limitaciones de conexiones concurrentes y respuestas.
- Manejo de ancho de banda.
- Proxy IMAP, POP3, SMTP.
- Procesamiento de datos XSLT (Muñoz, 2015).

Se define utilizar NGINX por su rapidez de respuesta a las peticiones y tráfico de datos desde cualquier lugar cuando se trata de las páginas estáticas como es el caso de la investigación presente.

1.4.3 Lenguajes de programación

JavaScript

JavaScript es un lenguaje de programación que permite a los desarrolladores crear acciones en sus páginas web. No requiere de compilación ya que el lenguaje funciona del lado del cliente. Tiene como principal característica ser un lenguaje independiente de la plataforma. Debido a sus peculiaridades también es muy utilizado para internet. Es un lenguaje basado en acciones que posee menos restricciones. Gran parte de la programación en este lenguaje está centrada en describir objetos, escribir funciones que respondan a movimientos del mouse, aperturas, utilización de teclas, cargas de páginas entre otros (Bernal, 2009).

Características:

- El código JavaScript incluido en páginas web es interpretado por cualquier navegador moderno.
- Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Modelo de Objetos del Documento (DOM, por sus siglas en inglés).
- Es un lenguaje estandarizado.
- Se utiliza en los controladores los cuales manejan los datos del modelo y los envía a la plantilla y esta renderiza la respuesta en el navegador.

Python

Un lenguaje de programación está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Peralta, y otros, 2014).

Para el desarrollo del sistema se emplea el lenguaje ***Python*** en su versión 3.6, el cual es un lenguaje de programación eficaz 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 (Python, 2017).

Se decide utilizar Python por ser un lenguaje interpretado (no compilado), es multiplataforma lo cual es ventajoso para hacer ejecutable su código fuente entre varios sistemas operativos. Es un lenguaje de programación multiparadigma, el cual soporta varios paradigmas de programación como orientación a objetos, estructurada, programación imperativa y, en menor medida, programación funcional adecuándose de esta manera con las necesidades del sistema (Python, 2017).

1.4.4 Frameworks de desarrollo

Django

Para el desarrollo del sistema se utiliza el marco de trabajo *Django* en su versión 2.0.5, que es un marco de trabajo de desarrollo web totalmente implementado sobre ***Python*****,** con el que se pueden crear y mantener aplicaciones de alta calidad e incluye un servidor web ligero que se puede usar mientras se desarrolla (Django, 2014).

Características:

- Un mapeador objeto-relacional.
- Aplicaciones "enchufables" que pueden instalarse en cualquier página gestionada con Django.
- Una API de base de datos robusta.
- Un sistema incorporado de "vistas genéricas" que ahorra tener que escribir la lógica de ciertas tareas comunes.
- Un sistema extensible de plantillas basado en etiquetas, con herencia de plantillas.
- Un despachador de URLs basado en expresiones regulares.

- Un sistema de pasarelas para desarrollar características adicionales; por ejemplo, la distribución principal de Django incluye componentes middleware que proporcionan cacheo, compresión de la salida, normalización de URLs, protección CSRF y soporte de sesiones.
- Soporte de internacionalización, incluyendo traducciones incorporadas de la interfaz de administración (Django, 2014).

Se eligió utilizar Django como *framework* porque se integra perfectamente con el lenguaje Python y hereda todas sus características. Lo que permitirá separar la lógica de negocio, la lógica de servidor y la presentación de la aplicación web logrando una mejor calidad del sistema. Django es el único *framework* que por defecto viene con un sistema de administración activo, utiliza mapeo objeto-relacional los que permite realizar las consultas a la base datos sin SQL y es una gran oportunidad al ser un software libre.

AngularJS

Es un proyecto de código abierto, realizado en Javascript que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. Es un *framework* para el desarrollo ágil del *frontend* desarrollado sobre JavaScript (Camacho, 2017).

CSS3

CSS3 es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. Es la mejor forma de separar los contenidos y su presentación, siendo imprescindible en la creación de páginas web complejas. Este lenguaje es utilizado para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la tabulación con la que se muestran los elementos de una lista y la separación entre titulares y párrafos (Cederholm, y otros, 2010).

HTML5

HTML5 (*HyperText Markup Language*, versión 5) es la quinta revisión del lenguaje HTML. Esta nueva versión (aún en desarrollo), y en conjunto con CSS3, define los nuevos estándares de desarrollo web, rediseñando el código para resolver problemas y actualizándolo así a nuevas necesidades. No se limita solo a crear nuevas etiquetas o atributos, sino que incorpora muchas características nuevas y proporciona una plataforma de desarrollo de complejas aplicaciones web (Lubbers, 2010).

1.4.5 Sistema Gestor de Base de Datos

Un sistema gestor de base de datos se define como el conjunto de programas que administran y gestionan la información contenida en una base de datos. Ayuda a realizar las siguientes acciones: Definición de los datos, Mantenimiento de la integridad de los datos dentro de la base de datos, Control de la seguridad y privacidad de los datos, Manipulación de los datos (M A, 2010).

PostgreSQL 9.4

Es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia Distribución De Software De Berkeley o Berkeley Software *Distribution* (BSD) y con su código fuente disponible libremente. Es un sistema de gestión de bases de datos de código abierto potente en el mercado y en sus últimas versiones tiene todas las funcionalidades que otras bases de datos comerciales puedan tener. PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando. Sus características técnicas lo hacen uno de los gestores de bases de datos más potentes y robustos del mercado. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (Lynch, 2009).

1.4.6 Administrador de Base Datos

pgAdmin3 v1.22.2 es la herramienta oficial para administrar las bases de datos en PostgreSQL. Nos permite desde hacer búsquedas SQL hasta desarrollar toda nuestra base de datos de forma muy fácil e intuitiva; directamente desde la interfaz gráfica. Con pgAdmin crear una nueva base de datos y definir sus propiedades es muy sencillo, por lo que esto permite que tanto principiantes como expertos se sientan cómodos con el sistema, también se pueden crear respaldos, restaurar la base de datos o ejecutar tareas de mantenimiento de forma muy sencilla, los usuarios podrán seguir accediendo a los datos durante el proceso. PostgreSQL te permite desarrollar bases de datos relacionales robustas y eficientes (Pérez, 2017).

1.4.7 Entorno de Desarrollo Integrados (IDE)

JetBrains PyCharm es un entorno de desarrollo integrado de código abierto y multiplataforma desarrollado por *JetBrains*, que se utiliza para programar en *Python*. Proporciona un análisis de código, depuración gráfica, probador de unidad integrada y apoya el desarrollo web con el marco de trabajo *Django*. El mismo incluye heurísticas sofisticadas para determinar lo que cada tipo de variable es, y proporcionar sugerencias

de autocompletado para dichas variables. También se puede aprovechar la información en tiempo de ejecución cuando se ejecuta la aplicación con el depurador integrado (Worthington, 2018).Para la implementación del módulo se utiliza el *IDE PyCharm* en su versión 2018.3.

Se eligió como entorno de desarrollo integrado el *Pycharm* el cual permite trabajar con un intérprete de los lenguajes a utilizar junto al *framework* de desarrollo definido de una manera conjunta a la hora de utilizarlo en cualquier parte del código.

1.4.8 Librerías

JQuery es una biblioteca multiplataforma de *JavaScript*, que permite simplificar la manera de interactuar con los documentos *HTML*[2], manipular el árbol *DOM*[3], manejar eventos, desarrollar animaciones y agregar interacción con la técnica *AJAX* [4] a páginas web. Es además un software libre y de código abierto, permitiendo su uso en proyectos tanto libres como privados. *JQuery*, al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en *JavaScript* que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio (Gutíerrez, 2009). Para la implementación del sistema se emplea la librería *JQuery* en su versión 3.1.1.

[2] Del inglés *HyperText Markup Language* (Lenguaje de marcas de hipertexto).

[3] Del inglés *Document Object Model* (Modelo de Objetos del Documento o Modelo en Objetos para la Representación de Documentos)

[4] Del inglés, acrónimo de JavaScript asíncrono y XML.

Conclusiones

En este capítulo se han abordado los elementos teóricos que dan sustento a la propuesta de solución del problema planteado, en tal sentido se puede arribar a las siguientes conclusiones:

1. La definición de los principales conceptos asociados al dominio de la presente investigación y las relaciones entre estos, permitió sentar las bases teóricas para el desarrollo de un sistema de gestión de la prenómina.

2. El análisis de los sistemas homólogos, permitió identificar funcionalidades a implementar en el Sistema para la gestión de la prenómina en el Centro de Ideoinformática

3. El análisis de la documentación del sistema de gestión de la prenómina, sobre la metodología AUP-UCI en el escenario 2, permitió definir como herramienta de modelado el *Visual Paradigm*, como tecnología Django, los lenguajes de programación Python y Java Script utilizados para su implementación, permitiendo especificar el ambiente de desarrollo para la propuesta de solución.

CAPÍTULO 2. DESCRIPCIÓN DE LA PROPUESTA PARA EL SISTEMA DE GESTIÓN DE PRENÓMINA

Introducción

En este capítulo se presenta la propuesta de solución al problema planteado en el marco teórico de la investigación. Se define el diseño del sistema teniendo en cuenta el modelo de procesos, así como la descripción de los requisitos funcionales y no funcionales del sistema. Además, se especifica la arquitectura, patrones de diseño y modelo de datos empleado. Finalmente se muestran los artefactos asociados al modelo de base de datos y el diagrama de despliegue.

2.1 Descripción textual

En la UCI, todos los meses se debe entregar según fecha solicitada la prenómina de pago de los trabajadores. El sistema informático que se presenta permitirá establecer un control constante de la asistencia de los trabajadores de CIDI a partir del reflejo de las diferentes incidencias por claves de los trabajadores, independientemente del horario de trabajo que tengan establecido, así como el reflejo de la evaluación de desempeño mensual y trimestral de los trabajadores, nocturnidad, etc. Además, el sistema será capaz de ir completando la prenómina según se vaya registrando la asistencia del trabajador. La propuesta será una valiosa herramienta de control que permitirá facilitar la confección de la prenómina salarial en el Centro de Ideoinformática para su posterior exportación, impresión y entrega en el departamento de recursos humanos. Se debe partir del asistente de control y las funcionalidades principales del sistema con las que trabajara el mismo tales como gestionar área, gestionar cargo, gestionar persona, gestionar evaluación, generar reportes, generar prenómina y exportarla. No obstante, se define como flujo principal del sistema generar la prenómina el cual concatena de forma consecuente los demás subprocesos mencionados anteriormente. El administrador tendrá además la funcionalidad de gestionar los usuarios por roles dentro del sistema.

2.2 Modelo del dominio

Un modelo de dominio, es una representación de las clases conceptuales del mundo real, no de componentes de software. Podría considerarse como un diccionario visual de las abstracciones relevantes, vocabulario e información del dominio (Pérez, 2017).

Para lograr un mejor entendimiento de los procesos que requieren informatización, se realizó un modelo conceptual, (ver Figura 2), con un total de diez (10) clases y catorce (14) relaciones, el cual recoge y describe los conceptos más importantes dentro del contexto del sistema, así como las relaciones entre ellos.

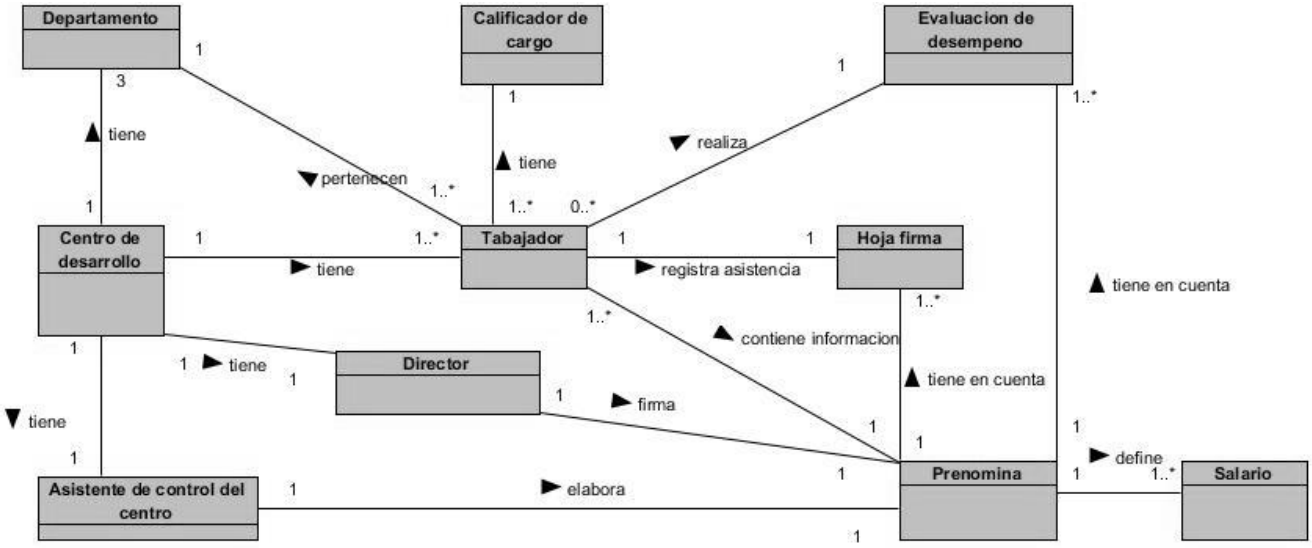


Figura 1. Diagrama del dominio. Fuente: Elaboración propia.

2.3 Requisitos de la propuesta de solución

Los requerimientos para un software son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas (Sommerville, 2011). Estos se dividen en:

2.3.1 Requisitos Funcionales

Los requisitos funcionales definen las funciones que el sistema será capaz de realizar.
A continuación, se listan los requisitos funcionales identificados para el sistema:

RF1-Autenticar usuario.

RF2-Insertar área.

RF3-Modificar área.

RF4- Mostrar área.

RF5-Eliminar área.

RF6-Listar área.

RF7-Buscar área.

RF8-Insertar persona.

RF9-Modificar persona.

RF10- Mostrar persona.

RF11-Eliminar persona.

RF12-Listar personas.

RF13-Buscar persona.

RF14-Insertar usuario.

RF15-Modificar usuario.

RF16- Mostrar usuario.

RF17-Eliminar usuario.

RF18-Buscar usuario

RF19-Insertar clave de ausencia.

RF20-Modificar clave de ausencia.

RF21- Mostrar clave de ausencia.

RF22-Eliminar clave ausencia.

RF23-Listar clave ausencia.

RF24-Buscar clave ausencia.

RF25-Insertar calificador de cargo.

RF26-Modificar calificador de cargo.

RF27- Mostrar calificador de cargo.

RF28-Eliminar calificador de cargo.

RF29-Listar cargos.

RF30-Buscar cargo.

RF31-Insertar evaluación de desempeño.

RF32-Modificar evaluación de desempeño.

RF33- Mostrar evaluación de desempeño.

RF34-Eliminar evaluación de desempeño.

RF35-Listar evaluaciones de desempeños.

RF36-Buscar evaluación de desempeño.

RF37-Asignar permisos a usuarios.

RF38-Asignar clave de ausencia a persona.

RF39-Asignar requisito de evaluación por clave de ausencia.

RF40-Generar reportes.

RF41-Generar prenómina.

RF42-Insertar prenómina

RF43-Modificar prenómina.

RF44-Mostrar prenómina.

RF45-Eliminar prenómina.

RF46-Listar prenóminas.

RF47-Guardar prenómina por mes/año.

RF48-Buscar prenómina por área.

RF49-Buscar prenómina por persona.

RF50-Buscar prenómina por cargo.

RF51-Buscar prenómina por evaluación.

RF52-Exportar a PDF y Excel.

RF53-Imprimir prenómina.

2.3.2 Requisitos No Funcionales

Los requisitos no funcionales son aquellos que no se refieren directamente a las funciones específicas que proporciona el sistema, sino a las propiedades emergentes de este como la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento. Define las restricciones del sistema como la capacidad de los dispositivos de entrada salida y las representaciones de datos que se utilizan en las interfaces del sistema (Pressman, 2010).

Usabilidad

RnF1: para interactuar con el sistema se requiere una preparación previa, permitiendo la fácil comprensión del mismo.

RnF3: el sistema deberá ser utilizado por los usuarios con dominio uci.cu

Confiabilidad

RnF4: el sistema debe ser tolerante a fallos.

Portabilidad

RnF5: la propuesta de desarrollo debe ser capaz de ejecutarse en los navegadores Firefox y Chrome, así como adaptar su interfaz a cualquier tipo de monitor de una computadora.

Eficiencia

RnF6: el sistema debe permitir que los usuarios interactúen con él de forma concurrente.

RnF7: el tiempo de respuesta de una petición al servidor debe ser de menos de 5 segundos o aproximadamente.

Soporte

RnF10: el sistema estará bien documentado de forma tal que el tiempo de mantenimiento sea mínimo en caso de necesitarse.

Hardware

RnF11: teniendo en cuenta que en la computadora va a ejecutarse un servidor web, se requiere como mínimo 80GB de disco duro, tarjeta de red de 100 MB, un procesador Pentium 4 y 4 GB de RAM para que la aplicación funcione correctamente.

Rendimiento

RnF12: la aplicación debe contar con una velocidad óptima de carga y actualización de los datos de la página, no sobrepasando los 3 segundos.

Seguridad

RnF13: para acceder al sistema el supervisor debe introducir su usuario y contraseña.

RnF14: ante los errores que pueda ocasionarse en el sistema no se debe mostrar destalles de información que puedan comprometer su seguridad e integridad.

RnF15: se mostrará solo la información necesaria para orientar al usuario.

RnF16: la información manejada por el sistema estará protegida de acceso no autorizado.

2.4 Diagrama de caso de uso del sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Los casos de uso permiten mostrar el entorno (actores) y el alcance (requisitos funcionales expresados como casos de uso) de un sistema. Un caso de uso describe la secuencia de interacciones que se producen entre el sistema y los actores del mismo para realizar una determinada función. Los actores son elementos externos (personas, otros sistemas, etc.) que interactúan con el sistema como si de una caja negra se tratase. Un actor puede participar en varios casos de uso y un caso de uso puede interactuar con varios actores (Pressman, 2010).

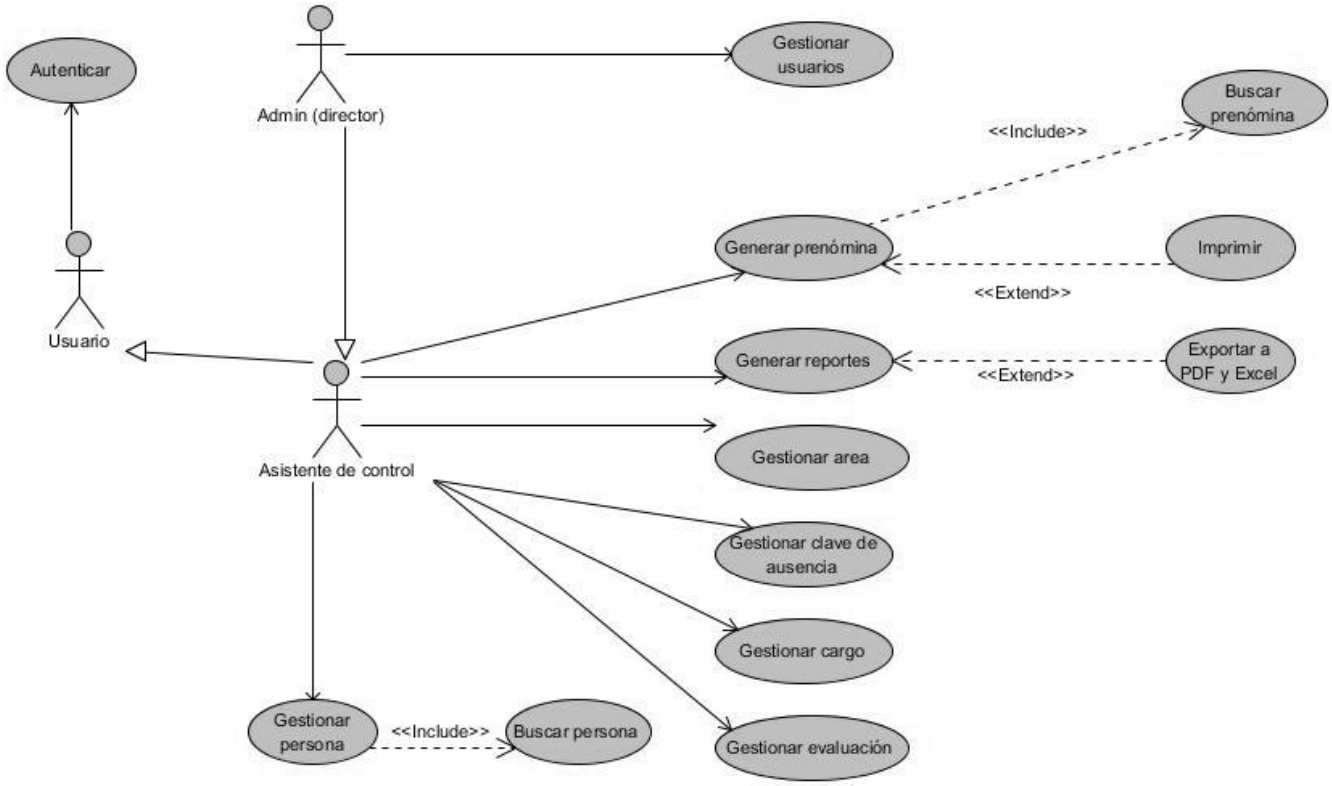
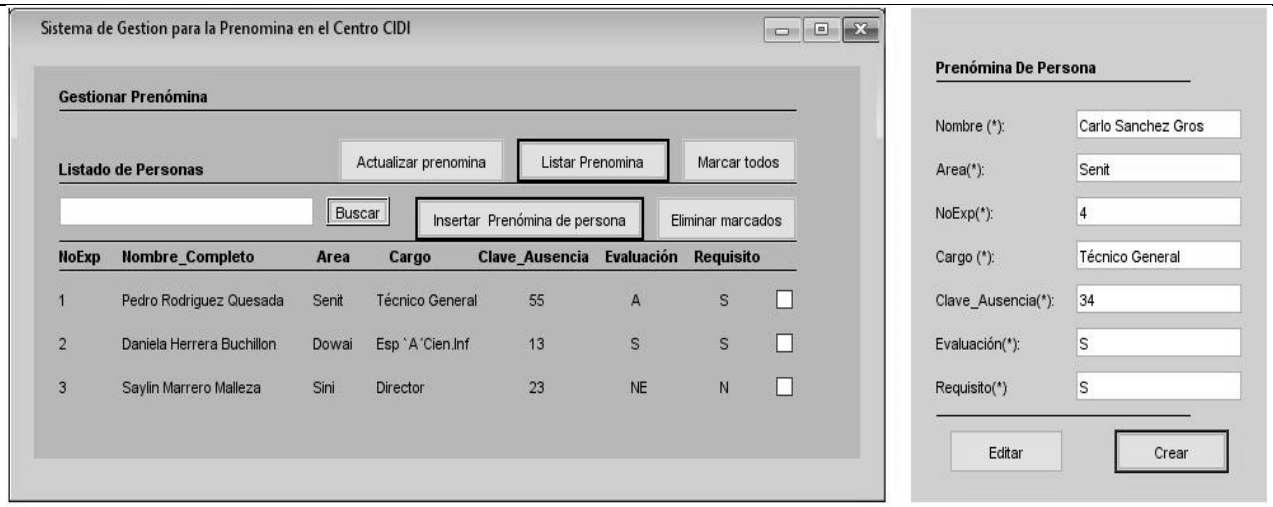


Figura 2. Diagrama de caso de uso. Fuente: Elaboración propia.

Tabla 2. Caso de Uso Gestionar Prenómina. Fuente: Elaboración propia.

Objetivo		Permitir insertar, modificar y eliminar datos acerca de una prenómina.
Actores		Administrador.
Resumen		El caso de uso se inicia cuando el administrador del sistema decide insertar, modificar o eliminar datos acerca de una prenómina.
Complejidad		Alta.
Prioridad		Alto.
Precondiciones		Administrador ya autenticado.
Postcondiciones		Se insertó, modificó o se eliminó usuario(s).
Flujo de eventos		
Flujo básico "Gestionar Prenómina"		
Actor		**Sistema**
1.	Selecciona de la página principal la opción "Gestionar Prenómina".	
2.		Muestra una pantalla con un listado de prenómina y permite insertar, Modificar o Eliminar prenómina(s).
3.	Desea insertar, modificar o eliminar prenómina(s).	
4.		Da la posibilidad de realizar alguna de las siguientes acciones: a) Si decide insertar una prenómina, ir a la sección "Insertar prenómina de persona". b) Si decide modificar los datos de un usuario, ir a la sección "Modificar prenómina". c) Si decide eliminar una(s) prenóminas(s), ir a la sección "Eliminar marcados".

Prototipo elemental de interfaz gráfica



Flujos Alternos

2a. Listado de usuarios vacío

Actor		**Sistema**
1.		Carga una pantalla en blanco sin prenóminas registradas.

3a. Desea realizar una búsqueda

Actor		**Sistema**
1.		Ver el CU Buscar prenómina.

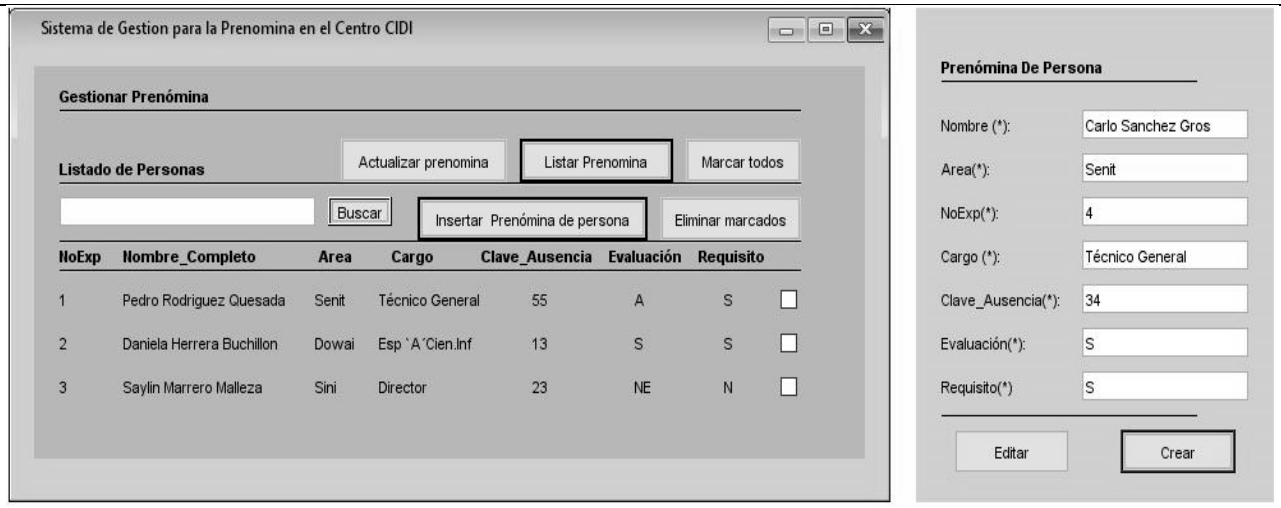
Sección 1: "Registrar prenómina"

Flujo básico Registrar prenómina

Actor		**Sistema**
1.	Presiona el botón "Insertar prenómina".	
2.		Muestra una ventana con los siguientes campos a introducir: • Nombre(s). • Área. • Cargo. • Clave Ausencia.

		• Evaluación. • Requisito. • NoExP. Y el botón “Crear”.
3.	Introduce los datos y presiona el botón “Crear”.	
4.		Verifica que todos los campos estén llenos.
5.		Verifica que los datos introducidos estén correctos.
6.		Verifica que este usuario no exista.
7.		Almacena los datos del usuario y muestra el mensaje “Registro completo”. *Finalizando así el caso de uso.*

Prototipo elemental de interfaz gráfica

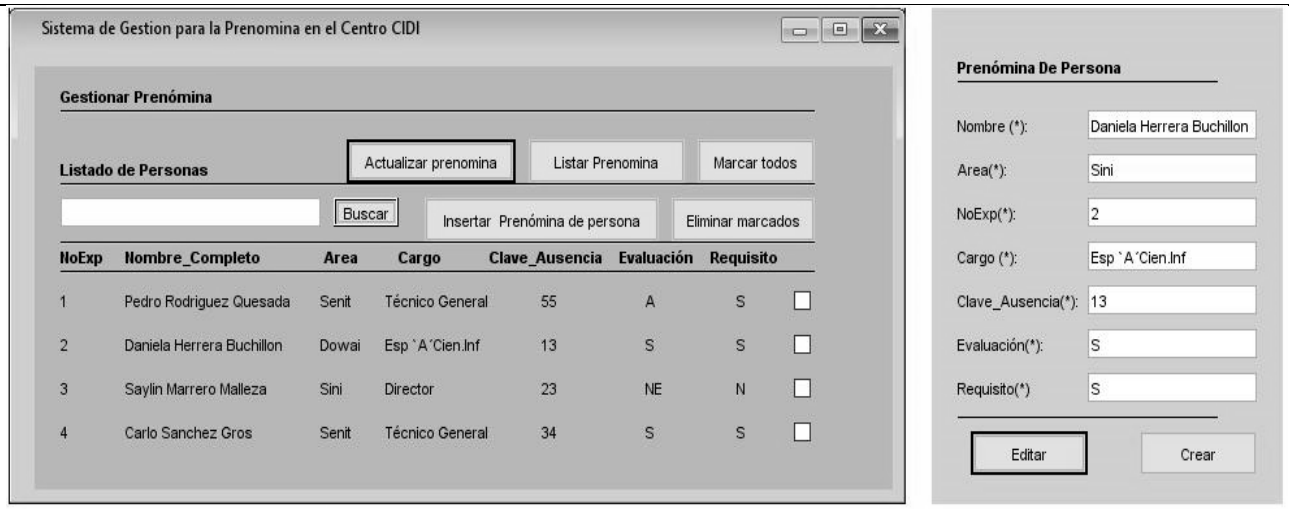


Flujos Alternos

4a. Campos vacíos

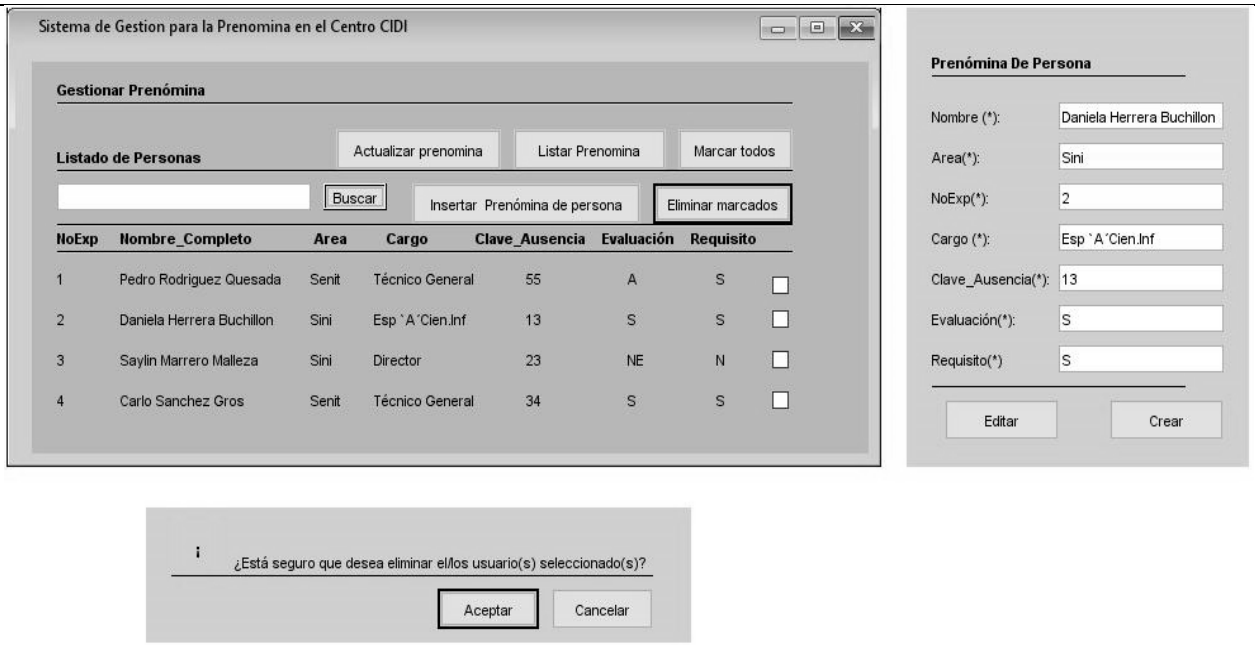
Actor		**Sistema**
1.		Muestra el mensaje “Campos vacíos”.

5a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
6a. Usuario existente		
Actor		**Sistema**
1.		Muestra el mensaje “Usuario existente”.
Sección 2: “Editar Prenómina”		
Flujo básico Editar Prenómina		
	Actor	**Sistema**
1.	Marca la opción “Editar” de uno de los usuarios mostrado y presiona el botón “Actualizar Prenómina”.	
2.		Muestra una ventana con los datos a modificar: • Nombre(s). • Área. • Cargo. • Clave Ausencia. • Evaluación. • Requisito. • NoExP Y el botón “editar”.
3.	Realiza las actualizaciones deseadas y presiona el botón “editar”.	
4.		Verifica que todos los campos estén llenos.
5.		Verifica que los datos introducidos estén correctos.
6.		Actualiza la información incorporada al usuario y se emite un mensaje “Edición satisfactoria”.

		Finalizando así el caso de uso.
Prototipo elemental de interfaz gráfica		
		
Flujos Alternos		
4a. Campos vacíos		
Actor		**Sistema**
1.		Muestra el mensaje “Campos vacíos”.
5a. Datos incorrectos		
Actor		**Sistema**
1.		Muestra el mensaje “Datos incorrectos”.
Sección 3: “Eliminar Prenómina”		
Flujo básico Eliminar Prenómina		
Actor		**Sistema**
1.	Marca la opción “Eliminar Marcados” de una(os) prenómina(s) selecionada(s) y presiona la opción “Aceptar”.	
		Muestra el mensaje de confirmación “¿Está seguro que desea eliminar el(los) usuario(s) seleccionado(s)?”.

		Y los botones “Aceptar” y “Cancelar”.
2.	Presiona el botón “Aceptar”.	
		Elimina el(los) usuario(s) seleccionado(s) y emite el mensaje “Eliminación satisfactoria”. *Finalizando así el caso de uso.*

Prototipo elemental de interfaz gráfica



Flujo Alterno

2a. Cancelar eliminación de usuario

Actor		**Sistema**
1.	Presiona el botón “Cancelar”.	
2.		Vuelve al paso 2 del flujo básico “Gestionar usuario”.

Relaciones	**CU Incluidos**	No Aplica.

	CU Extendidos	CU Buscar prenómina.
Requisitos no funcionales	No Aplica.	
Asuntos Pendientes	No Aplica.	

2.5 Arquitectura utilizada para la propuesta de solución

Modelo – Plantilla – Vista

Según (Infante, 2012), ***Django*** es un marco de trabajo *MTV* del inglés *Model Template View* que es una modificación de MVC (Modelo-Vista-Controlador) debido a que los desarrolladores del marco de trabajo no tuvieron la intención de seguir algún patrón de desarrollo sino hacerlo lo más funcional posible. Para comenzar a entender *Django* se debe tener en cuenta su analogía con MVC de la siguiente forma:

- El modelo sigue siendo Modelo (M) en *Django*
- La vista pasa a llamarse Plantilla (P) en *Django*
- El controlador pasa a llamarse Vista (V) en *Django*

A partir de lo anterior, se asumirá la arquitectura Modelo-Plantilla-Vista (MPV) para *Django*, de la cual se explica su funcionamiento a continuación, el cual se puede observar en la Figura 3:

1. El navegador envía una solicitud.
2. La vista interactúa con el modelo para obtener datos.
3. La vista llama a la plantilla.
4. La plantilla renderiza la respuesta a la solicitud del navegador.

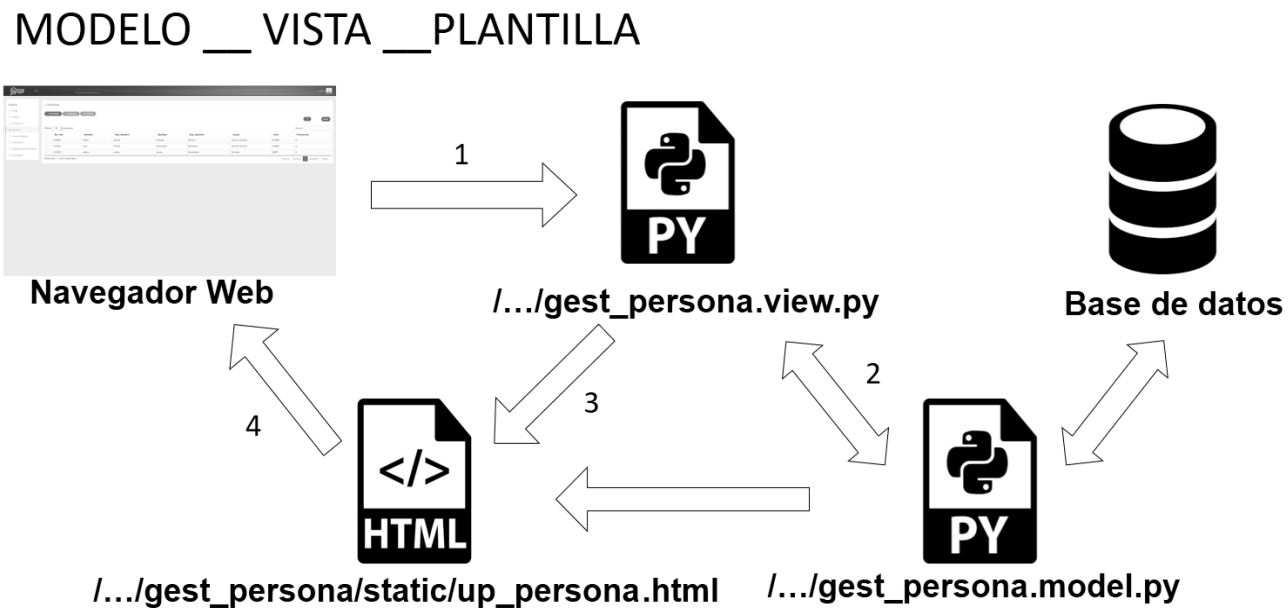


Figura 3. Arquitectura. Fuente: Elaboración propia.

A partir de lo planteado anteriormente, se deriva lo siguiente (Infante, 2012):

- **El modelo:** define lo datos almacenados, es representado en forma de clases de Python, cada tipo de dato que debe ser almacenado se encuentra en una variable con ciertos parámetros, posee métodos también. Todo esto permite indicar y controlar el comportamiento de los datos.
- **La plantilla:** recibe los datos de la vista y luego los organiza para la presentación al navegador web. Básicamente es una página HTML (*HyperText Markup Language*) con algunas etiquetas extras que son propias del Django, dichas etiquetas permiten flexibilidad para los desarrolladores del *frontend*[5].

  La vista: su propósito es determinar qué datos serán visualizados, es representado en forma de funciones. El ORM (*Object Relational Mapping*) de Django permite escribir código Python en lugar de SQL (*Structured Query Language*) para hacer las consultas.

[5] Es la parte del desarrollo web que se dedica de la parte frontal de un sitio web, en pocas palabras, se encarga del diseño de un sitio web, desde la estructura del sitio hasta los estilos como colores, fondos, tamaños hasta llegar a las animaciones y efectos.

2.6 Patrones de diseño

Los diseñadores expertos en orientación a objetos (y también otros diseñadores de software), van formando un amplio repertorio de principios generales y de expresiones que los guían a crear el software. A unos y a otras podemos asignarles el nombre de patrones, si se codifican en un formato estructurado que describe el problema y su solución, y si se les asigna un nombre (Larman, 2017).

A continuación, se presentan los patrones utilizados en el desarrollo del Sistema de Gestión para la Prenómina en el Centro de IdeoInformática.

2.6.1 Patrones Generales de Software para la Asignación de Responsabilidades (*GRASP*):

Según (Grosso, 2011), son patrones basados en la asignación de responsabilidades a objetos. Es una buena práctica para el desarrollo eficaz de la Programación Orientada a Objetos (POO).

- **Experto:** la responsabilidad de realizar una labor es de la clase que tiene o puede tener los datos involucrados. Con el uso de este patrón se alientan las definiciones de clase “sencillas” y más cohesivas que son más fáciles de comprender y de mantener. Brinda soporte a una alta cohesión (Visconti, 2011).

En el sistema, este patrón se utiliza, por ejemplo, para determinar las personas que han cobrado el requisito en un mes determinado. La clase entidad **Prenómina**, es la que tiene la información necesaria para brindar esta información, por tanto, es la clase experta en información.

- **Creador:** se encarga de darle a la clase A la responsabilidad de crear objetos de la clase B. En este caso A es creador de los objetos B (Visconti, 2011).El propósito fundamental de este patrón, es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (Larman, 2017).

Este patrón se puede evidenciar en el uso de la función ***Admin*** la cual se encarga de mostrar los datos del usuario cuyo identificador coincide con el pasado por parámetro, para esto crea una instancia del modelo Usuario y luego es devuelto a la plantilla.

- **Bajo 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 (o débil) acoplamiento no depende de muchas otras clases.

Este patrón ya viene incluido con Django (Django, 2014), 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*[6], en la BD, plantillas HTML, etc., basta solo con realizarlo una sola vez.

- **Alta cohesión:** la cohesión es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Las mismas poseen un número relativamente pequeño de responsabilidades, definiendo así que cada clase realice solo las funcionalidades para las cuales fueron creadas, generando un bajo acoplamiento y fomentando la reutilización (Visconti, 2011).

Una de las características de *Django* es la organización del trabajo en cuanto a la estructura del proyecto, lo cual permite crear y trabajar con clases con una alta cohesión. Por ejemplo, se puede observar en el sistema que cada clase controladora se ajusta a manejar solo las responsabilidades correspondientes a las entidades con las que se relaciona. Esto hace posible que el sistema sea flexible a cambios sustanciales con efecto mínimo.

- **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. El mismo asigna la responsabilidad del manejo de un mensaje de los eventos de un sistema a una clase que represente el "sistema" global (Visconti, 2011).

Este patrón se pone de manifiesto en todo el sistema debido a que cada uno de los eventos generados por el usuario es redirigido a una clase controladora que realiza las operaciones solicitadas, manteniendo siempre la alta cohesión.

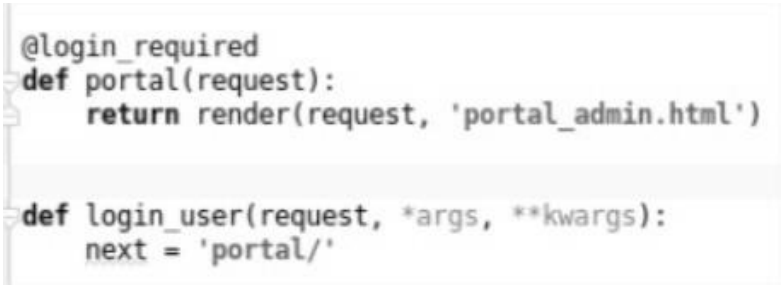
2.6.2 Patrones Gang of Four *(GOF)*

Los patrones de diseño son el esqueleto de las soluciones a problemas comunes en el desarrollo de software. En otras palabras, brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño facilitan la reutilización de arquitecturas y diseños de software exitosos (Pérez, 2017).

[6] Del inglés *Uniform Resource Locator,* Localizador De Recursos Uniforme.

Los patrones de diseño **GOF** son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de software y otros ámbitos referentes al diseño de interacción o interfaces. Por lo que se muestra a continuación los usados en la aplicación.

- El patrón **Decorator** es sencillo e interesante, permite añadir funcionalidades a un objeto en aquellos casos en los que no sea necesario o recomendable hacerlo mediante herencia (Pérez, 2017).



```
@login_required
def portal(request):
    return render(request, 'portal_admin.html')


def login_user(request, *args, **kwargs):
    next = 'portal/'
```

Figura 4. Patrón Decorador. Fuente: Elaboración propia.

- El patrón **Singleton** garantiza que una clase sólo tenga una instancia y proporciona un punto de acceso global a esta instancia (Pérez, 2017).

  En la siguiente figura se muestra un ejemplo de conexión a la base de datos donde se evidencia el uso de este patrón.

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'tesis',
        'USER': 'postgres',
        'PASSWORD': '123',
        'HOST': '127.0.0.1',
        'PORT': '5432'
    }
}
```

Figura 5. Patrón Singleton. Fuente: Elaboración propia.

- El patrón **Mapeo objeto-relacional**(ORM) es una técnica de programación para convertir datos del sistema de tipos utilizado en un lenguaje de programación orientado a objetos utilizado en una base de datos relacional. En la práctica esto crea una base de datos virtual orientada a objetos sobre la base de datos relacional. Esto posibilita el uso de las características propias de la orientación a objetos (esencialmente la herencia y el polimorfismo) (Pérez, 2017).

```python
class DatPersona(models.Model):
    num_expediente = models.CharField(_('Número Expediente'), max_length=11, blank=True, null=True)
    nombre = models.CharField(_('Nombre'), max_length=255, null=False, )
    snombre = models.CharField(_('Segundo Nombre'), max_length=255, blank=True, null=True, default="-")
    papellido = models.CharField(_('Primer Apellido'), max_length=255, blank=True)
    sapellido = models.CharField(_('Segundo Apellido'), max_length=255, blank=True)
    id_cargo = models.ForeignKey(Datacargo, null=True, on_delete=models.CASCADE)
    id_area = models.ForeignKey(Datarea, null=True, on_delete=models.CASCADE)
    id_evaluacion = models.ForeignKey(Dataevaluacion, null=True, on_delete=models.CASCADE)

    class Meta:
        verbose_name = _('Persona')
        verbose_name_plural = _('Personas')
        db_table = 'dat_persona'

    def __str__(self):
        return self.nombre + " " + self.papellido
```

Figura 6. Patrón ORM. Fuente: Elaboración propia.

- El patrón **Observer** puede ser utilizado cuando hay objetos que dependen de otro, necesitando ser notificados en caso de que se produzca algún cambio en él (Pérez, 2017).

```python
from django.db import models
from django.utils.translation import gettext_lazy as _
from modulos.gest_area.models import Datarea
from modulos.gest_cargo.models import Datacargo
from modulos.gest_evaluacion.models import Dataevaluacion
```

Figura 7. Patrón Observer. Fuente: Elaboración propia.

2.7 Diagrama de clases del diseño

En el diseño de clases se resume la definición de las clases que se pueden implementar en el software, se visualizan las relaciones entre ellas y se muestra gráficamente la interacción de los objetos para comunicarse entre sí (Sommerville, 2011). A continuación, se muestra el diagrama de clases para gestionar la prenómina.

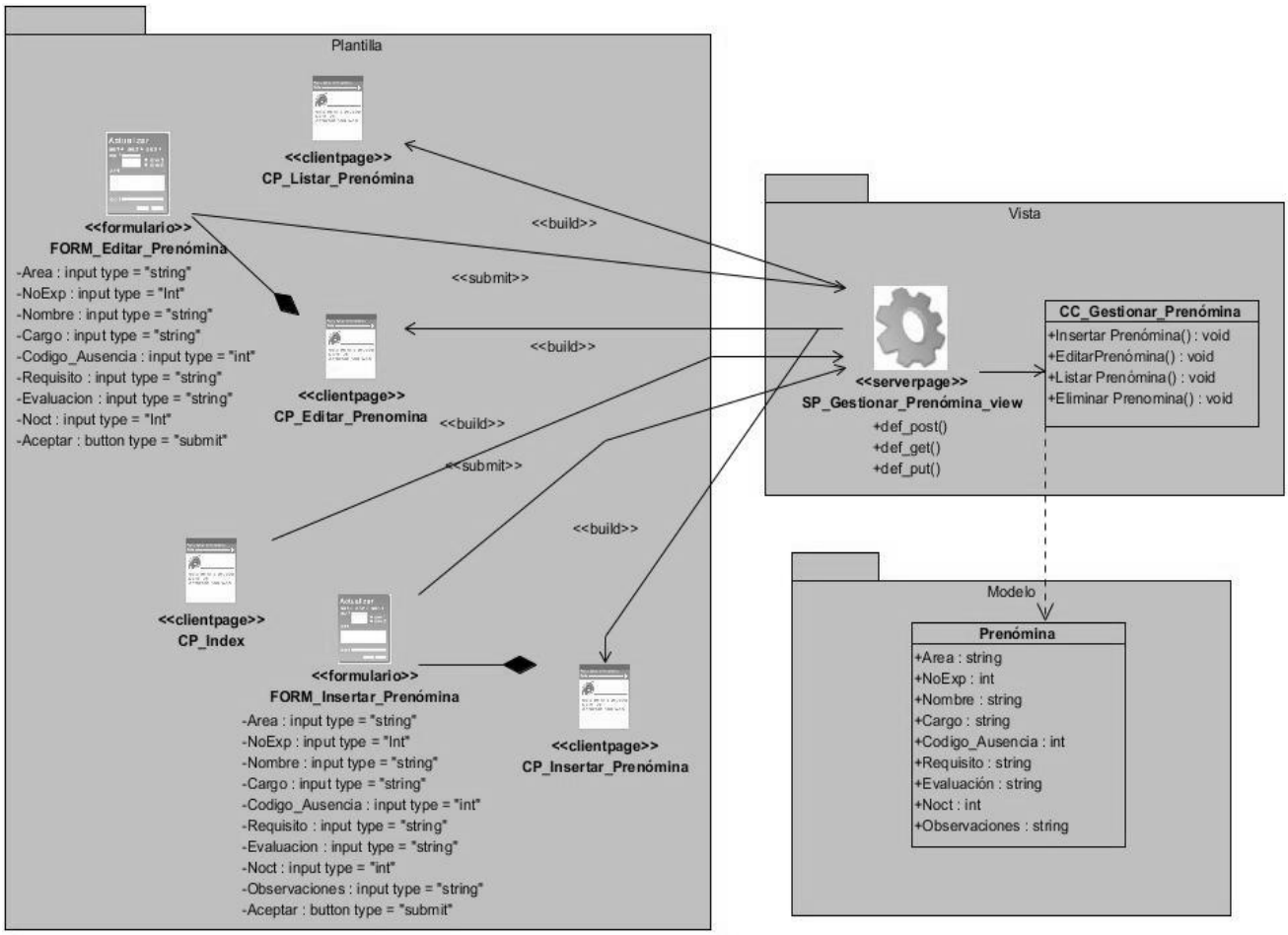


Figura 8. DCD con estereotipos web. Fuente: Elaboración propia

2.8 Modelo de datos

Un modelo de datos está orientado a representar los elementos que intervienen en la realidad o en un problema dado y la forma en que se relacionan dichos elementos entre sí (Sommerville, 2011).

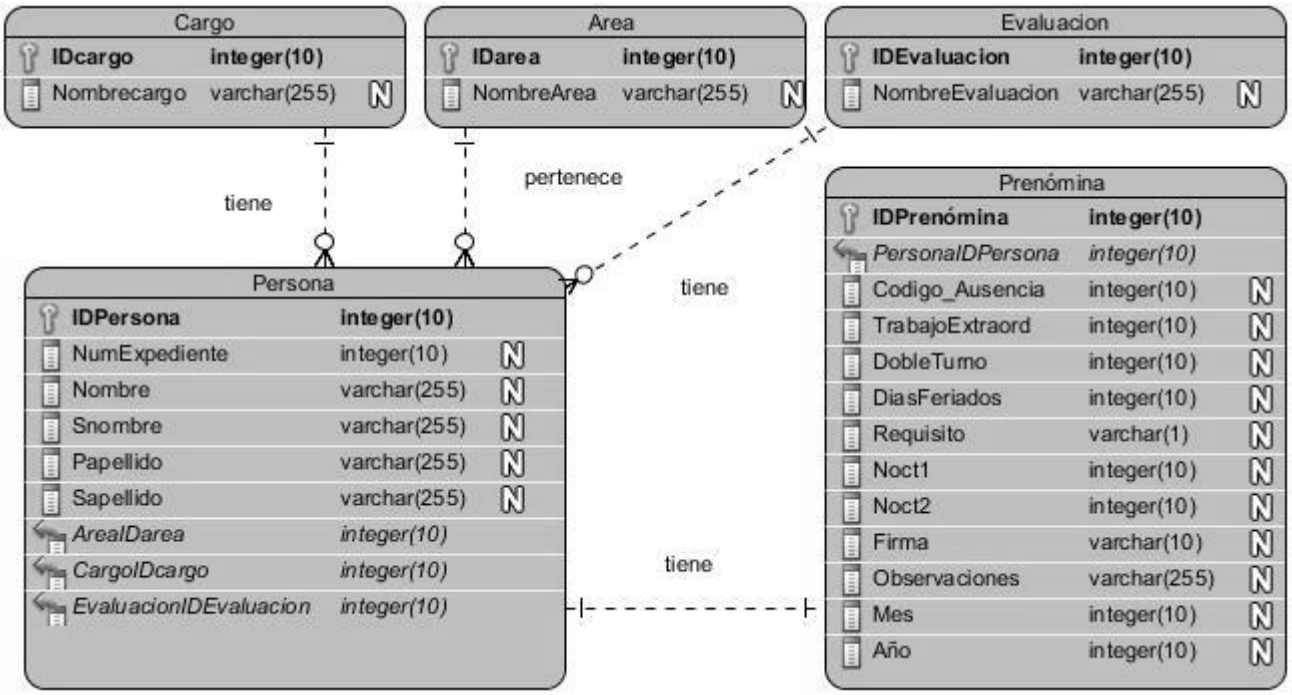


Figura 9. Modelo de datos. Fuente: Elaboración propia.

Conclusiones

Después de realizado el análisis y diseño de la propuesta de solución y haber generado los diferentes artefactos que dispone la metodología *AUP*, se puede concluir lo siguiente:

1. El análisis de las características del sistema y la modelación del dominio, permitió identificar los principales requisitos funcionales y no funcionales del Sistema de Gestión de la Prenómina en el centro de Ideoinformática, los cuales fueron agrupados y categorizados por casos de usos.
2. El diseño de los diagramas de clases y de secuencia, facilitó la visión en cuanto a composición física y lógica del sistema.
3. La generación de todos los artefactos requeridos por el modelo de desarrollo, documentaron la solución propuesta, lo cual facilitó su posterior mantenimiento (actualización o adición de funcionalidades).

CAPÍTULO 3. VALIDACIÓN DE LA PROPUESTA PARA SISTEMA DE GESTIÓN DE PRENÓMINA

Introducción

En la actualidad, los ingenieros de sistemas deben estar en la capacidad de conocer y aplicar las diferentes normas y procedimientos para garantizar la calidad de los productos de software, ya que durante su implementación se pueden presentar equivocaciones que, si no son identificadas y la aplicación se ejecuta, hay un alto riesgo de que la aplicación no funcione de la manera que se desea. Es por ello, que las pruebas del software son la actividad más común de control de la calidad para reducir los riesgos y asegurar el correcto funcionamiento de estos.

Durante este capítulo se realizan las pruebas al sistema implementado con el objetivo de detectar y corregir los errores que no se encontraron durante su implementación.

3.1 Modelo de Despliegue

Representa de forma visual las relaciones físicas que existen entre los componentes de software y *hardware* en el sistema. Los nodos son elementos de *hardware* sobre los cuales pueden ejecutarse los elementos de software. Se utiliza como entrada fundamental en las actividades de diseño e implementación debido a que la distribución del sistema tiene una influencia principal en su diseño. A continuación se presenta al modelo de despliegue correspondiente a la propuesta de solución.

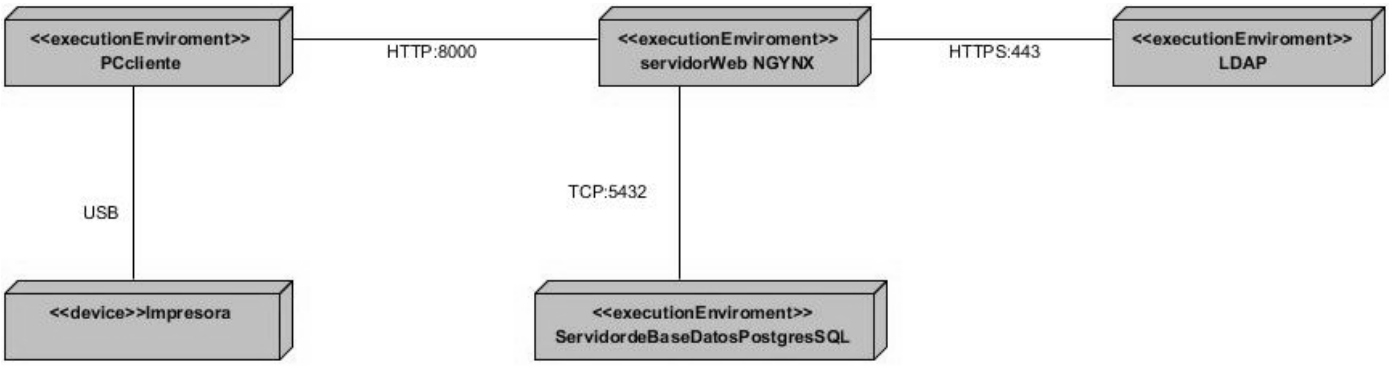


Figura 10. Modelo de Despliegue. Fuente: Elaboración propia.

Descripción de elementos e interfaces de comunicación

Dispositivo PC_Cliente: La estación de trabajo necesita un navegador web para conectarse al sistema hospedado en el servidor de aplicaciones utilizando el protocolo de comunicación HTTP/HTTPS.

Servidor de aplicaciones: Es la estación de trabajo que hospeda el código fuente de la aplicación y que le brinda al usuario las interfaces para realizar los procesos del sistema. Esta estación se comunica con el servidor de base de datos donde se almacenan los datos de la aplicación realizando la comunicación mediante el protocolo TCP.

Servidor de BD: Este servidor es el encargado del almacenamiento de los datos del sistema. Se comunica con el servidor de aplicaciones del sistema, posibilitando el acceso mediante el usuario con privilegios para las operaciones determinadas a realizarse en el mismo.

LDAP: (Protocolo Ligero/Simplificado de Acceso a Directorio) es un protocolo de tipo cliente-servidor para acceder a un servicio de directorio. Los servicios de directorio facilitan el acceso a información organizada a una gran variedad de aplicaciones. Estos servicios le permiten a los usuarios y aplicaciones autorizadas buscar información de personas, computadoras, aplicaciones y dispositivos red.

Impresora: Es un dispositivo periférico del ordenador que permite producir una gama permanente de textos o gráficos de documentos almacenados en un formato electrónico, imprimiéndolos en medios físicos

3.2 Modelo de Componentes

El modelo de componentes es una forma de representar una vista estática del sistema, que representa la organización y dependencia entre los componentes físicos que se necesitan para ejecutar la aplicación. Además, presenta cómo un sistema de software es dividido en componentes y muestra las dependencias entre estos componentes (Castro, 2016).

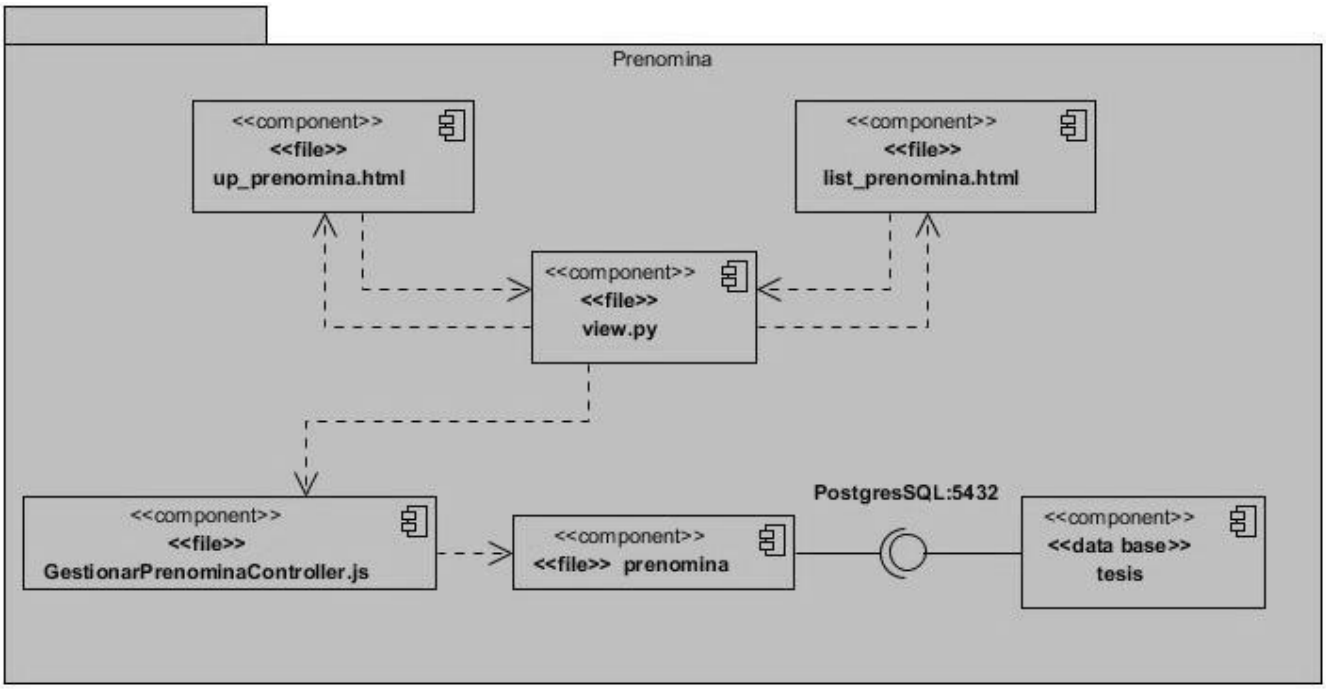


Figura 11. Diagrama de Componente. Fuente: Elaboración propia.

3.3 Estándares de Codificación

Python

Según Guido van Rossum (científico de la computación, conocido por ser el autor del lenguaje de programación Python), el código es leído muchas más veces de lo que es escrito. Por tanto, se hace necesario definir pautas para lograr una mejor legibilidad del código y hacerlo consistente.

Para lograr este objetivo se utilizó la Guía de estilo para el código Python (Rossum, 2013). Esta guía posee una gran cantidad de convenciones para escribir código legible, dentro de las cuales se destacan:

- Usar cuatro espacios por indentación.
- Nunca mezclar tabulaciones y espacios.
- Limitar todas las líneas a un máximo de caracteres (125 en este proyecto).
- Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las definiciones de métodos dentro de una clase son separadas por una línea en blanco.
- Codificación UTF-8 en todos los módulos.

- Las importaciones deben estar en líneas separadas.
- Evitar usar espacios en blanco innecesarios.
- Utilizar el estilo *CamelCase* para nombrar clases, y el *lower_case_with_underscores* para funciones y métodos (Rossum, 2013).

Usar cuatro espacios por indentación

Para código realmente antiguo que no se quiera estropear, se puede continuar usando indentaciones de 8 (ocho) espacios. Las líneas de continuación deben alinearse verticalmente con el carácter que se ha utilizado (paréntesis, llaves, corchetes) o haciendo uso de la "*hanging indent*" (aplicar tabulaciones en todas las líneas con excepción de la primera). Al utilizar este último método, no debe haber argumentos en la primera línea, y más tabulación debe utilizarse para que la actual se entienda como una (línea) de continuación (Rossum, 2013).

```
(function () {
    'use strict';
    angular.module("TesisApp")
        .config(['$stateProvider', '$urlRouterProvider', function ($stateProvider, $urlRouterProvider) {
            $urlRouterProvider.otherwise("dashboard");
            $stateProvider
                .state('persona', {
                    url: "/persona",
                    templateUrl: "/persona/",
                    data: {pageTitle: 'persona'},
                    controller: "PersonaController",
                    controllerAs: "ctrl",
                    resolve: {
                        deps: ['$ocLazyLoad', function ($ocLazyLoad) {
                            return $ocLazyLoad.load({
                                name: 'TesisApp',
                                insertBefore: '#ng_load_plugins_before',
                                files: [
                                    '/static/services/serv_persona.js',
                                    '/static/controllers/PersonaController.js'
                                ]
                            });
                        }]
                    }
                })
```

Figura 12. Indentación. Fuente: Elaboración propia.

Nunca se mezclan tabulaciones y espacios

El método de indentación más popular en Python es con espacios. El segundo más popular es con tabulaciones, sin mezclarse unos con otros. Cualquier código indentado con una mezcla de espacios y tabulaciones debe ser convertido a espacios exclusivamente. Al iniciarse la línea de comandos del intérprete con la opción “-t”, informa en modo de advertencias si se está utilizando un código que mezcla tabulaciones y espacios. Al utilizarse la opción “-tt”, estas advertencias se vuelven errores. Para proyectos nuevos, es preferible únicamente espacios y recomendado antes que tabulaciones. (Rossum, 2013).

```
def create(self, request, *args, **kwargs):
    instance = DatPersona()
    num_expediente = request.data.get('num_expediente', '')
    nombre = request.data.get('nombre', '')
    snombre = request.data.get('snombre', '')
    papellido = request.data.get('papellido', '')
    sapellido = request.data.get('sapellido', '')
    cargo_per = Datacargo.objects.get(id=request.data.get('cargo', {'id': None}).get('id'))
    area_per = Datarea.objects.get(id=request.data.get('area', {'id': None}).get('id'))
    eva_per = Dataevaluacion.objects.get(id=request.data.get('evaluacion', {'id': None}).get('id'))

    persona = DatPersona.objects.create(num_expediente=num_expediente, nombre=nombre, snombre=snombre,
                                        papellido=papellido, sapellido=sapellido, id_cargo=cargo_per,
                                        id_area=area_per, id_evaluacion=eva_per)
    prenomina = DataPrenomina.objects.create(id_persona=persona)
    prenomina.save()
    persona.save()

    return Response({"message": "Datos guardados", "data": request.data})
```

Figura 13. Tabulaciones y Espacios. Fuente: Elaboración Propia

Máxima longitud de las líneas

Limitar todas las líneas a un máximo de 79 caracteres. Todavía hay varios dispositivos que limitan las líneas a 80 caracteres, limitando las ventanas a 80 caracteres se hace posible tener varias ventanas de lado a lado. El estilo en estos dispositivos corrompe la estructura o aspecto visual del código, haciéndolo más dificultoso para comprenderlo. Por lo tanto, se limita todas las líneas a 79 caracteres. En el caso de largos bloques de texto (“*docstrings*” o comentarios), limitarlos a 72 caracteres es recomendado.

El método para “cortar” líneas largas es utilizar la continuación implícita dentro de paréntesis, corchetes o llaves. Además, éstas pueden ser divididas en múltiples líneas envolviéndolas en paréntesis. Esto debe ser

aplicado en preferencia a usar la barra invertida (“\”). La barra invertida aún puede ser apropiada en diversas ocasiones. Por ejemplo, largas, múltiples sentencias *with* no pueden utilizar continuación implícita, por lo que dicho carácter es aceptable (Rossum, 2013):

Figura 14. Longitud máxima de líneas. Fuente: Elaboración propia.

Separar funciones de alto nivel y definiciones de clase con dos líneas en blanco, mientras que las definiciones de métodos dentro de una clase son separadas por una línea en blanco:

Separa funciones de alto nivel y definiciones de clase con dos líneas en blanco. Definiciones de métodos dentro de una clase son separadas por una línea en blanco. Líneas en blanco adicionales pueden ser utilizadas (escasamente) para separar grupos de funciones relacionadas. Se pueden omitir entre un grupo (de funciones) de una línea relacionadas (por ejemplo, un conjunto de implementaciones ficticias). Usa líneas en blanco en funciones, escasamente, para indicar secciones lógicas. Python acepta el carácter control-L (^L) como un espacio en blanco; muchas herramientas tratan a estos caracteres como separadores de página, por lo que puedes utilizarlos para separar páginas de secciones relacionadas en un archivo. Nota: algunos editores y visores de código basados en la web pueden no reconocer control-L como dicho carácter y mostrarán otro glifo en su lugar (Rossum, 2013).

```python
class PersonaViewSet(viewsets.ModelViewSet):
    queryset = DatPersona.objects.all()
    serializer_class = PersonaSerializer
    permission_classes = (permissions.AllowAny,)

    def filter_queryset(self, queryset):
        return self.queryset

    def create(self, request, *args, **kwargs):
        instance = DatPersona()
        num_expediente = request.data.get('num_expediente', '')
        nombre = request.data.get('nombre', '')
        snombre = request.data.get('snombre', '')
        papellido = request.data.get('papellido', '')
        sapellido = request.data.get('sapellido', '')
        cargo_per = Datacargo.objects.get(id=request.data.get('cargo', {'id': None}).get('id'))
        area_per = Datarea.objects.get(id=request.data.get('area', {'id': None}).get('id'))
        eva_per = Dataevaluacion.objects.get(id=request.data.get('evaluacion', {'id': None}).get('id'))

        persona = DatPersona.objects.create(num_expediente=num_expediente, nombre=nombre, snombre=snombre,
                                            papellido=papellido, sapellido=sapellido, id_cargo=cargo_per,
                                            id_area=area_per, id_evaluacion=eva_per)
        prenomina = DataPrenomina.objects.create(id_persona=persona)
        prenomina.save()
        persona.save()

        return Response({"message": "Datos guardados", "data": request.data})

    def update(self, request, *args, **kwargs):
        instance = self.get_object()
        instance.num_expediente = request.data.get('num_expediente', '')
        instance.nombre = request.data.get('nombre', '')
        instance.snombre = request.data.get('snombre', '')
        instance.papellido = request.data.get('papellido', '')
        instance.sapellido = request.data.get('sapellido', '')
        instance.id_cargo = Datacargo.objects.get(id=request.data.get('id_cargo', {'id': None}).get('id'))
        instance.id_area = Datarea.objects.get(id=request.data.get('id_area', {'id': None}).get('id'))
        instance.id_evaluacion = Dataevaluacion.objects.get(id=request.data.get('id_evaluacion', {'id': None}).get('id'))
        instance.save()
        return Response({"message": "Datos guardados", "data": request.data})

    def destroy(self, request, *args, **kwargs):
        arrs = kwargs.get('pk', '').split(',')
        ids = [int(el) for el in arrs]
        DatPersona.objects.filter(pk__in=ids).delete()
        return Response({"message": "Datos guardados", "data": request.data})


class PersonaDatatables(DataTableListCreateApi):
    serializer_class = PersonaSerializer
    search_parameters = ['persona']
    default_order_by = 'persona'
    queryset = DatPersona.objects.all()
```

Figura 15. Funciones de Alto Nivel y Definiciones de clases. Fuente: Elaboración propia.

Codificación UTF-8 en todos los módulos

El código en el núcleo de la distribución de Python siempre debería utilizar la codificación ASCII. Para Python 3.0 y en adelante, UTF-8 es preferible ante Latin-1 (Rossum, 2019).Los archivos usando ASCII no deberían

tener una “*coding cookie*” (especificación de la codificación al comienzo del archivo); de lo contrario, usar \x, \u o \U es la manera para incluir caracteres que no correspondan a dicha codificación en cadenas (*strings*).

Para Python 3.0 y en adelante, la siguiente política es prescrita para la librería estándar (Rossum, 2019): todos los identificadores en la librería estándar de Python deben usar caracteres correspondientes a la codificación ASCII, y deberían usar palabras en inglés siempre que sea posible (en muchos casos, abreviaciones y términos técnicos son usados que no corresponden al idioma). Además, las cadenas literales (*string literals*) y los comentarios deben ser también ASCII. Las únicas excepciones son (a) pruebas para características no ASCII, y (b) nombres de autores. Autores cuyos nombres no están basados en el alfabeto latín deben proveer una transcripción. Se les recomienda a los proyectos de código abierto con gran audiencia adoptar una política similar (Rossum, 2013).

Las importaciones deben estar en líneas separadas

Las importaciones siempre se colocan al comienzo del archivo, simplemente luego de cualquier comentario o documentación del módulo, y antes de globales y constantes.

Las importaciones deben estar agrupadas en el siguiente orden:

1. importaciones de la librería estándar
2. importaciones terceras relacionadas
3. importaciones locales de la aplicación / librería

Al importar una clase desde un módulo que contiene una clase, generalmente está bien realizar esto (Rossum, 2013):

```
from rest_framework import viewsets, permissions
from rest_framework.response import Response
from comun.datatablesOpts import DataTableListCreateApi
from comun.fns_comun import fn_filter_queryset
from modulos.gest_area.models import Datarea
from modulos.gest_evaluacion.models import Dataevaluacion
from modulos.gest_cargo.models import Datacargo
from modulos.gest_prenomina.models import DataPrenomina
from .serializers import PersonaSerializer
from .models import DatPersona
```

Figura 16. Importaciones. Fuente: Elaboración propia.

Evitar usar espacios en blanco innecesarios

Evitar usar espacios en blanco extraños en las siguientes situaciones:

- Inmediatamente dentro de paréntesis, corchetes o llaves.
- Inmediatamente antes de una coma, un punto y coma o dos puntos.
- Inmediatamente antes del paréntesis que comienza la lista de argumentos en la llamada a una función.
- Inmediatamente antes de un corchete que empieza una indexación o "*slicing*" (término utilizado tanto en el ámbito de habla inglesa como española).
- Más de un espacio alrededor de un operador de asignación (u otro) para alinearlo con otro.

Otras recomendaciones

Siempre se encierra estos operadores binarios con un espacio en cada lado: asignación (=), asignación de aumentación (+=, -=, etc.), comparaciones (==, <, >, !=, <>, <=, >=, *in, not in, is, is not*), "*Booleans*" (*and, or, not*). Si se utilizan operadores con prioridad diferente, considera agregar espacios alrededor del operador con la menor prioridad (Rossum, 2013).

Utilizar el estilo *CamelCase* para nombrar clases, y el *lower_case_with_underscores* para funciones y métodos

Nombrar Clases: PalabrasConMayúscula (*CapitalizedWords*) ("*CapWords*" o "*CamelCase*").

Funciones y Métodos: minúscula_con_guiones_bajos (*lower_case_with_underscores*).

Ejemplo a continuación: (Rossum, 2013).

```
class DatPersona(models.Model):
    num_expediente = models.CharField(_('Número Expediente'), max_length=11, blank=True, null=True)
    nombre = models.CharField(_('Nombre'), max_length=255, null=False, )
    snombre = models.CharField(_('Segundo Nombre'), max_length=255, blank=True, null=True, default="-")
    papellido = models.CharField(_('Primer Apellido'), max_length=255, blank=True)
    sapellido = models.CharField(_('Segundo Apellido'), max_length=255, blank=True)
    id_cargo = models.ForeignKey(Datacargo, null=True, on_delete=models.CASCADE)
    id_area = models.ForeignKey(Datarea, null=True, on_delete=models.CASCADE)
    id_evaluacion = models.ForeignKey(Dataevaluacion, null=True, on_delete=models.CASCADE)

    class Meta:
        verbose_name = _('Persona')
        verbose_name_plural = _('Personas')
        db_table = 'dat_persona'

    def __str__(self):
        return self.nombre + " " + self.papellido
```

Figura 17. *CamelCase* y *lower_case_with_underscores*. Fuente: Elaboración propia.

3.4 Validación del sistema de Gestión de la Prenómina para el centro de Ideoinformática

Durante la etapa de implementación, pueden cometerse algunos errores y pueden pasarse por alto algunos elementos que son importantes para el correcto funcionamiento del sistema. Por tal motivo, es esencial llevar a cabo la fase de validación, en la cual, a través de varios tipos y métodos de pruebas de software (estrategia de pruebas), se pretende comprobar el cumplimiento de las especificaciones del diseño y de la codificación, identificar los posibles errores cometidos y validar la solución propuesta en los capítulos anteriores. En este epígrafe se muestran los resultados de la estrategia de prueba diseñada para el Sistema de Gestión de la prenómina en el centro de Ideoinformática.

3.4.1 Pruebas de rendimiento (carga y estrés)

La prueba de carga y estrés se refiere, generalmente, a la práctica de comprobar el comportamiento de una aplicación mediante cargas o entradas pesadas. Las mismas se realizan con el fin de verificar si el sistema satisface los requisitos de rendimiento para situaciones críticas como pueden ser: la cantidad límite de usuarios accediendo de forma concurrente a los servicios brindados, documentos extremadamente grandes, cantidad de transacciones que se pueden procesar de forma concurrente cada minuto, tiempo de respuesta, entre otros (ITI., 2017.).

Para la realización de esta prueba se utilizó la herramienta *Apache JMeter*. Las pruebas se realizaron desde un ordenador con 4GB de RAM, microprocesador Intel Core i3 con 2.00 GHz y sistema operativo *GNU/Linux*.

Tabla 3. Prueba de Apache Jmeter. Fuente: Elaboración propia

Etiqueta	**# Muestras**	**Media**	**Mediana**	**Linea 90%**	**Mín**	**Máx**	**% Error**	**Rendimiento (peticiones/ segundos)**	**Kb/s Recibidos**
Área	100	450	513	708	13	828	0.00 %	38.0/min	2.6
Cargo	100	786	686	1320	4	1532	0.00 %	37.7/min	1.7
Evaluación	100	1100	1128	1889	32	2190	0.00 %	37.4/min	1.7
Persona	100	854	959	1327	55	2068	0.00 %	37.1/min	1.7
ClaveAusencia	100	751	951	1253	63	2108	0.00 %	36.9/min	1.7
Prenómina	100	668	748	1188	81	2117	0.00 %	36.7/min	1.7

A continuación, se describen las variables que miden el resultado de las pruebas de carga y estrés realizadas al sistema:

Muestra: Cantidad de peticiones realizadas para cada *URL*.
Media: Tiempo promedio en milisegundos en el que se obtienen los resultados.
Mediana: Tiempo en milisegundos en el que se obtuvo el resultado que ocupa la posición central.
Línea 90 %: Máximo tiempo utilizado por el 90 % de la muestra, al resto de la misma le llevo más tiempo.
Min: Tiempo mínimo que demora un hilo en acceder a una página.
Max: Tiempo máximo que demora un hilo en acceder a una página.
% Error: Por ciento de error de las páginas que no se llegaron a cargar de manera satisfactoria.
Rendimiento (peticiones/segundos): El rendimiento se mide en cantidad de solicitudes por segundo.
Kb/s: Velocidad de carga de las páginas.

Como se muestra en la siguiente tabla, se simularon las peticiones realizadas al sistema por un total de 50 usuarios simultáneamente en cada caso, para 200 muestras los cuales realizan hasta 5 peticiones por segundo. Se obtuvieron los siguientes resultados:

Tabla 4. Resumen de Prueba Jmeter. Fuente: Elaboración Propia.

Usuarios	**# Muestras**	**Media**	**Mediana**	**Linea 90%**	**Mín**	**Máx**	**% Error**	**Rendimiento (peticiones/ segundos)**	**Kb/s Recibidos**
50	600	768	680	1317	4	2190	0.00 %	3.7	10.8

Las pruebas realizadas muestran que el sistema es capaz de responder a 600 peticiones de 50 usuarios conectados simultáneamente en un tiempo promedio de 3.7 segundos con 0.00 % de error, Atendiendo a los resultados arrojados y a las prestaciones del *hardware* donde se realizaron las pruebas se considera que constituye un resultado satisfactorio.

3.4.2 Pruebas funcionales

Este tipo de pruebas se realiza para validar que las funcionalidades implementadas funcionen de acuerdo a las descripciones de los requisitos especificados previamente. Para la realización de estas pruebas, se emplean dos métodos: el método de Caja Blanca y el método de Caja Negra. El primero está encaminado a pruebas al código de las aplicaciones; mientras que el segundo, a través del estudio de los datos de entrada y de salida, permite encauzar la atención en el funcionamiento de la interfaz del sistema.

Se denomina caja negra a aquel elemento que es estudiado desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce. Las pruebas de caja negra se llevan a cabo sobre la interfaz del software, para obviar el comportamiento interno y la estructura del programa (García, 2013).La prueba de caja negra no es una alternativa a las técnicas de prueba de caja blanca. Más bien se trata de un enfoque complementario que intenta descubrir diferentes tipos de errores (Pressman, 2010).

Los casos de prueba de la caja negra pretenden demostrar que:

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

A continuación, se muestra el caso de prueba resultante en las iteraciones realizadas al siguiente caso de uso.

Tabla 5. Caso de Prueba Gestionar persona. Fuente: Elaboración propia.

Caso de prueba Gestionar persona	
Código de caso de prueba	**Nombre de requisito:** Gestionar persona
Nombre de la persona que realiza la prueba: Luis Ernesto Fernández Buchillón	
Descripción de la prueba: Prueba a la funcionalidad gestionar persona.	
Entrada/Pasos de ejecución: Se seleccionan los siguientes datos para gestionar persona.	
Nombre **Segundo nombre** **Primer apellido** **Segundo apellido** **No. Expediente** **Cargo** **Área** **Evaluación** El asistente de control selecciona los datos para gestionar a una persona y si los datos están correctos se almacena en el sistema, si existe algún dato incorrecto el sistema mostrará un mensaje de error y señalará el campo erróneo en rojo para su posterior corrección.	
Evaluación de la prueba: Satisfactoria.	

Resultado de las pruebas funcionales

Para validar que el sistema cumpla con las funciones específicas para las cuales ha sido creado en la primera iteración se encontraron 35 no conformidades y se resolvieron 7. En la segunda iteración quedando pendiente 28 no conformidades, se resuelven 20. En la tercera iteración se resolvieron todas las no conformidades restantes y en la cuarta iteración no se arrojó ninguna nueva. En la tabla se

muestran los resultados obtenidos en las iteraciones de pruebas realizadas al Sistema para la Gestión de la Prenómina en el centro de Ideoinformática.

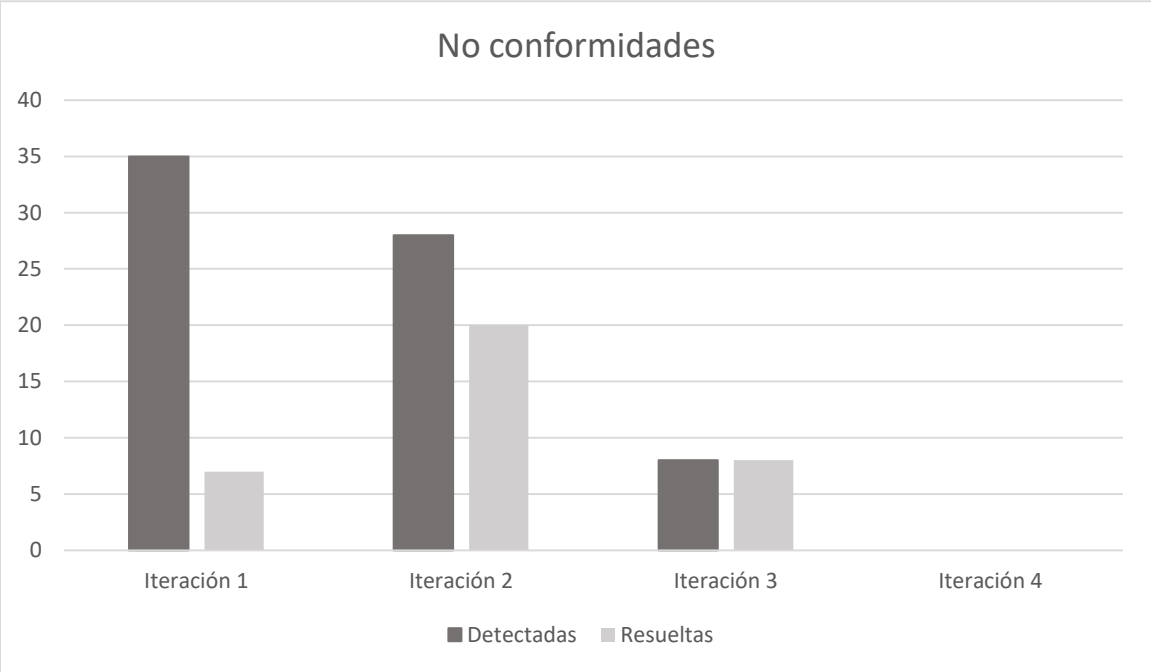


Figura 18. Cantidades de No Conformidades en las Pruebas Funcionales. Fuente: Elaboración propia.

Entre las no conformidades detectadas en el proceso de pruebas funcionales se encuentran:

- Errores de estructuración de los contenidos mostrados en las vistas.
- Los datos introducidos por el usuario de forma incorrecta son guardados en la base de datos sin validación previa.
- Opciones que no funcionan.
- Validaciones que no admite el sistema.
- Campos en blanco.

3.4.3 Pruebas de seguridad

Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de hechos de acceso impropio. Durante las pruebas de seguridad, el responsable de la prueba desempeña el papel de un

individuo que desea entrar en el sistema. Debe intentar conseguir las claves de acceso por cualquier medio, debe bloquear el sistema, negando así el servicio a otras personas (Pressman, 2005).

Para la realización de este tipo de prueba, se empleó la herramienta *Acunetix Web Vulnerability Scanner*. En una primera iteración, se obtuvo un total de veintidós (22) no conformidades, divididas en seis (6) de nivel medio, once (11) de nivel bajo y cinco (5) de carácter informacional. En la segunda iteración se resolvieron tres (3) de nivel medio, cinco (5) de nivel bajo y dos (2) de carácter informacional. En la tercera iteración se resolvieron las no conformidades pendientes. De las de nivel medio, destacó el uso del protocolo no seguro para el envío de datos, así como los mensajes de error que se muestran en el modo DEBUG de Django para el desarrollo. Las de nivel bajo estuvieron relacionadas con problemas para la protección contra ataques de fuerza bruta a la página de autenticación, así como directorios que pueden ser accesibles directamente sin pasar la autenticación y la protección de las *cookies* y las sesiones en el navegador. De carácter informativo fueron detectadas una dirección de correo y una posible cuenta de usuario en un fichero. Los resultados antes descritos, se muestran a continuación en la siguiente gráfica:

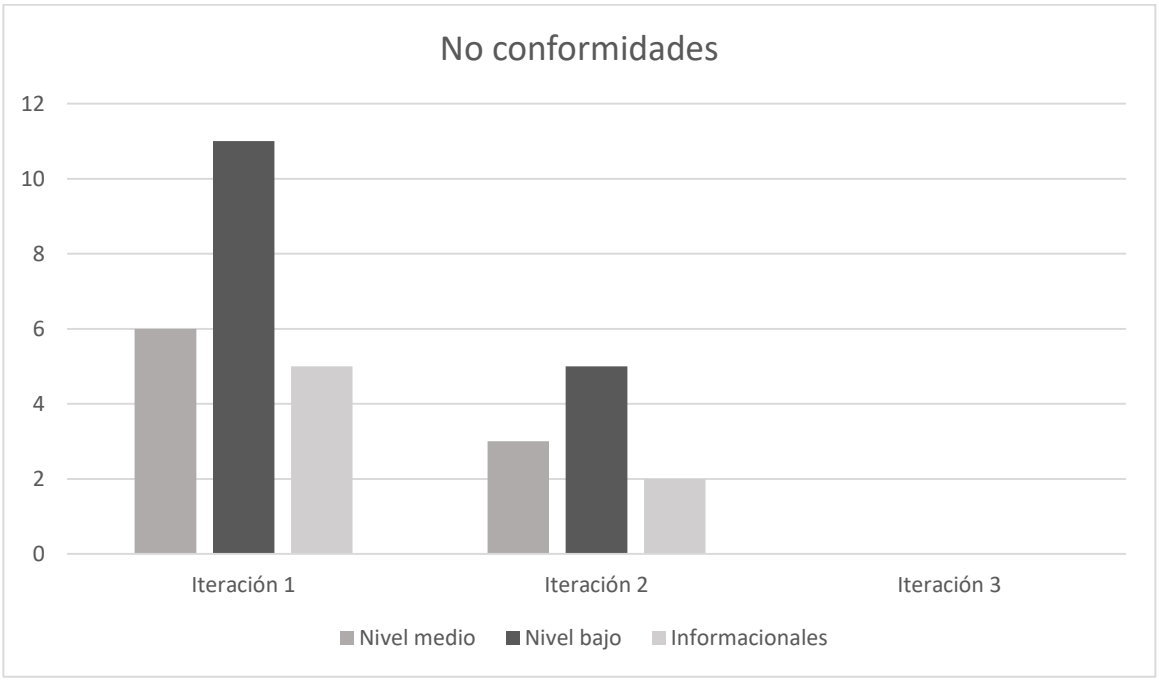


Figura 19. Cantidad de No Conformidades en las Pruebas de Seguridad. Fuente: Elaboración propia.

3.4.4 Pruebas de usabilidad

Se puede decir que el proceso de prueba de usabilidad se enfoca en satisfacer las necesidades de los usuarios finales basados en métricas definidas durante la planeación. Para la realización de las pruebas de usabilidad, se hace uso de la “Lista de Chequeo de Usabilidad para sitios web”, desarrollada por los especialistas del grupo de Seguridad del Departamento de Evaluación de Productos de Software (DEPSW), perteneciente al Centro Nacional de Calidad de Software (CALISOFT). A continuación, se muestran los resultados de dichas pruebas.

Tabla 6. Resultado de Aceptación. Fuente: Elaboración propia.

Categoría de los indicadores	**Indicadores**	**Proceden**	**Correctos**	**Incorrectos**
Visibilidad del sistema	17	14	12	2
Lenguaje común entre sistema y usuario	12	10	10	0
Libertad y control por parte del usuario	29	27	17	10
Consistencia y estándares	33	28	21	7
Estética y diseño minimalista	18	13	9	4
Prevención de errores	8	8	8	0
Ayuda y documentación	11	7	3	4
Flexibilidad y eficiencia	6	6	4	2
Ayuda a los usuarios a reconocer, diagnosticar y recuperarse de los errores	11	7	3	4
Total	**144**	**120**	**87**	**33**

Como se observa en la tabla, de los 144 parámetros originales de la lista de chequeo, solo proceden a evaluarse 120 de los mismos ya que los restantes 24 no son compatibles con las especificaciones del sistema por lo que no se evalúan en este caso. En la primera iteración se evaluaron como correctos 87 parámetros, identificando 33 no conformidades, para un 73.10 % de usabilidad. Los principales problemas estuvieron relacionados con la libertad y control por parte del usuario en el sistema.

3.4.5 Pruebas de aceptación

Las pruebas de aceptación son básicamente pruebas funcionales sobre el sistema completo, ya que tienen como objetivo obtener la aceptación final del cliente antes de la entrega del producto para su utilización. Son realizadas por el usuario final permitiendo la valoración del producto, donde el cliente confirma que las funcionalidades exigidas y descritas en los casos de uso funcionan correctamente. Intentan encontrar errores de las siguientes categorías: funciones incorrectas o ausentes, errores de interfaz, errores en estructuras de datos o en accesos a bases de datos, errores de rendimiento y errores de inicialización y terminación (Fontanela, 2003).

Cuando se realizan este tipo de pruebas, el producto está listo para implantarse en el entorno del cliente. Para su desarrollo se emplearon casos de prueba de aceptación donde se describe el proceso a realizar y en los cuales tuvieron destacada participación miembros del centro para el cual se desarrolla esta aplicación. A continuación, se describe uno de los casos de pruebas de aceptación, para el requisito Gestionar prenómina.

Tabla 7. Prueba de Aceptación para el cliente caso de prueba Gestionar prenómina. Fuente: Elaboración propia.

Caso de prueba Gestionar prenómina	
Código de caso de prueba	**Nombre de requisito:** Gestionar prenómina
Nombre de la persona que realiza la prueba: Luis Ernesto Fernández Buchillón	
Descripción de la prueba: Prueba a la funcionalidad gestionar prenómina	
Entrada/Pasos de ejecución: Se seleccionan los siguientes datos para gestionar prenómina.	

Clave Ausencia **Trabajo Extraordinario** **Doble Turno** **Días Feriados** **Noct1** **Noct2** **Firma** **Observaciones** El asistente de control selecciona los datos para gestionar a una persona y si los datos están correctos se almacena en el sistema, si existe algún dato incorrecto el sistema mostrará un mensaje de error y señalará el campo erróneo en rojo para su posterior corrección.
Evaluación de la prueba: Satisfactoria.

Resultados de las pruebas de aceptación

Para verificar el correcto funcionamiento de las funcionalidades del sistema esperadas por el cliente se realizaron las pruebas de aceptación, las cuales arrojaron un número de no conformidades, las que fueron corregidas de manera satisfactoria paulatinamente en cada iteración correspondiente como se muestra a continuación en la Tabla 9. En la primera iteración se encontraron 13 no conformidades y se resolvieron 8, en la segunda iteración no se arrojaron nuevas no conformidades y se resolvieron las 5 pendientes y en la tercera iteración no se arrojó ninguna nueva no conformidad del sistema. Dentro de las no conformidades que se arrojaron se encuentran faltas de ortografía en el formulario, los campos del formulario no estaban correctamente validados y los datos se introducían de forma incorrecta a la base datos creando conflicto con el Modelo.

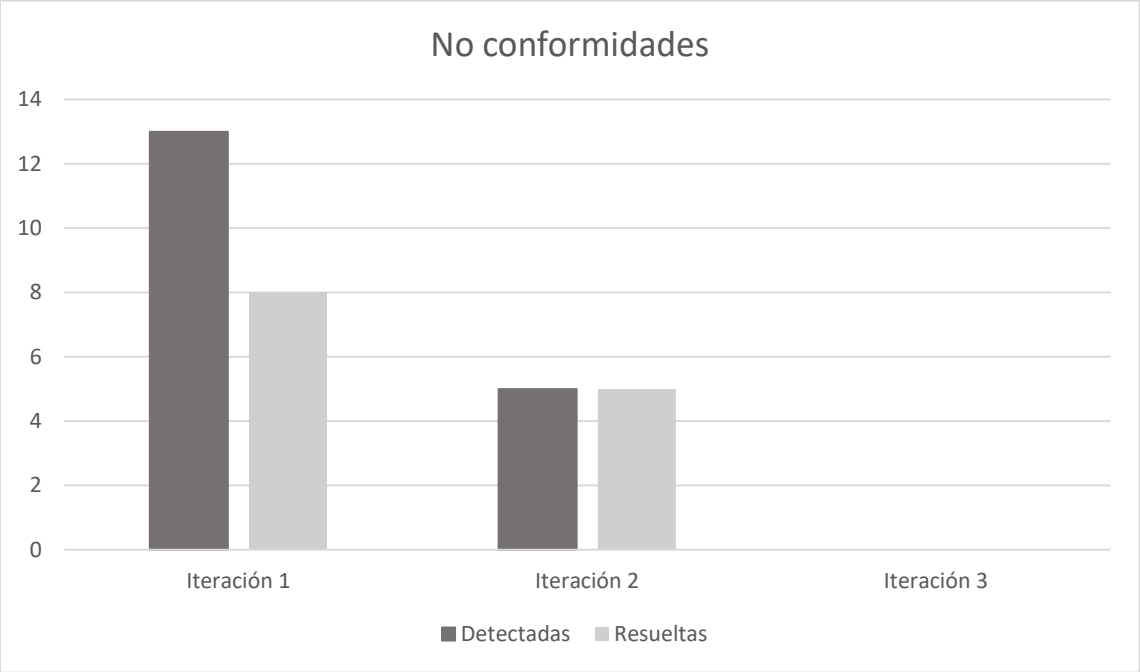


Figura 20. Cantidades de no conformidades en las pruebas de aceptación. Fuente: Elaboración propia.

Con las pruebas de aceptación el cliente válido los requerimientos previstos quedando satisfecho con la implementación del sistema para gestionar la prenómina en el área. El acta de aceptación puede consultarse en el anexo 5.

Conclusiones

1. La construcción del diagrama de componentes y la descripción de los mismos, brindó una mayor comprensión del sistema desarrollado.
2. Al aplicar los estándares de codificación se logró una mayor legibilidad, limpieza y organización del código.
3. Las diferentes pruebas realizadas permitieron demostrar la calidad del sistema de gestión de prenómina desarrollado.

CONCLUSIONES GENERALES

Al culminar la presente investigación se arribó a los siguientes resultados:

- La construcción de los referentes teóricos sustentó la investigación relacionada con el desarrollo de herramientas para la gestión de la información de los recursos humanos.
- El estudio de los sistemas homólogos en los diferentes ámbitos arrojó que no existía un sistema que brindara la facilidad de gestionar la prenómina, lo que permitió una mayor comprensión del objeto de estudio, quedando evidenciada la necesidad de la implementación de un nuevo sistema.
- La caracterización de varias herramientas y tecnologías ayudaron a escoger las adecuadas para la ejecución del sistema, lo que favoreció a un correcto proceso de desarrollo, siendo guiado por la metodología AUP-UCI en el escenario No.2.
- La identificación de los requisitos funcionales garantizó que la propuesta de solución respondiera a los requerimientos planteados por el cliente, quedando evidente lo que el sistema debía hacer y con los requisitos no funcionales se plasmaron claramente las cualidades del producto.
- La modelación de los artefactos facilitó el soporte a la implementación de los requisitos previamente mencionados por el cliente y sentó las bases para la organización lógica del código fuente.
- El sistema desarrollado garantizó la obtención de un sistema para la gestión de la prenómina en el Centro de Ideoinformática y así facilitar la gestión de la información.
- La realización de las pruebas de software eliminó las no conformidades detectadas en el sistema, permitiendo su corrección, con el objetivo de lograr un producto con calidad y que cumpliera las necesidades del cliente.

RECOMENDACIONES

- Realizar el despliegue del sistema a todas las áreas de la Facultad 1 y posteriormente a otras áreas de la UCI.
- Incorporar nuevas funcionalidades al sistema como controlar y gestionar la asistencia a partir del propio sistema.

REFERNCIAS BIBLIOGRÁFICAS

1. **Abalos, J.C. 2012.** *solución para la gestion de productos y servicios .* 2012.
2. **Alegsa, L. 2018.** [En línea] 2018. http://www.alegsa.com.ar/Dic/sistema_informatico.php.
3. **Angulo, R. 2017.** clickbalance.com. [En línea] 2017. https://clickbalance.com/blog/contabilidad-y-administracion/nomina-descubre-en-que-consiste-la-prenomina/.
4. **Bernal, Y. 2009.** *Desarrollo de aplicaciones comerciales en Java.* 2009.
5. **Camacho, C.A.T. 2017.** *Analisis, Diseño y Desarrollo del modulo de gestion de nesesidades y solicitud de disponibilidad para el sistema ARGO(Sistema Contractual y Compras).* 2017.
6. **Castro, S Romero, J. 2016.** *Herramienta de gestión de proyecto empleando buenas practicas de las metodologías ágiles.* s.l. : Universidad de la Ciencias Informáticas, 2016.
7. **Cederholm, D y Zeldman, J. 2010.** *CSS3 For Web Desingners Edition No2.* 2010.
8. **Cervantes, A. 2012.** Grupo Tress Internacional. [En línea] 2012. http://www.tress.com.mx/esp/LinkClick.aspx?fileticket=Vby5GtGCOn8%3D&tabid=61..
9. **Decreto No 281, Cuba. 2007.** *Decreto No 281.* Cuba : s.n., 2007.
10. **Desoft, SOFTWARE, E.N. 2011.** *Fastos y pagos RRHH y Nóminas.* s.l. : https://www.ecured.cu/Fastos_%26_Pagus . http://www.expomatanzas.cu/empresa.php?prd=936&emp=151&catprd=5. http://www.expomatanzas.cu/ofertas/fastos, 2011.
11. **Django. 2014.** *Django Software Foundation. The django Book.* 2014.
12. **Figueroa, R G Solis, C J Coelho, F. 2012.** *Metodologias tradicionales VS. Metodologias agiles.* 2012.
13. **Fontanela, C, Suarez, P. 2003.** *Documentación y pruebas del paradigma de objetos.* 2003.
14. **García, M L. 2013.** *Ingenieria de Software 2.* 2013.
15. **García, M, Sanchez, K Zapata, A. 2008.** *Sanchez, K Zapata, A.* 2008.
16. **Gracía, A. T. 2016.** [En línea] 2016. https://www.researchgate.net/publication/284909371_Valoracion_de_las_TIC_por_los_estudiantes_universitarios_y_su_relacion_con_los_enfoques_de_aprendizaje.
17. **Grosso, A. 2011.** *Practiccas de software. Experiencias sobre la ingenieria y Management de software.* 2011.

18. **Gutíerrez, E. 2009.** *JavaScript Conceptos básicos y avanzados.* Barcelona : s.n., 2009.
19. **Infante, S. 2012.** *Curso Django para perfeccionistas con deadlines.* 2012.
20. **ITI. 2017..** *Instituto Tecnológico de Informática. Testeo de estrés y carga. .* 2017.
21. **Larman, C. 2007.** *Paradigma Visual para UML UML, P. visual.* 2007.
22. **—. 2017.** *UML y Patrones.* 2017.
23. **Lubbers, P. 2010.** *Pro HTML5 Programming: Powerful APIs for Richer Internet Application Development.* 2010.
24. **Lynch, K. 2009.** *sqlmanage.* 2009.
25. **M A, Rafael. 2010.** *Sobre PostgresSQL.* 2010.
26. **M.T, Elena. 2008.** *Bibdigital.* 2008.
27. **Mok, P C.A I.C. 2012.** Infinite Consulting. [En línea] 2012. http://www.infiniteconsulting.com.ve/.
28. **Muñoz, W.R. 2015.** *Portal web de la facultad 1 de UCI.* 2015.
29. **Ordoñez, N. 2017.** PROCEDIMIENTO GESTIÓN NOMINA DEL PROCESO ADMINISTRACION Y GESTION LEGAL DEL PERSONAL. [En línea] 2017. https://supernotariado.gov.co/portalsnr/images/archivosupernotariado/SIG2011/GestionHumana/Procedimientos/Administracion/procesos/gestionomina.pdf.
30. **Paez, L. 2016.** Doce2. [En línea] 2016. http://doce2.softonic.com/. http://www.abcdatos.com/programa/control-asistencia.html.
31. **Peralta, C y Duran, D. 2014.** *Módulos de edición de plantillas y recepción de órdenes de impresión para el sistema de personalización de documentos de identidad basado en tecnologías libres.* 2014.
32. **Pérez, Y M. 2017.** *Herramienta de gestion de la configuracion para la DIvisión de Automática Y Comunicaciones.* 2017.
33. **Pressman, R. 2005.** *Software Engineering A Practitioner's Approach Seventh Edition.* 2005.
34. **Pressman, R. 2010.** *Ingeniería del software, un enfoque práctico.7ma Edición.* 2010.
35. **Python. 2017.** *Guía de estilo para el código python.* 2017.
36. **Rodríguez Sánchez, T. 2014.** *Metodología de desarrollo para la Actividad productiva de la UCI.* 2014.
37. **Rossum, G V. 2013.** *Guia de estilo para Python - PEP8.* 2013.
38. **—. 2019.** Rodas XXI. [En línea] 2019. http://www.rodasxxi.cu/..

39. **—. 2019.** www.python.org. [En línea] 2019. https://www.python.org/dev/peps/pep-3131/ y https://www.python.org/dev/peps/pep-3120/.
40. **Sánchez, D L. 2010.** *Supervisión de agentes en tiempo real del Modulo Call Center de Elastix.* 2010.
41. **Silveiro, Y A C. 2015.** Sistema para control de tiempo y asistencia de personal. *Sistema para control de tiempo y asistencia de personal.* [En línea] 2015. httpswww.monografias.comtrabajos107sistema-control-tiempo-y-asistencia-personalsistema-control-tiempo-y-asistencia-personal.shtml.
42. **Sommerville, I. 2011.** *Ingeniería de Software Septima Edición.* 2011.
43. **Sommerville, I. 2011.** *Ingenieria de Software.* 2011. Septima Edición.
44. **Visconti, M Astudillo, H. 2011.** *Fundamentos de Ingenieria de Software.* 2011.
45. **Worthington, J. 2018.** PyCharm. *https://blog.jetbrains.com/blog/2018/07/31/a-chat-with-jonathan-worthington-creator-of-comma-a-perl-6-ide-built-on-top-of-the-intellij-platform/.* [En línea] 2018. https://www.jetbrains.com/pycharm/.

ANEXOS

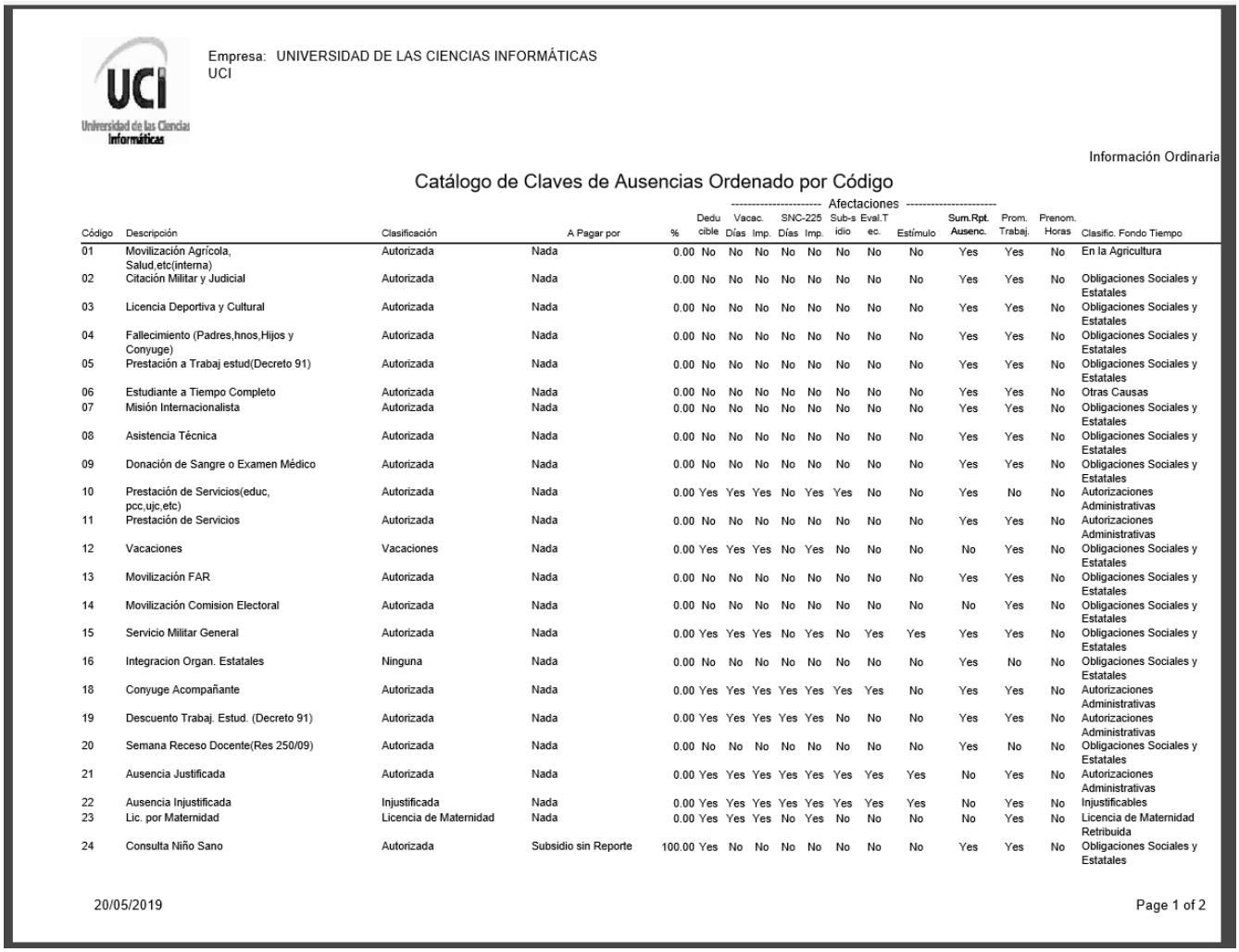
Anexo 1. Guía de entrevista dirigida a los directivos del Centro CIDI. Fuente: Elaboración propia.

El siguiente cuestionario se ha diseñado con el propósito de obtener información relevante que pueda servir para diseñar el sistema para gestionar la prenómina de una entidad, así como para conocer el proceso en aras de potenciar su desarrollo a través de la entrevista.

La información que se obtenga será utilizada para la mejora del proceso y se garantizará su absoluta reserva. Por favor, responda con sinceridad las siguientes preguntas.

1. ¿Cómo se realiza el proceso de la gestión de la prenómina en el centro de Ideoinformática?
2. ¿Cada cuánto tiempo se realiza este proceso?
3. ¿Quiénes son los principales involucrados en este proceso?
4. ¿Cuántos departamentos posee el centro? ¿Cuántos recursos humanos?
5. ¿Existe algún subproceso que influya en el proceso principal que se desea informatizar?
6. ¿Cuáles son las principales funcionalidades que debe cumplir el sistema?

Anexo 2. Catálogo de claves de ausencia ordenado por código. Página 1. Fuente: Dirección de capital humano.



Empresa: UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS
UCI

Información Ordinaria

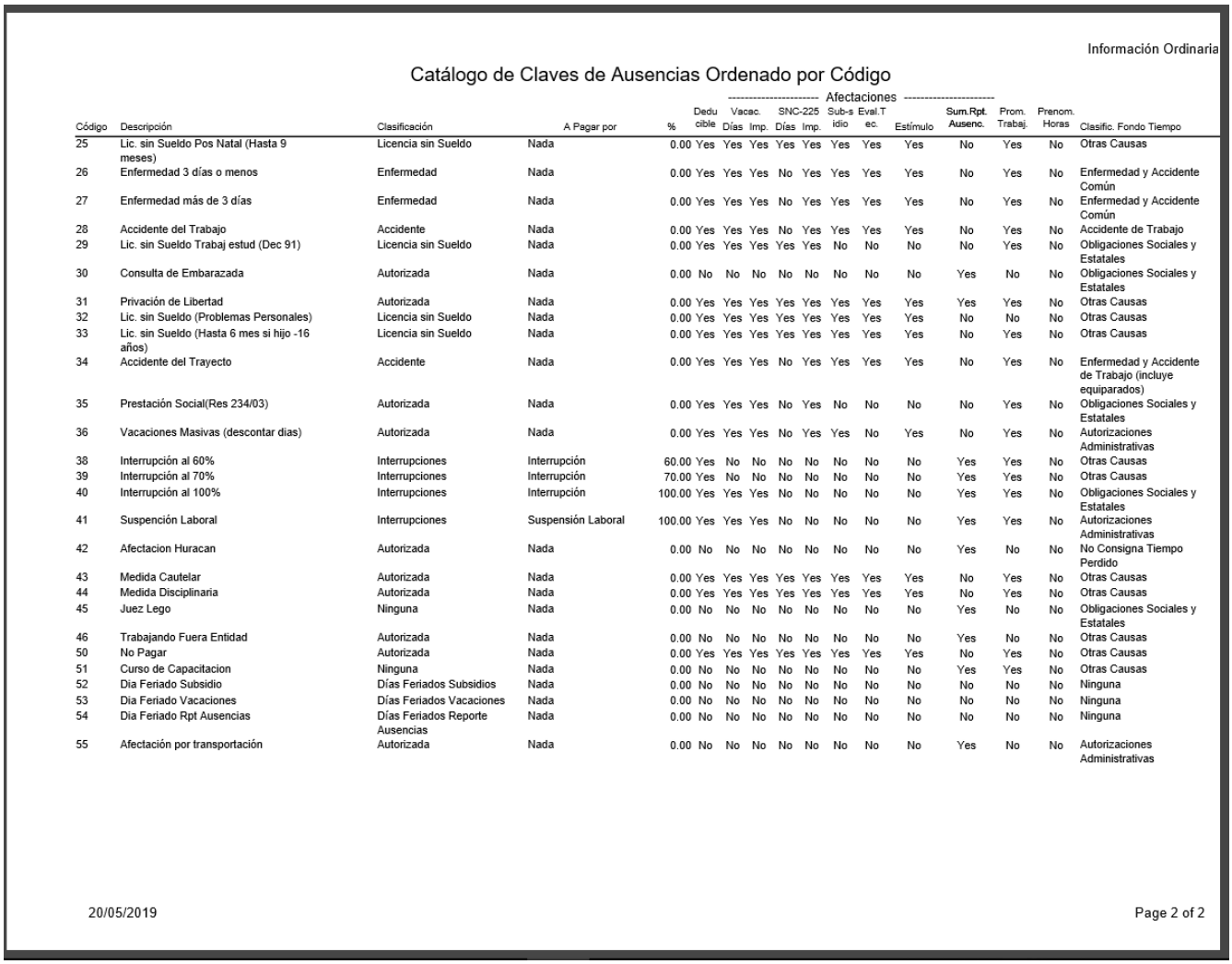
Catálogo de Claves de Ausencias Ordenado por Código

						Afectaciones									
Código	Descripción	Clasificación	A Pagar por	%	Deducible	Vacac. Días	Vacac. Imp.	SNC-225 Días	SNC-225 Imp.	Sub-sidio	Eval.Tec.	Estímulo	Sum.Rpt. Ausenc.	Prom. Trabaj.	Prenom. Horas	Clasific. Fondo Tiempo
01	Movilización Agrícola, Salud,etc(interna)	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	En la Agricultura
02	Citación Militar y Judicial	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
03	Licencia Deportiva y Cultural	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
04	Fallecimiento (Padres,hnos,Hijos y Conyuge)	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
05	Prestación a Trabaj estud(Decreto 91)	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
06	Estudiante a Tiempo Completo	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Otras Causas
07	Misión Internacionalista	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
08	Asistencia Técnica	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
09	Donación de Sangre o Examen Médico	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
10	Prestación de Servicios(educ, pcc,ujc,etc)	Autorizada	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	No	No	Autorizaciones Administrativas
11	Prestación de Servicios	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Autorizaciones Administrativas
12	Vacaciones	Vacaciones	Nada	0.00	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes	No	Obligaciones Sociales y Estatales
13	Movilización FAR	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
14	Movilización Comision Electoral	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	No	Yes	No	Obligaciones Sociales y Estatales
15	Servicio Militar General	Autorizada	Nada	0.00	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	Yes	No	Obligaciones Sociales y Estatales
16	Integracion Organ. Estatales	Ninguna	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Obligaciones Sociales y Estatales
18	Conyuge Acompañante	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Autorizaciones Administrativas
19	Descuento Trabaj. Estud. (Decreto 91)	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	No	Autorizaciones Administrativas
20	Semana Receso Docente(Res 250/09)	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Obligaciones Sociales y Estatales
21	Ausencia Justificada	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Autorizaciones Administrativas
22	Ausencia Injustificada	Injustificada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Injustificables
23	Lic. por Maternidad	Licencia de Maternidad	Nada	0.00	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes	No	Licencia de Maternidad Retribuida
24	Consulta Niño Sano	Autorizada	Subsidio sin Reporte	100.00	Yes	No	No	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales

20/05/2019

Page 1 of 2

Anexo 3. Catálogo de claves de ausencia ordenado por código. Página 2. Fuente: Dirección de capital humano.



Información Ordinaria

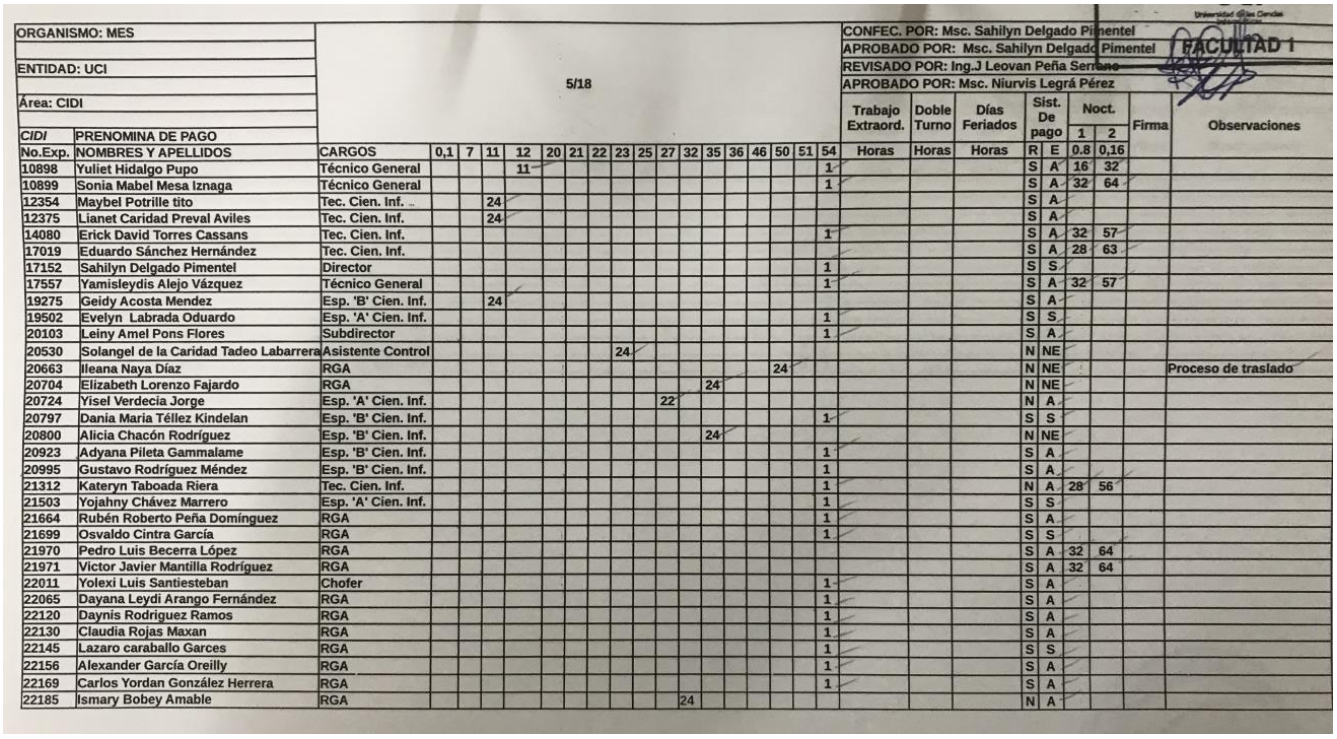
Catálogo de Claves de Ausencias Ordenado por Código

						Afectaciones										
Código	Descripción	Clasificación	A Pagar por	%	Deducible	Vacac. Días	Vacac. Imp.	SNC-225 Días	SNC-225 Imp.	Sub-sidio	Eval.Tec.	Estímulo	Sum.Rpt. Ausenc.	Prom. Trabaj.	Prenom. Horas	Clasific. Fondo Tiempo
25	Lic. sin Sueldo Pos Natal (Hasta 9 meses)	Licencia sin Sueldo	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Otras Causas
26	Enfermedad 3 días o menos	Enfermedad	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	Enfermedad y Accidente Común
27	Enfermedad más de 3 días	Enfermedad	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	Enfermedad y Accidente Común
28	Accidente del Trabajo	Accidente	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	Accidente de Trabajo
29	Lic. sin Sueldo Trabaj estud (Dec 91)	Licencia sin Sueldo	Nada	0.00	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	Obligaciones Sociales y Estatales
30	Consulta de Embarazada	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Obligaciones Sociales y Estatales
31	Privación de Libertad	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Otras Causas
32	Lic. sin Sueldo (Problemas Personales)	Licencia sin Sueldo	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Otras Causas
33	Lic. sin Sueldo (Hasta 6 mes si hijo -16 años)	Licencia sin Sueldo	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Otras Causas
34	Accidente del Trayecto	Accidente	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	No	Enfermedad y Accidente de Trabajo (incluye equiparados)
35	Prestación Social(Res 234/03)	Autorizada	Nada	0.00	Yes	Yes	Yes	No	Yes	No	No	No	No	Yes	No	Obligaciones Sociales y Estatales
36	Vacaciones Masivas (descontar dias)	Autorizada	Nada	0.00	Yes	Yes	Yes	No	Yes	Yes	No	Yes	No	Yes	No	Autorizaciones Administrativas
38	Interrupción al 60%	Interrupciones	Interrupción	60.00	Yes	No	No	No	No	No	No	No	Yes	Yes	No	Otras Causas
39	Interrupción al 70%	Interrupciones	Interrupción	70.00	Yes	No	No	No	No	No	No	No	Yes	Yes	No	Otras Causas
40	Interrupción al 100%	Interrupciones	Interrupción	100.00	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	No	Obligaciones Sociales y Estatales
41	Suspención Laboral	Interrupciones	Suspensión Laboral	100.00	Yes	Yes	Yes	No	No	No	No	No	Yes	Yes	No	Autorizaciones Administrativas
42	Afectacion Huracan	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	No Consigna Tiempo Perdido
43	Medida Cautelar	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Otras Causas
44	Medida Disciplinaria	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Otras Causas
45	Juez Lego	Ninguna	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Obligaciones Sociales y Estatales
46	Trabajando Fuera Entidad	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Otras Causas
50	No Pagar	Autorizada	Nada	0.00	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Otras Causas
51	Curso de Capacitacion	Ninguna	Nada	0.00	No	No	No	No	No	No	No	No	Yes	Yes	No	Otras Causas
52	Dia Feriado Subsidio	Días Feriados Subsidios	Nada	0.00	No	No	No	No	No	No	No	No	No	No	No	Ninguna
53	Dia Feriado Vacaciones	Días Feriados Vacaciones	Nada	0.00	No	No	No	No	No	No	No	No	No	No	No	Ninguna
54	Dia Feriado Rpt Ausencias	Días Feriados Reporte Ausencias	Nada	0.00	No	No	No	No	No	No	No	No	No	No	No	Ninguna
55	Afectación por transportación	Autorizada	Nada	0.00	No	No	No	No	No	No	No	No	Yes	No	No	Autorizaciones Administrativas

20/05/2019

Page 2 of 2

Anexo 4. Modelo de Prenómina en el Centro de IdeoInformática. Fuente: Dirección de Capital Humano.



ORGANISMO: MES

ENTIDAD: UCI

Área: CIDI

5/18

CONFEC. POR: Msc. Sahilyn Delgado Pimentel
APROBADO POR: Msc. Sahilyn Delgado Pimentel
REVISADO POR: Ing.J Leovan Peña Serrano
APROBADO POR: Msc. Niurvis Legrá Pérez

FACULTAD 1

CIDI	PRENOMINA DE PAGO																	Trabajo Extraord.	Doble Turno	Días Feriados	Sist. De pago		Noct.		Firma	Observaciones
No.Exp.	NOMBRES Y APELLIDOS	CARGOS	0,1	7	11	12	20	21	22	23	25	27	32	35	36	46	50	51	54	Horas	Horas	Horas	R	E	1	2		
10898	Yuliet Hidalgo Pupo	Técnico General				11													1				S	A	16	32		
10899	Sonia Mabel Mesa Iznaga	Técnico General																	1				S	A	32	64		
12354	Maybel Potrille tito	Tec. Cien. Inf.			24																		S	A				
12375	Lianet Caridad Preval Aviles	Tec. Cien. Inf.			24																		S	A				
14080	Erick David Torres Cassans	Tec. Cien. Inf.																	1				S	A	32	57		
17019	Eduardo Sánchez Hernández	Tec. Cien. Inf.																					S	A	28	63		
17152	Sahilyn Delgado Pimentel	Director																	1				S	S				
17557	Yamisleydis Alejo Vázquez	Técnico General																	1				S	A	32	57		
19275	Geidy Acosta Mendez	Esp. 'B' Cien. Inf.			24																		S	A				
19502	Evelyn Labrada Oduardo	Esp. 'A' Cien. Inf.																	1				S	S				
20103	Leiny Amel Pons Flores	Subdirector																	1				S	A				
20530	Solangel de la Caridad Tadeo Labarrera	Asistente Control								24													N	NE				
20663	Ileana Naya Díaz	RGA															24						N	NE				Proceso de traslado
20704	Elizabeth Lorenzo Fajardo	RGA												24									N	NE				
20724	Yisel Verdecia Jorge	Esp. 'A' Cien. Inf.										22											N	A				
20797	Dania Maria Téllez Kindelan	Esp. 'B' Cien. Inf.																	1				S	S				
20800	Alicia Chacón Rodríguez	Esp. 'B' Cien. Inf.												24									N	NE				
20923	Adyana Pileta Gammalame	Esp. 'B' Cien. Inf.																	1				S	A				
20995	Gustavo Rodríguez Méndez	Esp. 'B' Cien. Inf.																	1				S	A				
21312	Kateryn Taboada Riera	Tec. Cien. Inf.																	1				N	A	28	56		
21503	Yojahny Chávez Marrero	Esp. 'A' Cien. Inf.																	1				S	S				
21664	Rubén Roberto Peña Domínguez	RGA																	1				S	A				
21699	Osvaldo Cintra García	RGA																	1				S	S				
21970	Pedro Luis Becerra López	RGA																					S	A	32	64		
21971	Víctor Javier Mantilla Rodríguez	RGA																					S	A	32	64		
22011	Yolexi Luis Santiesteban	Chofer																	1				S	A				
22065	Dayana Leydi Arango Fernández	RGA																	1				S	A				
22120	Daynis Rodriguez Ramos	RGA																	1				S	A				
22130	Claudia Rojas Maxan	RGA																	1				S	A				
22145	Lazaro caraballo Garces	RGA																	1				S	S				
22156	Alexander García Oreilly	RGA																	1				S	A				
22169	Carlos Yordan González Herrera	RGA																	1				S	A				
22185	Ismary Bobey Amable	RGA											24										N	A				

Anexo 5 Carta de aceptación por el cliente. Fuente: Dirección de CIDI

Universidad de las Ciencias Informáticas

Acta de aceptación

Acta de Aceptación

En cumplimiento a lo pactado en el levantamiento de requisitos y en función de la ejecución del proyecto: **Sistema para la gestión de la prenómina en el Centro de Ideoinformática**,y quedando el cliente totalmente conforme con la solución desarrollada, se hace entrega de los productos que se relacionan a continuación:

- **Especificación de requisitos.**
- **Sistema desarrollado.**

Entrega

Nombre y apellidos: Luis Ernesto Fernández Buchillón

Cargo: Desarrollador

Recibe

Nombre y Apellidos: Sahilyn Delgado Pimentel

Cargo: Directora del Centro CIDI

UCI Universidad de las Ciencias Informáticas — Centro de Ideoinformática

Fecha: 05/ 06 / 2019