Universidad de las Ciencias Informáticas

Facultad 1

"Módulos de edición de plantillas y recepción de órdenes de impresión para el Sistema de Personalización de Documentos de Identidad basado en tecnologías libres"

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

Autores:

Claudia Peralta González

Dennis Durán Arzuaga

Tutores:

Ing. Dannier Sierra Obregón

MSc. Damaris Cruz Amarán

"No hacen falta alas para hacer un sueño, basta con las manos, basta con el pecho, basta con las piernas y con el empeño..."

Silvio Rodríguez

Declaración de Autoría

Declaramos que somos los únicos autores de este trabajo de diploma y conferimos a la Universidad de las Ciencias Informáticas (UCI) los derechos patrimoniales del mismo, con carácter exclusivo.
Para que así conste firmamos los presentes a los ___ días del mes de ____ del año ____.

Claudia Peralta González

Autor

Dennis Durán Arzuaga

Autor

Ing. Dannier Sierra Obregón

Tutor

MSc. Damaris Cruz Amarán

Tutor

Agradecimientos

Claudia

A mi mamá por acompañarme en mis tristezas y alegrías, por apoyarme cuando se tornaba imposible hacerlo, por todo el cariño que me ha brindado y por ser incondicional en todo momento.

A toda mi familia, en especial a mi abuela Lucy por todo el amor y educación que me ha dado.

A mi padre por su apoyo.

A todos mis compañeros de aula, por aceptarme como soy y quererme durante estos cinco años.

A todos mis amigos, por apoyarme en las buenas y en las malas, por ser mi familia en la universidad.

A la persona que me enseño que no hay meta difícil de cumplir, que siempre podemos dar más y superarnos sin importar los obstáculos, por su amor y comprensión.

A mis tutores por ayudarme y guiarme durante el desarrollo de la tesis y contribuir a la materialización de este trabajo.

A mi compañero de tesis por su ayuda y dedicación, por ser siempre tan alegre y hacerme reír en todo momento, por hacer que este trabajo fuera menos difícil de lo esperado.

A todos realmente muchas gracias.

Dennis

A mi mamá y mi abuela por todo el esfuerzo, la dedicación y el amor incondicional con el que me han guiado.

A mi papá y su esposa, por apoyarme en todo momento y siempre haber estado ahí para mí.

A todos mis hermanos por su dedicación, cariño y apoyo. En especial a mi hermanito Guicho y a mi hermanita Irsa, aunque hoy no puedan estar aquí.

A mi novia y su familia por sus consejos, su compañía, su dedicación y por quererme tanto como me quieren.

A mi compañera de tesis por ser tan preocupada y esforzarse tanto en la realización de este trabajo. Gracias por todo tu cariño y simpatía.

A mis tutores por todo el apoyo que nos brindaron, por la confianza que depositaron en nosotros, por dedicarnos su tiempo y paciencia.

A mis compañeros que están conmigo desde 1er año, a los muchachos del apartamento que siempre me ayudaron en todo lo posible, Raúl, José, Yoandri, Dayrol y el Dany.

A todas las personas que hicieron posible que este sueño se hiciera realidad, sinceramente muchas gracias.

Resumen

En el Centro de Identificación y Seguridad Digital (CISED) de la Universidad de las Ciencias Informáticas (UCI) se desarrolló un Sistema de Personalización de Documentos de Identidad (SPDI), el cual se enfoca en la personalización centralizada de credenciales y documentos de identificación. Este sistema presenta varias limitaciones que obstaculizan su uso a saber: no es multiplataforma, el editor de plantillas que posee no permite añadir al diseño figuras geométricas complejas, imágenes y capas, además la recepción de órdenes de impresión se realiza a partir de plantillas ya implementadas, por lo que al sistema tendrían que hacerle modificaciones en el código fuente de los servicios para poder imprimir un nuevo tipo de documento. Esta investigación persigue como objetivo desarrollar, sobre tecnologías libres, los módulos de edición de plantillas y recepción de órdenes de impresión del Sistema de Personalización de Documentos de Identidad para mejorar su calidad y uso en varias plataformas. El aporte práctico de la investigación radica, en la implementación de una solución multiplataforma que facilite añadir capas, figuras geométricas complejas, gráficos e imágenes vectoriales a las plantillas de los documentos de identificación y que el servicio de recepción permita recibir cualquier tipo de solicitud y procesarla con solo definir un tipo de documento en la administración sin realizar modificaciones en el código de los servicios.

Palabras clave: multiplataforma, personalización, software libre.

Índice de Contenido

Índice de Figuras

Índice de Tablas

Introducción

La constante evolución de las tecnologías, unida a la interconexión y globalización de la economía, ha convertido a las Tecnologías de la Información y las Comunicaciones (TIC) en un elemento estratégico para el crecimiento, maduración y transformación de las organizaciones. Lo anterior proporciona un perfeccionamiento de los productos de *software,* que hacen posible la informatización de procesos.

Uno de los procesos que necesita continuo perfeccionamiento es la emisión de documentos de identificación. Las medidas de seguridad incluidas en los documentos de identificación (hologramas asociados al laminado, impresiones del pulgar, códigos de barras, datos biométricos) nunca son suficientes, es necesario garantizar la confiabilidad de estos documentos con un perfeccionamiento que vaya a la par de las nuevas tecnologías, seguridad y durabilidad.

El proceso de emisión consta de varios subprocesos: enrolamiento, personalización y entrega. El subproceso de personalización se encarga de incorporar al documento de identidad el retrato, la firma y los datos personales, así como los elementos de seguridad para la protección de los datos que fueron captados en el subproceso de enrolamiento (1).

Actualmente existen múltiples sistemas que realizan la personalización de documentos de identidad y brindan una solución a las empresas que demandan una adecuada personalización. En su mayoría estos sistemas están desarrollados con herramientas y tecnologías privativas, los cuales a nivel de mercado poseen un alto precio de uso, limitando su adquisición por países de bajos recursos económicos. Los sistemas que están desarrollados sobre tecnologías libres tienen como limitante que son software a la medida para resolver un problema determinado lo cual hace muy difícil su reutilización.

Teniendo en cuenta lo anteriormente expuesto, muchos países a nivel mundial se han propuesto alcanzar la soberanía tecnológica, a partir de desarrollar sus propias soluciones, apoyándose en el auge que hoy tiene el uso del *software* libre. Motivados por el libre uso, la modificación, la adaptabilidad, así como la redistribución que este posibilita, se han trazado planes concretos para realizar un proceso de migración hacia *software* libre, como la alternativa de futuro con la cual cada país pueda adquirir la soberanía tecnológica deseada.

Cuba es uno de estos países interesados en la soberanía tecnológica*,* viéndolo como una necesidad en los aspectos políticos, económicos, sociales y tecnológicos. Por lo que, el país decide sumergirse en un

proceso de migración a *software* libre, que tuvo sus inicios en marzo de 2006, contando con el apoyo de varias instituciones nacionales, tal es el caso de la Universidad de las Ciencias Informáticas (UCI).

Actualmente la UCI constituye un eslabón fundamental en el proceso de migración que se realiza en el país, siendo esta una de las principales instituciones que promueve el proceso en empresas y universidades del país. Además esta entidad también está inmersa en el proceso de migración al *software* libre; por lo que se han trazado varias estrategias para un buen desarrollo del mismo, tomando como base la resolución rectoral (2) que regula la forma en que se ejecutará la migración.

Dentro de la UCI el Centro de Identificación y Seguridad Digital (CISED) es quien se encarga de desarrollar soluciones integrales, productos y servicios en el campo de la identificación y la seguridad digital. Este centro cuenta con un Sistema de Personalización de Documentos de Identidad (SPDI), el cual se enfoca en la personalización centralizada de credenciales y documentos de identificación.

El SPDI en estos momentos presenta varias limitaciones que obstaculizan su uso a saber:

- ✓ Al estar desarrollado sobre tecnologías privativas y no ser multiplataforma depende de servidores sobre *Windows* para la explotación, reduciendo así el mercado nacional del mismo, pues las empresas del país se encuentran inmersas en el proceso de migración hacia *software* libre.
- ✓ El editor de plantillas que posee no permite añadir al diseño figuras geométricas complejas, imágenes y capas; lo que implica limitaciones en el diseño y que los documentos sean más propensos a falsificar, al tener elementos más simples en el diseño.
- ✓ La recepción de órdenes de impresión se realiza a partir de plantillas ya implementadas, por lo que al sistema tendrían que hacerle modificaciones en el código fuente de los servicios para poder imprimir un nuevo tipo de documento, lo que trae consigo demoras en el proceso de impresión de documentos y costos adicionales de mantenimiento.

El análisis anterior permite definir como **problema de investigación**: ¿Cómo mejorar la calidad y el uso multiplataforma del Sistema de Personalización de Documentos de Identidad para la recepción de órdenes de impresión y edición de plantillas?

Ante el problema definido, se determina como **objeto de estudio** de la investigación el proceso de personalización de documentos de identidad.

El **objetivo general** de la investigación es desarrollar, sobre tecnologías libres, los módulos de edición de plantillas y recepción de órdenes de impresión del Sistema de Personalización de Documentos de Identidad para mejorar su calidad y uso multiplataforma.

Como **objetivos específicos** se identificaron los siguientes:

- ✓ Elaborar el marco teórico de la investigación.
- ✓ Diseñar los módulos de edición de plantillas y recepción de órdenes de impresión.
- ✓ Implementar los módulos de edición de plantillas y recepción de órdenes de impresión.
- ✓ Validar el correcto funcionamiento de los módulos de edición de plantillas y recepción de órdenes de impresión.

Idea a defender: El desarrollo de los módulos de edición de plantillas y recepción de órdenes de impresión sobre tecnologías libres, permitirá mejorar la calidad del Sistema de Personalización de Documentos de Identidad y su uso en varias plataformas.

Para cumplir con los objetivos específicos planteados se proponen las siguientes **tareas de la investigación:**

- ✓ Elaboración del diseño teórico de la investigación.
- ✓ Análisis de los sistemas similares.
- ✓ Definición de la metodología, herramientas, tecnologías y lenguajes a utilizar para el desarrollo del sistema.
- ✓ Definición de los módulos.
- ✓ Análisis y diseño de los módulos definidos.
- ✓ Definición de la arquitectura y la estructura operacional de los módulos.
- ✓ Definición de los patrones de diseño a utilizar.
- ✓ Confección del modelo de datos para describir la estructura lógica de la información persistente manejada por los módulos definidos.
- ✓ Definición de los estándares de codificación de los módulos para mantener la uniformidad en el código.
- ✓ Desarrollo de los módulos definidos.
- ✓ Diseño de los casos de pruebas de los módulos desarrollados para verificar los requisitos funcionales del sistema.
- ✓ Realización de pruebas a los módulos desarrollados.

Para el desarrollo de la investigación se utilizaron los siguientes **métodos científicos**:

Métodos teóricos:

El **Analítico-Sintético** se utilizó para analizar libros, artículos y otras fuentes en la búsqueda de sistemas similares al SPDI existentes en el mundo.

La **Modelación** se utilizó para representar a través de diagramas, la lógica de negocio y las clases que conforman la solución informática.

Métodos empíricos:

La **Entrevista** se empleó para dialogar con especialistas del CISED con el objetivo de obtener información acerca del proceso de personalización de documentos de identidad y conocer los problemas que presenta el actual SPDI. Ver Anexo 1.

La realización de la investigación permitió como resultados:

- ✓ Los módulos de edición de plantillas, recepción de órdenes de impresión y administración correspondientes al nuevo Sistema de Personalización de Documentos de Identidad desarrollado sobre tecnologías libres.
- ✓ Un manual técnico actualizado para la herramienta.

El documento se encuentra estructurado de la siguiente manera:

Capítulo I: "Fundamentación teórica". Se aborda el análisis del estado del arte del tema tratado, haciendo referencia a elementos teóricos de la investigación tales como *software* libre, personalización de documentos, multiplataforma, metodología, lenguajes y herramientas de desarrollo que se utilizan para implementar la solución.

Capítulo II: "Análisis y diseño de la propuesta de solución". Se propone la solución al problema de investigación. Se define la arquitectura del sistema, los patrones de diseño a emplear y se diseñan las tarjetas Clase-Responsabilidades-Colaboraciones (CRC). Además se detallan las historias de usuario (HU).

Capítulo III: "Implementación y prueba de la propuesta de solución". Se definen los estándares de codificación a utilizar y se realiza el diagrama de componente y diagrama de despliegue de la solución. Además se realizan pruebas para verificar la calidad del sistema de personalización, se muestran los resultados y las interfaces principales.

1 Capítulo I: Fundamentación teórica

1.1 Introducción

En este capítulo se abordan aspectos que fundamentan teóricamente y describen el dominio de esta investigación. Se realiza un análisis del estado del arte de los sistemas de personalización de documentos, haciendo referencia a elementos teóricos de la investigación tales como *software* libre y personalización de documentos. De las principales metodologías, lenguajes y herramientas, se analizan sus características, ventajas y desventajas, en la búsqueda de las más indicadas para la construcción de la solución.

1.2 Conceptos asociados al dominio de la investigación

Software libre

El *software* libre se basa en la cooperación, la transparencia y garantiza una serie de libertades a los usuarios (3). Estas libertades son (4):

- ✓ La libertad de usar el programa, con cualquier propósito.
- ✓ La libertad de estudiar el funcionamiento del programa, y adaptarlo a sus necesidades.
- ✓ La libertad de distribuir copias para ayudar a los demás.
- ✓ La libertad de mejorar el programa y de publicar las mejoras, de modo que toda comunidad se beneficie. El acceso al código fuente es un prerrequisito para esto**.**

Multiplataforma

Se dice que un elemento (programa, sistema, aplicaciones) es multiplataforma cuando tiene la capacidad de funcionar en más de una plataforma (combinación de *hardware* y *software* usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos) (5).

Personalización de documentos de identificación

Actualmente existen varias empresas que se centran en la personalización de documentos de identidad, cada una tiene una concepción de lo que es la personalización. A continuación se realiza un análisis de diferentes definiciones de personalización de documentos.

Según la empresa *SAFRAN Morpho* los documentos de identidad son únicos y van dirigidos a una persona o a una entidad. El proceso de **personalización de documentos** es responsable de mucho más que simplemente crear el documento físico, incluye la recepción, procesamiento de los datos e inclusión de funciones de laminado de seguridad para proteger los datos, además engloba la impresión de los documentos. Cualquier fallo en el proceso de personalización puede comprometer tanto la integridad del documento como la reputación de la entidad emisora (1).

Según la empresa *Secured Boundaries Expansion* la **personalización de documentos** es un proceso encargado de la producción de los documentos deseados: tarjetas de identificación, licencias de conducción, pasaportes, visas u otros. Este proceso funciona como un módulo independiente, que recibe solicitudes para imprimir y devuelve como resultado: si la impresión se realizó con éxito o si se deniega. La personalización de documentos incluye 5 pasos principales (6):

- ✓ Recepción de solicitud de impresión y la comprobación.
- ✓ Personalización gráfica.
- ✓ Personalización electrónica.
- ✓ Control de calidad.
- ✓ Exportación de los resultados de impresión.

Los autores de esta investigación después de haber analizado varias bibliografías acerca de la personalización de documentos, arriban a la siguiente conclusión:

La **personalización de documentos** es el proceso que describe el flujo de trabajo para recibir, procesar, imprimir y asociar los datos que identifican a una persona con un documento emitido por una entidad. Por medio de este proceso se incorporan al documento el retrato, la firma y los datos personales, así como los elementos de seguridad para la protección de los datos. Existen varios tipos de personalización entre ellos se encuentran: gráfica[1], magnética[2] y eléctrica[3].

[1] Impresión realizada con tecnología de chorro de tinta, láser y matriz de puntos.

El proceso de personalización de documentos puede ser centralizado o descentralizado. El proceso de personalización centralizado implica el envío de una solicitud al centro donde se realizará la personalización y el proceso de personalización descentralizado se realiza en más de un sitio.

Elementos de seguridad de los documentos de identificación

En el proceso de personalización de documentos de identidad se le añaden al documento elementos de seguridad con el objetivo de evitar situaciones de fraudes, suplantación de la identidad y obtener un producto más seguro. Dentro de los elementos de seguridad se encuentran: elementos de lectura mecánica y elementos de seguridad física.

Los **elementos de lectura mecánica** son el código de barra unidimensional (son barras y espacios organizados en una línea que guardan información, ejemplos: UPC, EAN 39, *Interleaved 2 of 5*, *Code 128*) y el código de barra bidimensional. Este último es de alta capacidad y puede sostener cantidades significantes de datos en un solo símbolo, además contienen los datos en ambas direcciones: horizontal y vertical, por eso se llama bidimensional. Los más utilizados son: PDF417, *Datamatrix*, *Maxicode* y *QR Code.*

Los **elementos de seguridad física** son visibles a simple vista (tintas ópticamente variables, relieves, fondos de seguridad) y verificables mediante medios ópticos y electrónicos (tintas visibles con luz ultravioleta, microescrituras).

1.3 Sistemas similares

En la actualidad existen diversas soluciones informáticas que se dedican a la impresión y personalización de documentos de identidad con un alto grado de seguridad, a continuación se realiza un análisis a nivel nacional e internacional de algunas de ellas.

1.3.1 Nivel nacional

[2] Impresión realizada sobre una banda magnética de alta o baja coercitividad.

[3] Impresión realizada en un circuito electrónico sobre una lámina de policarbonato u otro material similar.

Empresa DATYS, sistema EMIMAR.

Es un sistema para la emisión del carné de identificación del marino, que cumple con las normas internacionales de la Organización de Aviación Civil Internacional (OACI) para documentos de lectura mecánica, que trabaja con estándares de seguridad y calidad internacionales, en un ambiente amigable y flexible. El sistema admite inclusión de código de barras bidimensional para datos biométricos y alfanuméricos, puede trabajar de manera independiente o integrada a un sistema de identidad, incluye mecanismos de seguridad y auditoría de la información, herramientas de detección y eliminación de duplicados, autenticación de los usuarios por contraseña y huella, procesos de transmisión seguros y auto-controlados, con verificación de la autenticidad de la información (7).

El sistema EMIMAR permite la captura y almacenamiento de información biométrica, además puede realizar verificaciones de calidad de imágenes según las normas internacionales de la OACI y ejecuta un control de los inventarios de las libretas del carné de verificación de la producción de los documentos. Sin embargo EMIMAR solamente se dedica a la personalización de un tipo de documento específico y el código fuente no está disponible públicamente.

SPDI

Es un sistema de personalización de documentos de identidad, que tiene como objetivo automatizar el proceso de personalización de documentos con los adecuados controles sobre los insumos, también permite imprimir diferentes tipos de documentos. El SPDI permite diseñar y configurar las plantillas de impresión y tipos de documentos, el balance de carga para los puestos de trabajo y la creación de reportes. La arquitectura de este sistema está conformada por cinco capas: Presentación, Procesos y Servicios, Negocios, Acceso a datos y Base de datos. El sistema puede imprimir códigos de barras bidimensionales.

El SPDI no presenta un editor de plantillas que permita añadir al diseño figuras geométricas complejas y para poder imprimir un nuevo tipo de documento tienen que hacerle modificaciones en el código fuente de los servicios, implicando demoras en el proceso de impresión de documentos. Además no es multiplataforma. Actualmente el SPDI está integrado a la Plataforma Modular de Identificación y Control de Acceso del CISED.

1.3.2 Nivel internacional

Morpho Perso

Sistema de personalización basado en la impresión hecha en diferentes tipos de material de tarjetas como: policarbonato, papel, PVC[4], PET-F[5]. También utiliza láser y la impresión sobre una amplia variedad de chips electrónicos, además incluye código QR[6] para enlazar directamente con el sitio web del registro civil para verificar si la cédula está vigente, no está bloqueada, o si fue emitida válidamente (8).

El sistema *Morpho Perso* de la Empresa SAFRAN Morpho está presente en Europa, México, Colombia, Chile entre otros países. Sus procedimientos de control y calidad (gráfica y chip) pueden ser adaptados con el fin de controlar el 100% de los documentos personalizados o para proceder en el modo de muestra, para ambos casos satisface los estándares de calidad. Este sistema no es de código abierto, por lo que el código fuente no puede ser reutilizado por otras instituciones.

Sistema de seguridad 3M

Sistema de personalización basado en el grabado láser en tarjetas de policarbonato, utilizando la impresión láser para tarjetas con sustrato basado en *Teslin*®[7]. Se enfoca en la impresión de varios tipos de documentos de identidad. Ofrece soluciones de laminado de seguridad, que también pueden utilizarse para asegurar los datos de personalización impresos e incluye tecnología de transferencia térmica en color para la personalización de tarjetas compuestas (9).

Este sistema de personalización cuenta con materiales de seguridad patentados con tecnologías visibles, ocultas y forenses para el laminado de seguridad, que podrán formar parte de una credencial de identificación segura. Además utiliza diferentes tipos de laminados de seguridad como son: laminado

[4] Tarjetas plásticas de policloruro de vinilo.

[5] Tarjetas plásticas de politereftalato de etileno.

[6] Código de respuesta rápida.

[7] Medio de impresión sintético impermeable.

holográfico, laminados holográficos transparente y laminado de seguridad con imagen flotante en color 3M[8]. El sistema de seguridad 3M no es de código abierto.

Asure ID Exchange

El sistema se basa en potentes módulos de integración, inscripción y codificación que garantizan una vía de acceso rápida y constante a cualquier otra fuente de base de datos existente, ya esté instalada en un computador o en una tableta. También incluye un módulo de configuración láser, *iCLASS SE*®[9] servicio de ayuda a la plataforma abierta de codificación y se integra con la impresora de grabado con láser *FARGO® LE*[10] permitiendo imprimir, codificar y laminar las credenciales de seguridad con una única plantilla de tarjetas integrada; proporcionando ahorrar tiempo valioso de producción y entrega (10). Exchange es perfecto para empresas que buscan un diseño intuitivo de tarjetas y herramientas de gestión de datos para aplicaciones avanzadas de personalización de credenciales. Uno de los principales beneficios de este sistema es que permite añadir códigos de barras lineales en 2D (PDF417, QR *Code* y *Datastripe*) o una banda magnética. Este sistema solamente se dedica a la personalización de un tipo de documento específico y el código fuente no está disponible públicamente.

Comparación de los sistemas similares

Sistema	Sistema de impresión	Soporte multi-documento	Lectura mecánica
Morpho Perso	Láser y chips electrónicos.	Sí	Código QR
Asure ID Exchange	Láser	No	Código QR, PDF 417 y *Datastripe*.
3M	Láser y holográfico	Sí	Imagen flotante, transferencia térmica

[8] *Minnesota Mining and Manufacturing Company.*

[9] Lectores de control de acceso de HID Global.

[10] Tipo de láser para credenciales.

			de color y laminado holográfico transparente.
EMIMAR	Láser	No	Código 2D, datos biométricos y alfanuméricos.
SPDI	Configurable a través del DGM[11].	Sí	Códigos (QR y barra).

Tabla 1. Comparación de los sistemas similares. Elaboración propia.

El análisis realizado sobre los sistemas similares tanto en el ámbito nacional como internacional, arrojó las siguientes conclusiones:

- ✓ Los sistemas de personalización nacionales analizados, incluyen las medidas de seguridad para la confección e impresión de documentos de identificación. Sin embargo, el editor de plantillas del SPDI no permite agregar al diseño figuras geométricas complejas, imposibilitando un mejor diseño y calidad de la impresión en estos documentos al incorporar capas e imágenes vectoriales. Además el código fuente del sistema EMIMAR no está disponible públicamente, lo que impide su reutilización.
- ✓ Los sistemas internacionales analizados no son de código abierto, lo que imposibilita la obtención y modificación del código fuente para ser utilizado por cualquier institución.
- ✓ El análisis de los sistemas permitió identificar que estos evidencian la necesidad de incrementar el uso de medidas de seguridad en los documentos de identificación, teniendo en cuenta que muchas de estas medidas se incorporan en el proceso de personalización de documentos.

1.4 Metodologías de desarrollo de *software*

Las metodologías ágiles se caracterizan por responder rápido a los cambios, por ajustarse a diseños simples e incrementos cortos, por facilitar una comunicación interpersonal e intuitiva con el cliente, por ser apropiadas para proyectos de poca duración y con equipos de desarrollo pequeños. Teniendo en cuenta

[11] DGM del inglés *Device Grid Manager*

lo anteriormente expresado y que el CISED utiliza este tipo de metodologías y además de ser el cliente de la solución a desarrollar en la investigación, se decide hacer un estudio entre metodologías ágiles.

1.4.1 Programación extrema

La metodología Programación Extrema (por sus siglas en inglés XP) guía el desarrollo de *software* haciendo énfasis en las relaciones interpersonales, fomentando el trabajo en equipo y la estrecha comunicación con el cliente. El producto de *software* se construye teniendo en cuenta que la solución más simple es la mejor. Está orientada a la adaptación paulatina de los requisitos y de sus cambios en cualquier punto de la vida del proyecto. Define historias de usuario para describir las funciones del sistema, las cuales son escritas por el cliente. El ciclo de desarrollo en XP es iterativo e incremental y la propia metodología está regida por varias fases (11), las cuales se muestran en la siguiente figura.

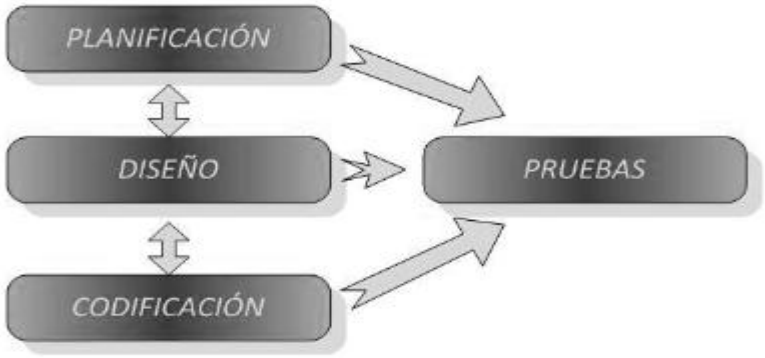

Figura 1. Fases de la metodología XP. Tomado de **(12)**.

Esta metodología aplica un conjunto de prácticas que hacen la entrega del componente menos complicada y más satisfactoria tanto para los clientes como para el equipo de entrega. Entre las prácticas más importantes se pueden mencionar la codificación por parejas posibilitando que el código sea discutido y revisado mientras se escribe. Pequeñas entregas del sistema en forma de versiones operativas aunque no incluyan todas las funcionalidades. Diseño simple y dirección de las pruebas unitarias en la codificación. Además permite la refactorización del código dirigida a simplificarlo para facilitar sus cambios en el futuro, presencia continua del cliente en la producción y utilización de estándares de programación, entre otras (13).

1.4.2 Desarrollo guiado por la funcionalidad

La metodología Desarrollo Guiado por la Funcionalidad (por sus siglas en inglés FDD) es un proceso iterativo, con iteraciones cortas (2 semanas), las cuales producen un *software* funcional que los clientes pueden ver y monitorizar. Ha sido pensado para proyectos con un tiempo de desarrollo menor de dos años. Las iteraciones se deciden teniendo en cuenta las funcionalidades, que representan fragmentos del *software* con significado para el cliente. FDD consta de 5 fases: desarrollo de un modelo global, construcción de la lista de funcionalidades, planificación por funcionalidad, diseño por funcionalidad e implementación por funcionalidad.

El trabajo de modelado y de desarrollo se realiza en grupo. Las funcionalidades a implementar en una entrega son repartidas entre los distintos subgrupos del equipo, las clases de *software* tienen propietario, es decir, sólo el creador puede modificarlas. Es por ello que el equipo que implementa cierta funcionalidad tiene que tener todos los propietarios de las clases implicadas. Implementar una funcionalidad lleva implícito la preparación y ejecución de pruebas, así como revisión del código e integración de las partes que componen el *software*.

1.4.3 Scrum

Scrum es una metodología de desarrollo de *software* iterativa e incremental. Define un marco para la gestión de proyectos orientado a qué construir y en qué orden hacerlo; debido a ello se utiliza en otras prácticas de ingeniería de *software* tales como Proceso Racional Unificado (por sus siglas en inglés RUP) o XP. Es muy sencilla y responde a los principios de inspección constante y continua, adaptación del producto a las necesidades del cliente en tiempo real e innovación (14).

Tiene como meta fomentar la efectividad y productividad de los equipos. Entre sus principales características destacan los ciclos de desarrollo llamados sprints[12] con una duración de 30 días, cuyo resultado es un incremento funcional del producto. Establece reuniones durante todo el proyecto, de ellas las más importantes son las diarias, en las cuales los miembros del equipo disponen de 15 minutos para chequear el trabajo realizado hasta la fecha y las previsiones para el día siguiente (15).

[12] Término utilizado por la metodología para referirse a una iteración del proceso de desarrollo de software.

Selección de la metodología de desarrollo de *software*

Después del análisis de diferentes metodologías de desarrollo teniendo en cuenta sus características y algunas de sus ventajas se decide seleccionar como metodología a seguir XP puesto que:
Tiene como prioridad satisfacer al cliente mediante un desarrollo iterativo e incremental, abierto a los cambios y caracterizado por un código simple. Esta metodología exhorta a la programación en pareja y presenta un conjunto de buenas prácticas, las cuales se muestran a continuación (16):

- ✓ **Versiones pequeñas:** Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no puedan ver funcionando.
- ✓ **Desarrollo guiado por las pruebas automáticas:** Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, más efectivo.
- ✓ **Integración continua:** Debe tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de una vez. Cuando exista alguna falla, no podría establecerse dónde ha sido, sino seguimos la buena práctica de hacerlo en su momento.
- ✓ **El código es de todos:** Cualquiera puede y debe interactuar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- ✓ **Normas de codificación:** Debe haber un estilo común de codificación (no importa cuál), de forma que parezca que ha sido realizado por una única persona.

Además XP se centra también en la generación de una documentación, aunque no exhaustiva, suficiente para soportar la solución, teniendo en cuenta la necesidad de resultados tangibles a corto plazo.

1.5 Tecnologías, herramientas y lenguajes utilizados

1.5.1 Lenguaje de modelado

En el proceso de desarrollo de un *software*, el modelado del mismo es de vital importancia por lo que se tiene como propuesta inicial utilizar: como lenguaje de modelado el Lenguaje Unificado de Modelado (por sus siglas en inglés UML) (17). Es un lenguaje para visualizar, especificar, construir y documentar los

artefactos de un sistema que involucra una gran cantidad de *software*. Se escoge UML porque permite modelar sistemas utilizando técnicas orientadas a objetos. Mediante este lenguaje se pueden especificar todas las decisiones de análisis y diseño, construyéndose así modelos precisos, no ambiguos y completos. Además puede conectarse con lenguajes de programación (ingeniería directa e inversa) y permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones).

1.5.2 Herramientas de modelado

Dentro de las herramientas clave en el desarrollo de aplicaciones informáticas se encuentran las herramientas de Ingeniería de *Software* Asistida por Ordenador (por sus siglas en inglés CASE), las cuales son las encargadas de ayudar en el ciclo de desarrollo, con el fin de aumentar la productividad y reducir el costo en términos de tiempo y dinero (18). Las herramientas CASE se han venido ampliando y desarrollando, algunos de sus ejemplos son: *Microsoft Project*, *Rational Rose*, *JDeveloper*, *Visual Paradigm*, *Microsoft Visio*, entre otros. En la presente investigación se analizaron algunas de ellas:

Rational Rose Enterprise Edition

Es una herramienta CASE desarrollada por *Rational Corporation* basada en UML, que permite crear los diagramas que se generan durante el proceso de ingeniería en el desarrollo del *software*. Los desarrolladores de sistemas pueden usar este producto para producir modelos visuales que sirvan de apoyo y guía para el desarrollo del *software*, centrándose en los casos de uso y enfocándose hacia un producto de mayor calidad, empleando un lenguaje estándar común que facilite la comunicación (19).
Las ventajas de esta herramienta residen en su utilidad en aplicaciones grandes y complejas, reduce el tiempo de desarrollo de manera automática para pasar de un esquema a otro y al código. Además los nombres y los datos se mantienen de manera consistente proporcionando una sincronización para diferentes desarrolladores. Sin embargo tiene como desventajas su enfoque fijo de desarrollo, limitación en la flexibilidad de la documentación y los costos en *software*, manuales y capacitación.

Visual Paradigm

Visual Paradigm es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Permite construir

todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (20).

Tiene la capacidad de crear el esquema de clases a partir de una base de datos y crear la definición de base de datos a partir del esquema de clases. Permite invertir código fuente de programas, archivos ejecutables y binarios en modelos UML al instante, creando de manera simple toda la documentación. Está diseñada para usuarios interesados en sistemas de *software* de gran escala con el uso del acercamiento orientado a objeto. Apoya los estándares más recientes de las notaciones de Java[13] y de UML. Incorpora el soporte para trabajo en equipo, que permite que varios desarrolladores trabajen a la vez en el mismo diagrama y vean en tiempo real los cambios hechos por sus compañeros.

Visual Paradigm cumple con las políticas actuales de migración a *software* libre, siendo multiplataforma, de forma tal que facilita la modelación del *software* independientemente del sistema operativo que se emplee (21).

1.5.3 Lenguajes de programación

Un lenguaje de programación es un idioma artificial diseñado para expresar procesos que pueden ser ejecutados por máquinas computadoras. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina. Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Durante el desarrollo de la investigación se efectuó un análisis de los lenguajes de programación Python y PHP de los cuales se consideraron sus características principales.

Python

Python es un lenguaje de programación con una sintaxis muy limpia y que favorece un código legible. Se trata de un lenguaje interpretado o de script (es aquel que se ejecuta utilizando un programa intermedio llamado intérprete), con tipado dinámico, fuertemente tipado, multiplataforma, orientado a objetos y que además, es de código abierto (22). El lenguaje de programación Python es de alto nivel, aunque permite modificar bits y bytes. Permite dividir el programa en módulos reutilizables desde otros programas. Viene

[13] lenguaje de programación de propósito general, concurrente, orientado a objetos.

con una gran colección de módulos estándar que se pueden utilizar como base de los programas (o como ejemplos para empezar a aprender Python) (23).

Ventajas del uso de Python (24):

- ✓ **Propósito general:** Se pueden crear distintos tipos de programas.
- ✓ **Multiplataforma:** Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible siempre y cuando exista un intérprete programado para este lenguaje.
- ✓ **Interpretado:** No se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador.
- ✓ **Interactivo:** Dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- ✓ **Orientado a objetos:** La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes reutilizables. Además, Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.
- ✓ **Funciones y librerías:** Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de *strings*[14], números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.
- ✓ **Sintaxis clara:** Tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.
- ✓ **Mixto:** Se puede integrar de manera "fácil" con otros lenguajes de programación.
- ✓ **Gratuito:** Una ventaja fundamental de Python es la gratuidad de su intérprete, se puede descargar desde la página web: http://www.python.org.

[14] cadena de caracteres.

PHP

PHP "es un lenguaje interpretado de alto nivel embebido en páginas HTML[15] y ejecutado en el servidor" (25). Se conecta a servidores de bases de datos tales como MySQL, Oracle, PostgreSQL, entre otros. Debido a su amplia distribución, está perfectamente soportado por una gran comunidad de desarrolladores. Es un lenguaje de programación de alto nivel, totalmente compatible con los modernos métodos orientados a objetos, prácticas y principios.

PHP compila para el código una serie de instrucciones siempre que estas son accedidas. La concepción de lenguaje interpretado es que las instrucciones son ejecutadas una por una, hasta que termina el código. Es usado principalmente en interpretación del lado del servidor para la generación de páginas web dinámicas. Las principales características de PHP son: rapidez, facilidad de aprendizaje, soporte multiplataforma tanto de diversos sistemas operativos como de servidores HTTP[16]. Este lenguaje se distribuye de forma gratuita bajo una Licencia Pública General (por sus siglas en inglés GPL) (26).

Selección del lenguaje a utilizar

El lenguaje de programación será Python en su versión 2.7.6 por ser potente y posibilitar la creación de una aplicación robusta, mientras que PHP no es identado, por lo que utiliza llaves en vez de estructura en bloques y no es un lenguaje muy claro, ni conciso. Por otra parte Python tiene eficaces estructuras de datos de alto nivel y una solución de programación orientada a objetos simple pero eficaz. Conjuntamente proporciona un equilibrio muy bueno entre lo práctico y lo conceptual puesto que es un lenguaje interpretado. Con Python, suele ocurrir que se vuelve algo lento en la ejecución de los programas, ya que en lugar de compilar los programas escritos en este lenguaje para obtener ficheros binarios y librerías, se ejecutan sobre la marcha.

[15] *HyperText Markup Language* («lenguaje de marcas de hipertexto»).

[16] *Hypertext Transfer Protocol* («protocolo de transferencia de hipertexto »).

1.5.4 Marcos de trabajo

Un *framework* web, es un marco de trabajo diseñado para apoyar el desarrollo de sitios web dinámicos. Ofrece un conjunto de componentes para acelerar el proceso de desarrollo, reutilizar código ya existente y promover buenas prácticas de desarrollo como el uso de patrones (27).

Django

Django promueve el desarrollo rápido, se construyen aplicaciones en cuestión de días y con el conocimiento suficiente esos días se pueden reducir a horas. Django impulsa el desarrollo de código limpio al promover buenas prácticas de desarrollo web, sigue el principio *Don't Repeat Yourself* (DRY) (conocido también como una vez y sólo una). Usa una modificación de la arquitectura Modelo-Vista-Controlador (MVC), llamada *Model-Template-View* (MTV), que sería Modelo-Plantilla-Vista, esta forma de trabajar permite que sea pragmático[17] (28).

Django es un marco de trabajo de desarrollo para la web. Su implementación es totalmente sobre Python. Con este marco de trabajo se pueden crear y mantener aplicaciones de alta calidad. Incluye un servidor web ligero que se puede usar mientras se desarrolla. Al mismo tiempo, Django permite trabajar fuera de su ámbito según sea necesario. Django ofrece las siguientes facilidades (29):

- ✓ Sistema de plantillas para separar la presentación de un documento de sus datos.
- ✓ Construcción automática de interfaces de administración.
- ✓ Vistas genéricas que recogen ciertos estilos y patrones comunes en su desarrollo y los abstraen, de modo que se puede escribir rápidamente vistas comunes de datos sin tener que escribir mucho código.
- ✓ Sistema de caché robusto y con un nivel de granularidad ajustable, que permite guardar páginas dinámicas para que no tengan que ser recalculadas cada vez que se piden.
- ✓ Integración con bases de datos y aplicaciones existentes.
- ✓ Construcción de aplicaciones multilenguaje permitiendo especificar cadenas de traducción de más de cuarenta idiomas.

[17] Significa práctico, define sus acciones en la práctica y los resultados que estas conllevan.

Web2py

Web2py es un *framework* enfocado en el desarrollo rápido y sigue un diseño Modelo-Vista-Controlador. Provee una amplia interfaz administrativa basada en web (no hay necesidad de escribir órdenes en la línea de comandos, salvo que así lo deseara). Incluye bibliotecas para manejar protocolos como: XML-RPC[18] y RSS *feeds*[19], entre otros. Este *framework* tiene la habilidad de generar formularios para las tablas de la base de datos e incluye un extenso conjunto de validadores. Web2py se diferencia de Django porque es más compacto, fácil de leer y no tiene ningún archivo de configuración a nivel de proyecto. Además viene en versión de código fuente (para cualquier Sistema Operativo que soporte Python) y en versiones binarias para OSX y *Windows* (30).

JQuery

JQuery es un *framework* JavaScript que permite de una manera ágil crear aplicaciones web, es libre, su código se distribuye bajo la Licencia del Instituto Tecnológico de Massachusetts (por sus siglas en inglés MIT) y la Licencia Pública General de GNU en su versión 3. Cuenta con funcionalidades para acceder al árbol del DOM[20] del documento HTML, posee selectores para elegir los elementos por clases e identificadores, o para seleccionar por tipo de elemento. Permite el manejo de los eventos que generan los periféricos de entrada como el *mouse* o ratón y el teclado. Es multiplataforma y extensible a través de *plugins*[21], permitiendo hacer más amplias sus funcionalidades, o agregar aquellas que no estén en existencia. Su tamaño es realmente pequeño, disminuyendo el ancho de banda o tráfico de red, desde el servidor al dispositivo cliente. Posee una amplia documentación en línea[22].

Las características de jQuery permitirán desarrollar las interfaces de usuario interactivas de manera ágil, simplificando la cantidad de líneas de código para realizar una operación y también aplicaciones que

[18] protocolo de llamada a procedimiento remoto (XML *Remote Procedure Call*).

[19] *Really Simple Syndication*, un formato XML para sindicar o compartir contenido en la web.

[20] Modelo de objeto del documento.

[21] Componente de *software* que permite ampliar las funcionalidades del mismo.

[22] http://docs.jquery.com

hacen uso de *Asynchronous* JavaScript And XML (AJAX) (técnica de desarrollo web para crear aplicaciones interactivas). La posibilidad de desarrollar a través de *plugins* las interfaces de usuario, será útil en el módulo de servicios porque permitirá obtener una aplicación fácil de mantener ante posibles cambios o personalización del sistema (31).

1.5.5 Sistemas gestores de base de datos

Un sistema gestor de base de datos (SGBD) es el *software* que permite a los usuarios procesar, describir, administrar y recuperar los datos almacenados en una base de datos (32). Estos sistemas facilitan el control sobre los datos almacenados y posibilitan garantizar la integridad y confiabilidad de la información a través de las políticas de seguridad que implementan.

PostgreSQL

Se trata de un SGBD que incorpora el modelo relacional para sus bases de datos y es compatible con el lenguaje de consulta *Structured Query Language* (SQL), que significa lenguaje de consulta estructurado. Resulta ser muy capaz y muy confiable, tiene buenas características de rendimiento. Es un sistema multiplataforma por lo que opera en varios sistemas operativos como Unix, Mac OS, *Windows*, Linux. Es de código abierto lo que hace posible que los usuarios puedan realizar las modificaciones pertinentes al código fuente según la necesidad de estos (33).

Entre sus principales características resaltan el uso de procedimientos almacenados, el soporte de integridad referencial y manejo de distintos tipos de datos. Utiliza la tecnología Control de Concurrencia Multi-Versión (por sus siglas en inglés MVCC) el cual permite que se ejecuten sobre una tabla varias transacciones a la vez, pues para cada transacción se muestra una versión de la tabla y no la original (34). Este gestor es robusto y brinda una API[23] flexible para programar en C/C++, Java y Python, entre otros. Se distribuye bajo la Licencia de Distribución de *Software* de *Berkeley* (por sus siglas en inglés BSD[24]), además cuenta con una comunidad de desarrollo que constantemente realiza mejoras al *software*.

[23] Interfaz de Programación de Aplicaciones.

[24] *Berkeley Software Distribution* («distribución de *software berkeley*»). Es una licencia de software libre permisiva como la licencia de OpenSSL o la MIT License.

MySQL

Es un SGBD relacional, rápido, sólido y flexible. Utiliza el lenguaje SQL, es el más usado y estandarizado para acceder a base de datos relacionales. Facilita la integración con programas desarrollados en C y C++ pues fue desarrollado en este lenguaje. Es portable pues está disponible en más de 20 plataformas incluyendo las distribuciones más usadas de Linux, sistema operativo Mac X, UNIX y Microsoft Windows. Es ideal para crear base de datos con acceso desde páginas web dinámicas o para cualquier otra solución profesional que implique almacenar datos, teniendo la posibilidad de realizar múltiples y rápidas consultas (35).

1.5.6 Entornos de Desarrollo Integrados

Los Entornos de Desarrollo Integrados (por sus siglas en inglés IDEs) son herramientas pensadas para escribir, compilar, depurar y ejecutar programas. Esta herramienta puede estar pensada para su utilización con un único lenguaje de programación o bien puede dar cabida a varios de estos (36).

Eclipse

Eclipse es una plataforma que ha sido diseñada desde el principio para la construcción de aplicaciones integrando herramientas de desarrollo. Por su diseño, la plataforma no proporciona una gran cantidad de funcionalidades para el usuario final por sí misma, sin embargo define una arquitectura abierta para que infinitos módulos o *plugins* puedan instalarse permitiendo a los desarrolladores centrarse en su área de especialización. Se considera un IDE que proporciona herramientas que facilitan el desarrollo de aplicaciones, permitiendo compilarlas, correrlas y depurarlas. Provee además herramientas para administrar el espacio de trabajo y control de versiones sobre el código fuente (37).
Este entorno de desarrollo permitirá escribir el código fuente del sistema, al mismo tiempo que consume una mínima cantidad de recursos de *hardware* y *software*.

PyCharm

PyCharm es un IDE basado en *IntelliJ* IDEA[25], por lo que hereda naturalmente sus funciones de edición JavaScript, HTML y CSS[26], además incluye un editor de código inteligente que entiende las capacidades específicas de Python (38). Presenta un completo conjunto de herramientas para el desarrollo web de aplicaciones en dicho lenguaje, ofreciendo compatibilidad con marcos de trabajo modernos como Django. Además ofrece autocompletamiento de código y señalamiento de errores con soluciones fáciles. Posibilita una fácil navegación para los proyectos y el código de estos debido a las vistas estructuradas, que facilitan un rápido movimiento entre archivos, clases y métodos. Mantiene el código bajo control de chequeos, asistencia de pruebas, refactorizaciones y un conjunto de inspecciones que ayudan a codificar de forma limpia y sostenible (39).

1.5.7 Servidor web a utilizar

Un servidor, como su nombre lo indica, brinda un servicio a otros dispositivos, a los cuales se les conoce como clientes. Su principal tarea es alojar sitios y/o aplicaciones, las cuales son accedidas por los clientes utilizando un navegador que se comunica con el servidor usando el protocolo HTTP. Este se mantiene a la espera de peticiones y le responde con el contenido según sea solicitado. Disponen de intérpretes de lenguajes de programación que ejecutan código embebido dentro del HTML de las páginas que contiene el sitio antes de enviar el resultado al cliente (40).

Apache

Es un servidor de red para el protocolo HTTP, elegido para poder funcionar como un proceso independiente, sin que eso solicite el apoyo de otras aplicaciones o directamente del usuario. Apache se distribuye como *software* libre de código abierto. Es un servidor modular, multiplataforma, extensible, popular (fácil de conseguir ayuda/soporte) y gratuito. Su licencia es una descendiente del tipo BSD que permite el uso comercial y no comercial de Apache (40).

[25] IDE de Java por *JetBrains*.

[26] *Cascading Style Sheets* («hojas de estilo en cascada»).

Características:

- ✓ Altamente configurable de diseño modular y en la creación y gestión de *logs*[27].
- ✓ Personaliza la respuesta ante los posibles errores que puedan ocurrir.

Se utiliza este en su versión 2.2.

Elección de las tecnologías, herramientas y lenguajes a utilizar.

- ✓ La herramienta CASE será la versión 8.0 de *Visual Paradigm* para UML, por ser un *software* libre que permite realizar ingeniería tanto directa como inversa y soportar múltiples usuarios trabajando sobre el mismo proyecto, además el equipo de proyecto está familiarizado con su empleo y ya se tiene una experiencia previa positiva.
- ✓ El lenguaje seleccionado se pondrá en práctica haciendo uso de PyCharm en su versión 4.0.4 como IDE, el cual ofrece destacados impulsores de productividad: formateo de código automático, finalización del código y navegación de código.
- ✓ Se decidió utilizar Django en su versión 1.7 como marco de trabajo, pues se centra en el desarrollo rápido, la reutilización y la seguridad. Además de contar con una comunidad de desarrollo *online* amplia.
- ✓ Se decidió utilizar JQuery en su versión 1.10.3 para las validaciones no funcionales en la parte del cliente, por ser un marco de trabajo que facilita la selección de elementos HTML, la creación de animaciones y evita la implementación de funcionalidades comunes.
- ✓ Entre MySQL y PostgreSQL, se decidió que este último era el que con mayor completitud ofrecía un balance entre la integridad, la eficiencia y las funcionalidades para la consulta de los datos. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios sin presentar incoherencias en la escritura de los datos debido al MVCC. Además, es muy tolerante a fallos debido al uso de multiprocesos y no de multihilos. Estas son características únicas de PostgreSQL que lo hacen la opción más idónea para un sistema de tiempo de respuesta rápido. Se propone la utilización de la versión 9.4.1.

[27] es un registro de actividad de un sistema, que generalmente se guarda en un fichero de texto.

1.6 Conclusiones

El estudio de los sistemas y los conceptos teóricos permitió sentar las bases teóricas para la personalización de los documentos de identificación, evidenciando la necesidad de tener en cuenta varios elementos que permiten mejorar la calidad de los documentos, a través de diseños más adaptables y seguros que incorporan medidas de seguridad. El análisis del ambiente de desarrollo permitió establecer la metodología, las herramientas, lenguajes y tecnologías a utilizar en el desarrollo del sistema de acuerdo a las características deseadas en este. Esto permite crear una solución multiplataforma y con el menor costo de construcción posible.

2 Capítulo II: Análisis y diseño de la propuesta de solución

2.1 Introducción

En el presente capítulo se abordan las principales características de la solución propuesta. Se define el modelo de dominio donde se agrupan los principales conceptos a tratar. Además se desarrollan las fases iniciales de la metodología de desarrollo XP: planificación y diseño, presentándose los diferentes artefactos que genera la misma.

2.2 Modelo de dominio

En el modelo de dominio se representan los conceptos más importantes y significativos en el desarrollo de un sistema. Su objetivo fundamental es definir las interrelaciones de los objetos más importantes representados mediante clases. Además, desempeña un papel central en la comprensión del entorno actual y en la planificación futura de la posible aplicación (25).

A continuación se muestra el modelo de dominio definido.

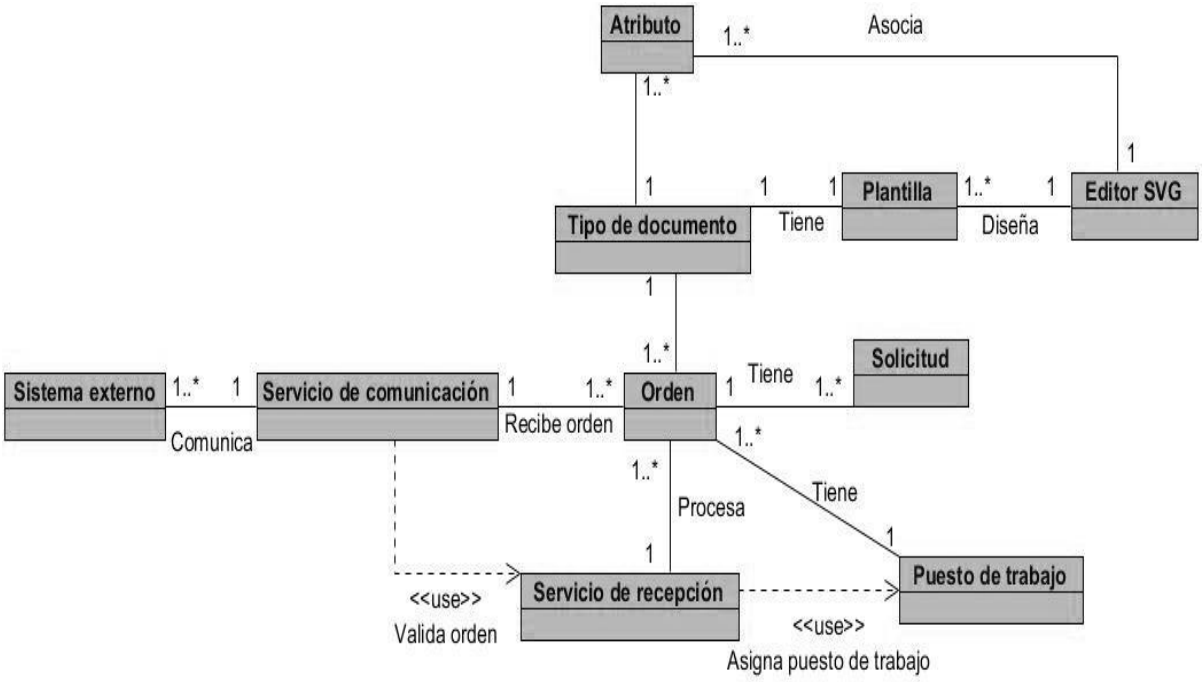

Figura 2. Modelo de dominio. Elaboración propia.

Sistema externo: Envía las órdenes de impresión que se desean imprimir al sistema.

Servicio de comunicación: Recibe las órdenes de impresión y usa el servicio de recepción para validar y procesar las órdenes.

Orden: Contiene el tipo de documento y un conjunto de solicitudes. La orden es asignada a un puesto de trabajo.

Servicio de recepción: Procesa las órdenes de impresión teniendo en cuenta el tipo de documento, atributos y los campos de las solicitudes, devuelve mensajes de aceptación o de error. Si el mensaje es de aceptación se selecciona un puesto de trabajo al que será asignada la orden.

Puesto de trabajo: Son las estaciones de trabajo, donde existe un operario encargado de procesar las órdenes.

Solicitud: Contiene los datos correspondientes a un documento de identificación.

Tipo de documento: Describe los campos esperados que hacen referencia a la orden recibida y la plantilla asociada al mismo.

Atributo: Son los campos que tienen los tipos de documentos y que serán esperados en cada solicitud que pertenece a una orden.

Plantilla: Está asociada a un tipo de documento y contiene un XML que representa la disposición gráfica de los elementos en el documento de identificación.

Editor SVG[28]: Contiene las herramientas necesarias para crear y editar plantillas, además asocia los atributos de un tipo de documento a la plantilla correspondiente.

2.3 Propuesta de solución

El Sistema de Personalización de Documentos de Identidad basado en tecnologías libres constará con siete módulos (Administración, Recepción de órdenes de impresión, Editor de plantillas, Impresión, Control de calidad, Inventario y Reportes), en la presente investigación solo se desarrollarán tres módulos. Los módulos propuestos son: Recepción de órdenes de impresión, Editor de plantillas y Administración.

El módulo de administración se encarga de gestionar todos los elementos del sistema, entiéndase como elementos del sistema: los usuarios, roles, permisos, idiomas, documentos, aplicaciones, etc. Además incluye la funcionalidad de autenticar usuario.

[28] Gráficos Vectoriales Redimensionables.

El módulo de recepción de órdenes de impresión brinda un servicio que permite que los sistemas externos envíen las órdenes a imprimir y consulten el estado de las mismas. Los estados de las órdenes son definidos por el administrador.

El módulo editor de plantillas permite la creación y edición de plantillas en formato de SVG, proporcionando mayor calidad de las imágenes contenidas en la misma, además añade al diseño figuras geométricas complejas, imágenes y capas.

El proceso de personalización de documentos de identificación comienza con la recepción de órdenes de impresión a partir de las órdenes enviadas por los sistemas externos. Cada orden encapsula los datos de un conjunto de solicitudes, así como el tipo de documento y la oficina de procedencia. Estas solicitudes almacenan los datos referentes a un documento de identificación, según su tipo.

Las órdenes son validadas por el sistema, emitiendo este último, mensajes de aceptación o de error. Si el mensaje es de error se notifica al sistema externo las causas por la que no se aceptó la orden. Si el mensaje es de aceptación, se inserta en el sistema la orden y se procede a continuar con el proceso. Las órdenes se asignan a las estaciones de trabajo según el balanceo de la carga de cada una.

Un documento de identificación contiene la información referente a los campos que contiene y la forma de validarlos (a través de expresiones regulares). A cada documento se le asocia una plantilla, que muestra la forma en que se visualizará el documento.

Las plantillas serán realizadas utilizando un editor SVG, el cual permite añadir al diseño figuras geométricas complejas, imágenes y diferentes capas. Además las plantillas creadas con este formato son muy compactas, escalables, con contornos suavizados y transparencias, facilitando el almacenamiento y calidad de las mismas. Es compatible con los estándares actuales de la web posibilitando su adaptación a cualquier sistema.

El servicio de recepción de órdenes de impresión se realizará a través de un protocolo de servicio web conocido como Transferencia de Estado Representacional (por sus siglas en inglés REST). Este protocolo de servicio web proporciona una buena infraestructura de almacenamiento en caché a través de HTTP, con lo cual mejora el rendimiento. Además los servicios REST son fáciles de integrar con los sitios web existentes y están expuestos a XML para que las páginas HTML puedan consumirlos con facilidad.

2.4 Requisitos del sistema

La captura de requisitos es un paso fundamental para saber exactamente lo que debe hacer y las cualidades o propiedades que deben tener los módulos definidos. Los requisitos funcionales y no funcionales identificados son los siguientes:

2.4.1 Requisitos funcionales (RF)

Módulo administración	**RF22.** Adicionar aplicación.
RF1. Adicionar usuario.	**RF23.** Eliminar aplicación.
RF2. Eliminar usuario.	**RF24.** Modificar aplicación.
RF3. Modificar usuario.	**RF25.** Listar aplicación.
RF4. Listar usuario.	**RF26.** Adicionar estación.
RF5. Adicionar rol.	**RF27.** Eliminar estación.
RF6. Eliminar rol.	**RF28.** Modificar estación.
RF7. Modificar rol.	**RF29.** Listar estación.
RF8. Listar rol.	**Módulo recepción de órdenes de impresión**
RF9. Adicionar permiso.	
RF10. Eliminar permiso.	**RF30.** Recibir orden de impresión.
RF11. Modificar permiso.	**RF31.** Consultar estado de la orden de impresión.
RF12. Listar permiso.	
RF13. Adicionar idioma.	**Módulo editor de plantillas**
RF14. Eliminar idioma.	
RF15. Listar idioma.	**RF32.** Cargar plantilla.
RF16. Autenticar usuario.	**RF33.** Diseñar plantilla de impresión.
RF17. Cambiar idioma.	**RF34.** Adicionar campos de tipos de documento.
RF18. Adicionar documento.	**RF35.** Eliminar campos de tipos de documento.
RF19. Eliminar documento.	**RF36.** Modificar campos de tipos de documento.
RF20. Modificar documento.	**RF37.** Listar campos de tipos de documento.
RF21. Listar documento.	**RF38.** Guardar plantilla.

Tabla 2. Requisitos funcionales. Elaboración propia.

2.4.2 Requisitos no funcionales (RNF)

La captura de los requisitos no funcionales correspondientes a los módulos a desarrollar se realizó teniendo en cuenta lo planteado por el autor Ian Sommerville (41).

- Requerimientos organizacionales.

- ✓ Requerimientos de implementación.

  RNF1. Los módulos deben implementarse usando el lenguaje Python, sobre el marco de trabajo Django.

  RNF2. El sistema gestor de bases de datos será PostgreSQL.

  RNF3. Los módulos deben desarrollarse usando el IDE PyCharm.

  RNF4. Deberá utilizarse la interfaz definida por el CISED, estrategia marcaria XABAL.

  RNF5. El editor de plantillas que se utilizará será SVG, permitiendo el trabajo multicapa y la gestión de figuras geométricas complejas en las plantillas correspondientes a los documentos de identificación.

  RNF6. Las pruebas de rendimiento que se realizarán a la solución se harán con la herramienta Apache JMeter 2.13.

- ✓ Requerimientos de estándares.

  RNF7. Los módulos deben codificarse siguiendo el estándar de codificación *CamelCase*.

- Requerimientos del producto.

- ✓ Requerimientos de rendimiento.

  RNF8. La solución está concebida para un ambiente donde se puede integrar con otros componentes, por lo que los tiempos de respuestas del servicio de recepción tienen que ser menor de 3000 milisegundos, así como la velocidad de procesamiento de información.

  RNF9. La solución para un óptimo rendimiento requiere como mínimo de un espacio de memoria RAM[29] de 128 megabyte.

  RNF10. El sistema debe ser capaz de recibir más de 50 solicitudes por segundo.

- ✓ Requerimientos de fiabilidad.

[29] *RAM* del inglés *Random Access Memory*

RNF11. El sistema debe estar disponible el 98% del tiempo, previendo el uso de un 2 % del tiempo para soporte, actualizaciones y corrección de errores.

RNF12. En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que pueda comprometer la seguridad e integridad del mismo.

- ✓ Requerimientos de eficiencia.

RNF13. El sistema debe demorar como promedio en una petición (tiempo que el servidor se toma en procesar exitosamente una solicitud), de dos (2) a cinco (5) segundos aproximadamente.

RNF14. El sistema debe permitir la navegación de varios usuarios simultáneamente sin que influya en su rendimiento.

2.5 Historias de Usuario (HU)

Las historias de usuario (HU) son la técnica utilizada en XP para especificar los requisitos del *software*, representando una breve descripción del comportamiento del mismo, con el empleo de una terminología del cliente sin lenguaje técnico. Deben tener como características fundamentales: ser independientes unas de otras, negociables, valoradas por los clientes o usuarios, estimables, pequeñas y verificables (42). A continuación se definen las historias de usuario arquitectónicamente más significativas en la solución propuesta, el resto está definido en el Anexo 2.

Historia de Usuario	
Número: HU_9	**Usuario:** Funcionario de recepción, Administrador.
Nombre historia: Recibir orden de impresión.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 2
Programador responsable: Claudia Peralta González.	
Descripción: El sistema realiza una validación con los campos esperados según el tipo de documento de la orden, si todos los campos están correctos el sistema recibe la orden de	

impresión, le asigna un estado, un puesto de trabajo y devuelve un acuse de recibo con el identificador de la orden. En caso contrario rechaza la orden y notifica al sistema externo el motivo del rechazo.
Observaciones: Deben existir el tipo de documento de la orden que se quiere imprimir y además una plantilla asociada a este.
Prototipo: 

No.	Prioridad	Documento	Plantilla	Aplicacion	IP	MAC	Asignada
1	5	documento_1	plantilla 1	Mozilla_1	127.0.0.1	38-60-77-AA-FB-31	PC_1

Tabla 3. HU Recibir orden de impresión. Elaboración propia.

Historia de Usuario	
Número: HU_10	**Usuario:** Sistemas externos.
Nombre historia: Consultar estado de la orden de impresión.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Baja
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Claudia Peralta González.	
Descripción: Permite obtener el estado de la orden de impresión a partir del identificador de la misma. Si no existe la orden devuelve un estado de error con el mensaje correspondiente.	
Observaciones: El sistema externo debe tener asignados los permisos correspondientes para realizar esta solicitud.	

La orden a consultar debe existir en el sistema.
Prototipo: 

Tabla 4. HU Consultar estado de orden de impresión. Elaboración propia.

Historia de Usuario	
Número: HU_12	**Usuario:** Administrador.
Nombre historia: Diseñar plantilla de impresión.	
Prioridad en el negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 3	**Iteración asignada:** 3
Programador responsable: Dennis Durán Arzuaga.	
Descripción: Permite diseñar una plantilla definiendo los campos que tendrá el documento a imprimir y asociar los campos de datos según el tipo de orden. Por cada objeto que se inserte en la plantilla se selecciona el campo al cual se aplicará un remplazo de los datos en el momento de imprimir.	
Observaciones: Una plantilla de impresión siempre está asociada a un tipo de documento.	
Prototipo:	

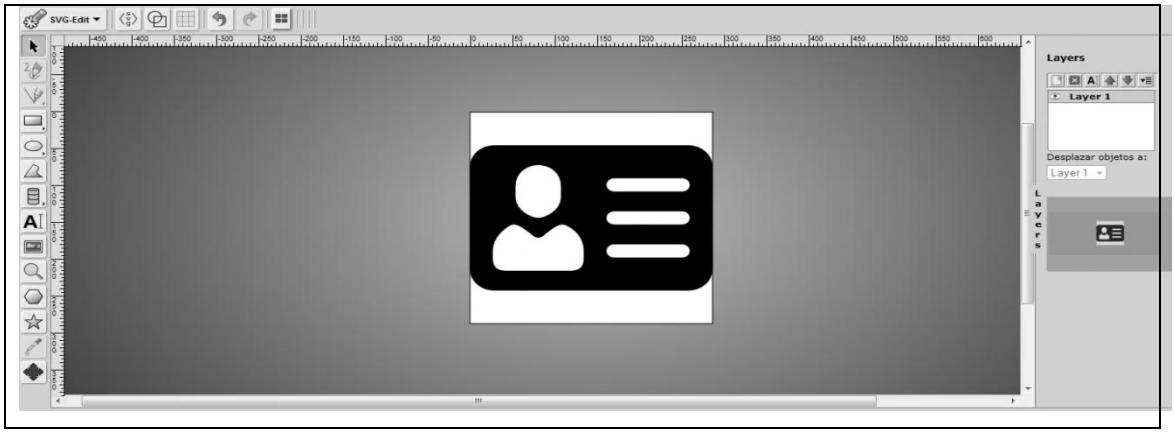

Tabla 5. HU Diseñar plantilla de impresión. Elaboración propia.

2.6 Planificación

La planificación en XP está basada en un conjunto de decisiones tomadas por el cliente de conjunto con los programadores. Los clientes representan las necesidades propias del negocio y los programadores definen los requerimientos técnicos que complementan las necesidades del negocio.
Seguidamente se define el plan de entrega y el plan de iteraciones que regirán el desarrollo.

2.6.1 Plan de entrega

Después de ser identificadas y descritas las HU, se procede a la planificación de la fase de implementación estableciendo una división de 3 iteraciones, en las que se codificarán las 14 HU obtenidas, para una duración total del proyecto de 16 semanas, según el plan de entrega realizado.

Entregable	Iteración	Fin de la iteración
Módulo de administración.	1	Enero de 2015
Módulo de recepción de órdenes de impresión.	2	Marzo de 2015
Módulo editor de plantillas.	3	Abril de 2015

Tabla 6. Plan de entregas. Elaboración propia.

2.6.2 Plan de iteraciones

El ciclo de desarrollo de *software* guiado por XP se caracteriza por ser iterativo e incremental, por lo que se realizan varias iteraciones sobre el sistema antes de su fase de producción.

Los elementos que deben tomarse en cuenta durante la elaboración del plan de iteración son las historias de usuario, la velocidad del proyecto, las pruebas no superadas en la iteración anterior y tareas no terminadas (42).

Para obtener un prototipo funcional al final de cada iteración, se ha definido un plan de iteraciones con el orden en el que se implementarán las historias de usuario.

Iteración	No.HU	Historia de usuario	Semanas estimadas
1	HU_1	Gestionar usuario.	6
	HU_2	Gestionar roles.	
	HU_3	Gestionar permisos.	
	HU_4	Gestionar idioma.	
	HU_5	Autenticar usuario.	
	HU_6	Cambiar idioma.	
	HU_7	Gestionar tipos de documentos.	
	HU_8	Gestionar aplicación.	
2	HU_9	Recibir orden de impresión.	4
	HU_10	Consultar estado de la orden de impresión.	
3	HU_11	Cargar plantilla.	6
	HU_12	Diseñar plantilla de impresión.	
	HU_13	Gestionar campos de tipos de documentos.	
	HU_14	Guardar plantilla.	

Tabla 7. Plan de iteraciones. Elaboración propia.

2.7 Diseño

En el diseño se indica la forma en que se construirá la solución mediante la definición de una estructura lo más sencilla posible que responda a la satisfacción de los requisitos funcionales y no funcionales. En este

punto del ciclo de desarrollo del *software* es cuando se descomponen los trabajos de implementación en partes más manejables que puedan ser realizadas por diferentes equipos de desarrollo.

2.7.1 Tarjetas CRC

Las Tarjetas CRC (Clase, Responsabilidad y Colaboración) constituyen uno de los artefactos de la metodología XP que guía el proceso de desarrollo de la solución propuesta. Sirven para diseñar el sistema en conjunto entre todo el equipo. Permiten reducir el modo de pensar procedural y apreciar la tecnología de objetos. A continuación se muestran las tarjetas CRC de las clases de mayor prioridad para el cliente, el resto está definido en el Anexo 3.

Clase ServicioRecepcion	
Responsabilidades	**Colaboradores**
• Recepcionar orden de impresión.	Solicitud, Orden, Estaciones, Aplicaciones.
• Consultar estado de la orden.	SeguimientoOrdenPorPaso, SeguimientoSolicitudPorPaso.
• Autenticar aplicación.	-
• Consultar campos según documento.	TipoDocumento, AtributoDocumento.
• Consultar tipo de documento.	TipoDocumento.

Tabla 8. Tarjeta CRC ServicioRecepcion. Elaboración propia.

Clase Documento	
Responsabilidades	**Colaboradores**
• Adicionar documento.	AtributoDocumento.
• Eliminar documento.	-
• Listar documento.	-
• Modificar documento.	-
• Actualizar plantillas.	-

Tabla 9. Tarjeta CRC Documento. Elaboración propia.

2.7.2 Patrones de diseño

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí, adaptados para resolver un problema de diseño general en un contexto particular. El mismo identifica: clases, instancias, roles, colaboraciones y la distribución de responsabilidades. Estos modelos que se presentan como parejas de problema/solución con un nombre, codifican buenos principios y sugerencias, relacionados con la asignación de responsabilidades, basados en la recopilación del conocimiento de los expertos en desarrollo de *software* (43).

Los Patrones Generales de *Software* para Asignar Responsabilidades (GRASP) describen los principios fundamentales de la asignación de responsabilidades a objetos. Los patrones de este tipo usados son:

Controlador: Este patrón se utiliza en la clase ControladoraPrincipal para servir como intermediario entre las interfaces y el algoritmo que las implementa. Este patrón se utiliza en las clases Usuarios, Ordenes y Documentos.

Bajo acoplamiento: Plantea que debe existir una alta reutilización entre las funcionalidades de las clases con una mínima dependencia, contribuyendo así al mantenimiento de las mismas. Este patrón se utiliza en todas las clases.

Alta cohesión: Basado en la asignación de responsabilidades teniendo en cuenta que la cohesión permanezca alta. Su utilización facilita la comprensión del diseño e incrementa las capacidades de reutilización. Una alta cohesión caracteriza a las clases con responsabilidades estrechamente relacionadas que no realicen un trabajo enorme. Este patrón se utiliza en todas las clases.

Los patrones *Gang of Four* (GOF) que se utilizaron en el desarrollo del sistema son:

Patrones de creación: El objetivo de estos patrones es abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados (44).

- ✓ **Instancia única:** Garantiza que solamente se crea una instancia de la clase y provee un punto de acceso global a esta. Todos los objetos que utilizan una instancia de esa clase usan la misma instancia.

Patrón de comportamiento: Comprenden la asignación de responsabilidades entre objetos y algoritmos. Estos no solo conciernen a los objetos y las clases sino también la comunicación entre estas, además caracterizan flujos de control complejos que son difíciles de seguir en tiempo de ejecución (43).

- ✓ **Observador:** Permite a los objetos captar dinámicamente las dependencias entre objetos, de tal forma que un objeto notificará a sus objetos dependientes cuando cambie de estado, siendo actualizados automáticamente. Ejemplo de esto se evidencia entre las clases Orden, EstadoOrden y ServicioOrden.

Patrones estructurales: Los patrones estructurales describen como pueden combinarse clases y objetos para formar estructuras más grandes (44).

- ✓ **Decorador:** Extiende la funcionalidad de un objeto dinámicamente de tal modo que es transparente a sus clientes, utilizando una instancia de una subclase de la clase original que delega las operaciones al objeto original. Provee una alternativa muy flexible para agregar funcionalidad a una clase.

  Ejemplo:

```
@login_required(login_url='/')
@permission_required(perm='auth.delete_user', login_url='/inicio/denegado')
def eliminar_usuario(request, id):
    if str(id) == str(request.user.id):
        return acceso_denegado(request)
    else:
        perfil = get_object_or_404(User, pk=id)
        perfil.is_deleted = True
        perfil.is_active = False
        perfil.save()
        return HttpResponseRedirect('/inicio/usuario/eliminado/listaEliminados')
```

Figura 3. Ejemplo del patrón Decorador. Elaboración propia.

2.7.3 Modelo de datos

Un modelo de datos es una colección de conceptos y reglas que se emplean para describir la estructura de una base de datos que incluye entidades, atributos y relaciones entre estos (45). Para diseñar la base de datos se utilizó el modelo físico de datos. Las clases persistentes que a continuación se muestran cubren las necesidades que los módulos a desarrollar puedan presentar.

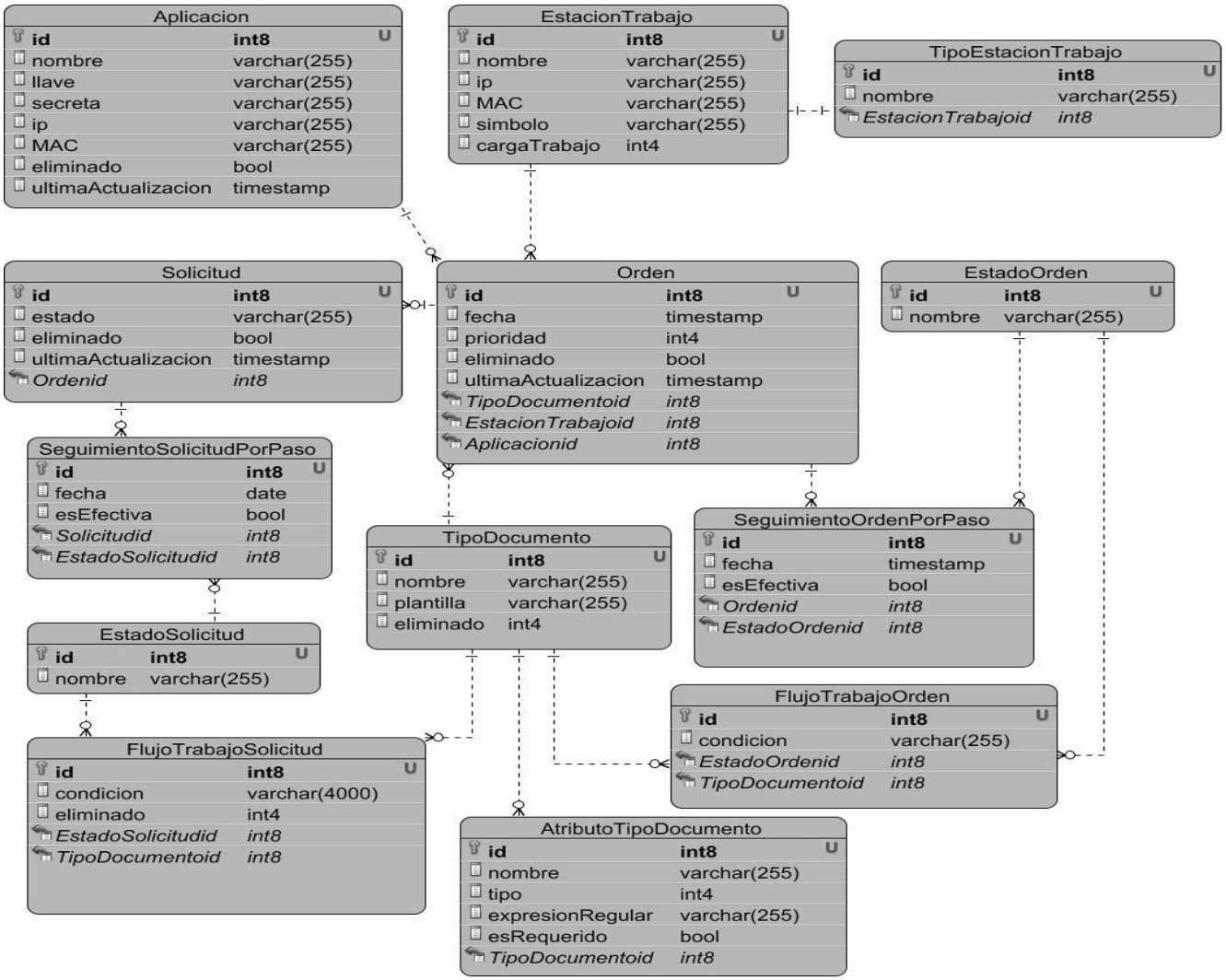

Figura 4. Modelo físico de datos. Elaboración propia.

Aplicación: Se almacenan los datos de los sistemas externos con el objetivo de conservar un registro de las aplicaciones autorizadas a acceder al sistema.

EstacionTrabajo: Contiene los datos referentes a un puesto de trabajo y permite una distribución del trabajo entre estas, balanceando la carga del sistema y agilizando los procesos.

TipoEstacionTrabajo: Nomenclador que contiene la información que identifica unívocamente el tipo de puestos de trabajo.

Orden: Esta entidad almacena los datos generales de un conjunto de solicitudes, así como el tipo de documento, la oficina de procedencia y la referencia a las solicitudes. Además contiene la información de la estación de trabajo a la que fue asignada y al tipo de documento a la que va dirigida.

EstadoOrden: Nomenclador que contiene la información de los estados que puede tener una orden.

SeguimientoOrdenPorPaso: Esta entidad almacena la información referente a los diferentes estados por los que puede pasar una orden y conserva un registro de si se llevaron a cabo satisfactoriamente o no.

Solicitud: Almacena los datos referentes a un documento de identificación, según el tipo de orden.

EstadoSolicitud: Nomenclador que contiene la información de los estados que puede tener una solicitud.

SeguimientoSolicitudPorPaso: Esta entidad almacena la información referente a los diferentes estados por los que puede pasar una solicitud y conserva un registro de si se llevaron a cabo satisfactoriamente o no.

TipoDocumento: Nomenclador que almacena la información que identifica unívocamente a un documento de identificación.

FlujoTrabajoOrden: A partir del estado actual de una orden configura el próximo estado correspondiente en el flujo de trabajo, almacenándolo en esta entidad.

AtributoTipoDocumento: Contiene los atributos con la expresión regular del documento de identificación.

FlujoTrabajoSolicitud: A partir del estado actual de una solicitud configura el próximo estado correspondiente en el flujo de trabajo, almacenándolo en esta entidad.

2.7.4 Arquitectura

La arquitectura de un *software* es la estructura u organización de un sistema que incluye los componentes de este, las propiedades visibles externas de esos componentes y las relaciones que existen entre ellos (46).

La arquitectura permite representar los principales componentes del *software*, las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema.
La arquitectura a utilizar es n-capas (Ver figura 5).

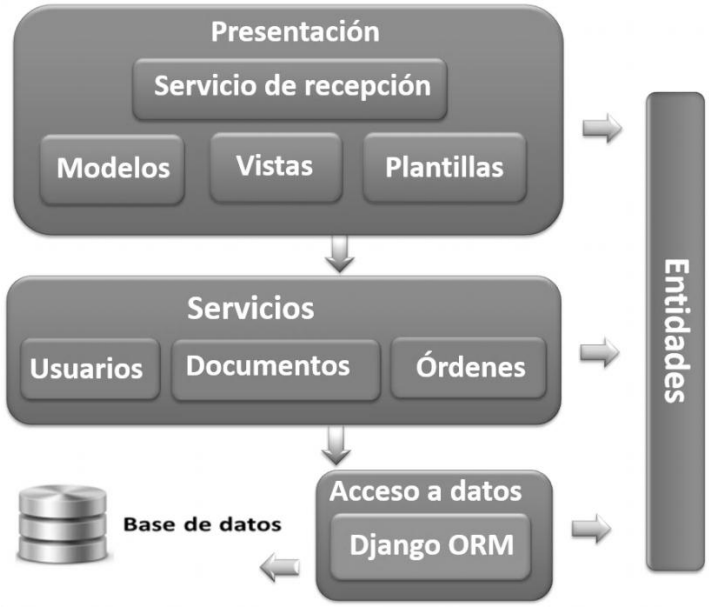

Figura 5. Arquitectura de la solución. Elaboración propia.

Presentación: Esta capa se relaciona con la capa de servicios y con las entidades. Además se trabaja con el patrón arquitectónico Modelo-Vista-Plantilla (MVP), que es una adaptación del patrón Modelo-Vista-Controlador (MVC). En este patrón el modelo contiene toda la información sobre los datos. La vista contiene la lógica que accede a los servicios para formar el modelo y la delega a la plantilla apropiada, es un intermediario entre el modelo y la plantilla. La plantilla recibe las acciones realizadas por el usuario y las envía a la vista, de la cual recibe la información que debe mostrar. En esta capa es donde reside el servicio de recepción para establecer la comunicación con los sistemas externos.
Servicios: Se almacenan todos los servicios necesarios para darle solución a los requerimientos del negocio. Esta capa agrupa las funcionalidades en común de manera que tribute a la reutilización del código y al rendimiento del sistema. Además establece la comunicación con la capa de acceso a datos y con las entidades.
Acceso a datos: En esta capa es donde se realiza el acceso a los datos. Es la encargada de acceder y modificar los mismos, además de comunicarse directamente con la base de datos.

Entidades: En esta capa se implementa la forma de acceder a los datos, contiene todas las clases que representan el dominio de entidades de la base de datos.

2.8 Conclusiones

Luego de definir las características que debe cumplir el Sistema de Personalización de Documentos de Identidad basado en tecnologías libres y de realizar el análisis y diseño correspondiente, se concluye que los requerimientos funcionales y no funcionales obtenidos a partir del proceso de identificación de los requisitos y los artefactos generados, constituyeron elementos claves en la construcción de la propuesta de solución. La utilización del editor SVG para la gestión de las plantillas de los documentos de identificación permitirá incorporarle nuevas capas, imágenes y figuras geométricas complejas. Además con la selección de la arquitectura n-capas y el uso de los patrones de diseño propuestos se garantizará una mayor organización, reutilización de funciones y código más legible.

3 Capítulo III: Implementación y prueba de la propuesta de solución

3.1 Introducción

En el presente capítulo se describe la implementación del *software*, fase donde finalmente se materializa el producto final y se cumple con los requisitos obtenidos al inicio de la investigación. Para ello se definen los estándares de codificación que debe cumplir el equipo de desarrollo, se crea el diagrama de componentes y el de despliegue, además de presentarse las principales interfaces de usuario de la solución. A partir del código resultante y su funcionamiento se ejecutan las pruebas unitarias, de validación y del sistema a los módulos desarrollados.

3.2 Implementación

3.2.1 Estándar de codificación

Es necesario establecer un criterio fijo que proporcione reglas para la creación de nombres para variables y métodos, permitiendo una mejor lectura del *software* y un mejor entendimiento del código por parte de los desarrolladores. Las pautas fundamentales definidas para la implementación son las siguientes:

- ✓ El tamaño máximo de las líneas de código debe ser aproximadamente de cien a ciento veinte caracteres, garantizando la completa visibilidad de las líneas de código sin necesidad de realizar desplazamiento horizontal.
- ✓ Los nombres de las clases, las funciones y las propiedades adoptarán la notación *UpperCamelCase* y no se utilizará el guión bajo como delimitador entre palabras.
- ✓ Los nombres de los atributos, las variables y los parámetros adoptarán la notación *lowerCamelCase*. Esta define que todas las palabras que conformen el nombre, excepto la primera, comenzarán con la primera letra en mayúsculas y el resto de las letras en minúsculas.

3.2.2 Tareas de ingeniería

La metodología XP plantea que el trabajo de una iteración es expresado en tareas de ingeniería que emergen de las historias de usuario. Estas tareas son asignadas a un programador, siendo el

programador el responsable de su implementación. Las historias de usuario brindan un escaso nivel de detalle para afrontar la implementación por lo que las tareas de ingeniería juegan un papel fundamental al indicar a los programadores las acciones a realizar por cada una de ellas.

A continuación se muestran las tareas de ingeniería correspondientes a la iteración 3. Las tareas de las demás iteraciones, así como la descripción de estas se muestran en el Anexo 4.

Iteración 3	
Historia de usuario	**Tareas**
Cargar plantilla.	1. Verificar si existen plantillas de documentos en la base de datos. 2. Obtener la plantilla para la edición.
Diseñar plantilla de impresión.	1. Crear una nueva plantilla.
Gestionar campos de tipos de documentos.	**Insertar campos de tipos de documentos** 1. Seleccionar los campos a añadir según el documento. 2. Guardar los campos de tipos de documentos. 3. Mostrar mensaje. **Eliminar campos de tipos de documentos** 1. Seleccionar el campo. 2. Eliminar el campo. 3. Mostrar mensaje de campo eliminado correctamente. **Modificar campos de tipos de documentos** 1. Seleccionar el documento. 2. Seleccionar el campo a modificar. 3. Modificar el campo en el documento. 4. Guardar modificaciones del documento. 5. Mostrar mensaje de modificación efectuada con éxito. **Listar campos de tipos de documentos** 1. Seleccionar documentos. 2. Seleccionar campos de tipos de documentos

	que desea listar.
Guardar plantilla.	1. Guardar plantilla en la base de datos.

Tabla 10. Tareas de ingeniería en la tercera iteración. Elaboración propia.

3.2.3 Diagrama de componentes

Los diagramas de componentes modelan la vista estática del *software*, mostrando la organización y las dependencias lógicas entre un conjunto de componentes del *software*, que pueden ser librerías, binarios, ejecutables y códigos fuentes. Las dependencias en este diagrama indican que un componente utiliza los servicios ofrecidos por otro componente. A continuación se expone el diagrama definido para la solución propuesta:

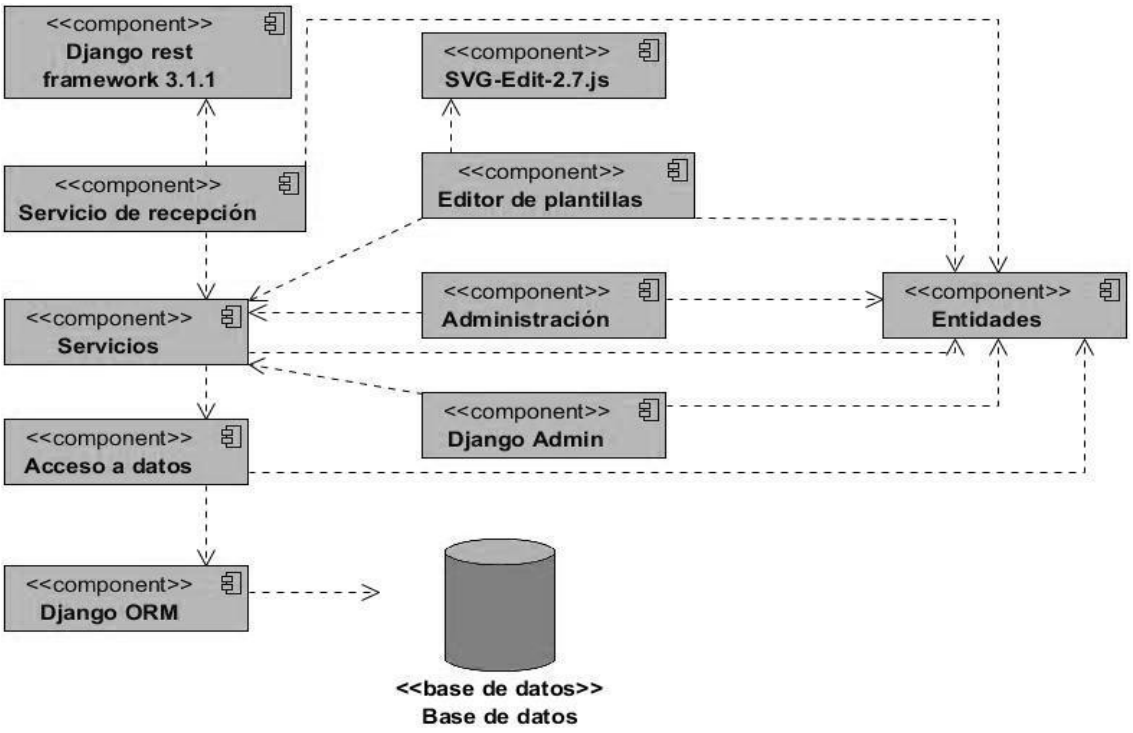

Figura 6. Diagrama de componentes. Elaboración propia.

Descripción del diagrama de componentes

Servicio de recepción: Componente encargado de brindar el servicio de recepción de órdenes de impresión a los sistemas externos.

Servicios: Componente encargado de controlar toda la lógica de negocio del sistema.

Django ORM: Componente encargado de crear, mapear, acceder y modificar la información alojada en la base de datos.

Acceso a datos: Componente encargado de acceder a los datos en la base de datos.

Editor de plantillas: Componente encargado de la gestión de las plantillas SVG.

Entidades: Componente que contiene todas las clases que representan el dominio de entidades de la base de datos.

SVG-*Edit:* Componente basado en JavaScript, HTML 5 y CSS utilizado para la edición de las plantillas SVG.

Django *Admin:* Componente encargado de gestionar todos los elementos (usuarios, aplicaciones, etc.) referentes al negocio, utilizando la administración por defecto del marco de trabajo Django.

Administración: Componente encargado de gestionar las plantillas, los tipos de documentos, los roles, usuarios, estaciones, aplicaciones, etc.

Django *rest framework:* Componente encargado de renderizar en formato json la respuesta del SPDI a las peticiones (GET, POST) de los sistemas externos. Además brinda un conjunto de herramientas para la construcción de APIs[30] web.

3.2.4 Diagrama de despliegue

El diagrama de despliegue es utilizado para capturar los elementos de configuración del procesamiento, las conexiones entre esos elementos y visualizar la distribución de los componentes de *software* en los nodos físicos. Entre los nodos existen relaciones que representan los protocolos de comunicación que se utilizan para acceder a cada uno.

[30] abreviatura de *Aplication Programming Interface*. Un API no es más que una serie de servicios o funciones que el Sistema Operativo ofrece al programador.

En la siguiente figura puede visualizarse el diagrama de despliegue definido para la solución propuesta. El Sistema Externo representa las aplicaciones que enviarán órdenes de impresión al Servidor Web. La PC Cliente representa las estaciones de trabajo de los usuarios que se conectan al sistema, las cuales realizan peticiones al Servidor Web mediante el protocolo HTTP\HTTPS. Este servidor establecerá una conexión mediante el protocolo TCP\IP al servidor de base de datos.

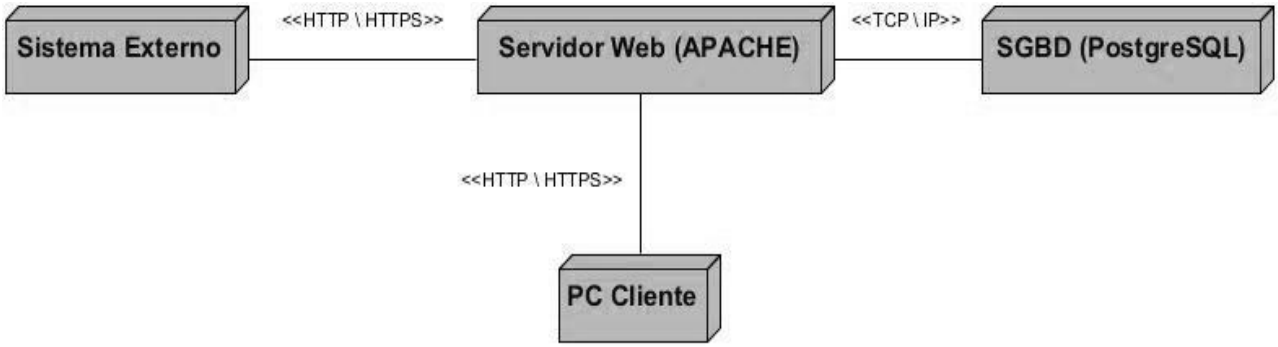

Figura 7. Diagrama de despliegue. Elaboración propia.

3.2.5 Interfaces de usuario

La interfaz de usuario es el medio con que el usuario puede comunicarse con una máquina, un equipo o una computadora, y comprende todos los puntos de contacto entre el usuario y el equipo. Normalmente suelen ser fáciles de entender y fáciles de accionar. A continuación se muestran algunas interfaces del sistema.

Figura 8. Interfaz de inicio del sistema. Elaboración propia.

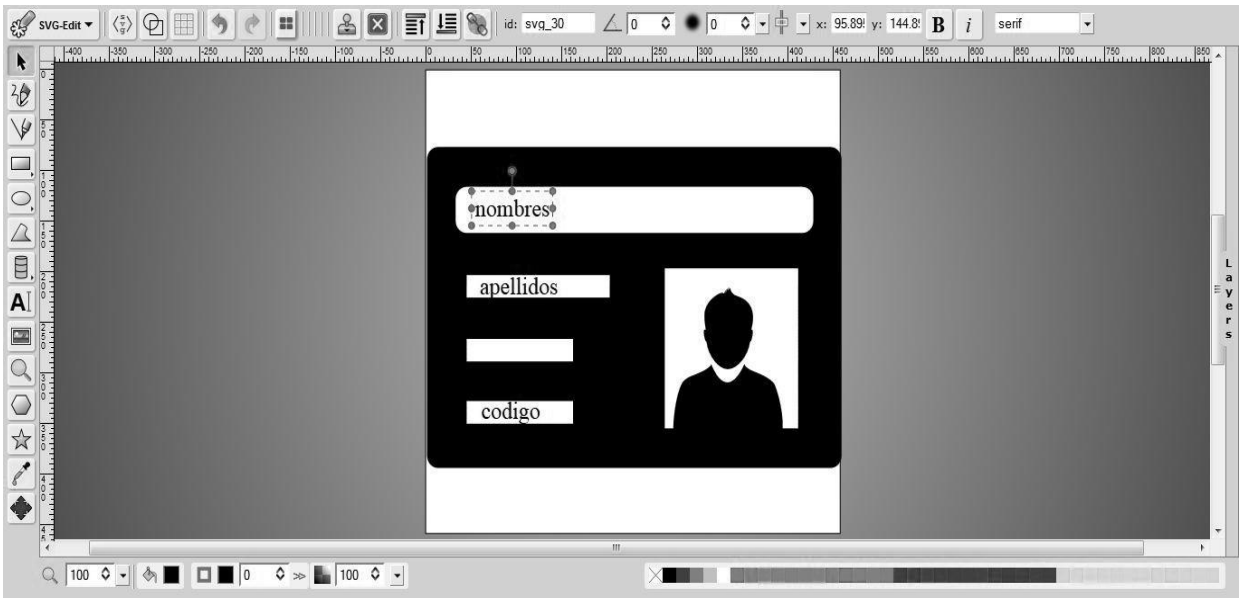

Figura 9. Interfaz del editor SVG. Elaboración propia.

3.3 Pruebas

Las pruebas de *software* son aquellos procedimientos que se realizan para verificar la calidad de un producto de *software* y pueden ser aplicadas periódicamente. Estas tienen como objetivo fundamental la identificación de posibles errores, además representan una revisión final de las especificaciones del diseño y de la codificación.

3.3.1 Estrategia de pruebas

Roger S. Pressman plantea que una estrategia de prueba del *software* integra los métodos de diseño de caso de pruebas en una serie bien planeada de pasos que desembocará en la eficaz construcción de *software*. Una estrategia de prueba debe ser lo suficientemente flexible como para promover un enfoque personalizado. Al mismo tiempo, debe ser lo adecuadamente rígido como para promover una planeación razonable y un seguimiento administrativo del avance del producto (47).

Las estrategias de prueba que plantea Pressman son las siguientes:

- ✓ Pruebas de unidad.
- ✓ Pruebas de integración.
- ✓ Pruebas de validación.
- ✓ Pruebas del sistema.

Se realizarán a la solución las pruebas de unidad, de validación y del sistema.

3.3.1.1 Pruebas de unidad

Las pruebas de unidad se realizan con el objetivo de ejecutar un código fuente llamando directamente a los métodos de una clase pasándole a estos los parámetros apropiados. Los métodos de pruebas unitarias residen en clases *Test*, que se almacenan en los archivos de código fuente (48).

Las pruebas de unidad son las que van enfocadas a los elementos más pequeños del software. Son aplicables a funcionalidades para verificar que los flujos de control y de datos están cubiertos, y funcionan como se espera. La prueba de unidad siempre está orientada a caja blanca.

Las **pruebas de caja blanca** se basan en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos, logrando como resultado que disminuya en un gran porciento el

número de errores existentes en los sistemas y por ende una mayor calidad y confiabilidad. Mediante la prueba de la caja blanca el ingeniero del *software* puede obtener casos de prueba que (49):

- ✓ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✓ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✓ Ejecuten todos los bucles en sus límites operacionales.
- ✓ Ejerciten las estructuras internas de datos para asegurar su validez.

En los módulos creados, se aplicó la técnica de pruebas de caja blanca del camino básico, puesto que permite obtener una medida de la complejidad lógica de un diseño y usarla como guía para la definición de un conjunto básico. La idea es derivar casos de prueba a partir de un conjunto dado de caminos independientes por los cuales puede circular el flujo de control. Para obtener dicho conjunto de caminos independientes se construye el grafo de flujo asociado y se calcula su complejidad ciclomática. Los casos de prueba derivados del camino básico garantizan que durante la prueba se ejecute por lo menos una vez cada sentencia del programa.

Se realizaron pruebas de caja blanca a las funcionalidades adicionar aplicación y consultar campos de documento.

1. Funcionalidad: AdicionarAplicacion.

La funcionalidad implementada para adicionar una aplicación se muestra en el Anexo 5.

Grafo de flujo

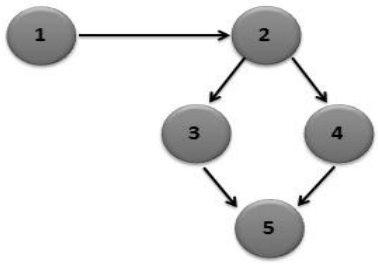

Figura 10. Grafo de Flujo: Funcionalidad AdicionarAplicacion. Elaboración propia.

Caminos independientes

Camino 1. (1-2-3-5)

Camino 2. (1-2-4-5)

Complejidad ciclomática

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) –N (Nodos)) + 2 V (G) = (5 –5) + 2 = 2	V (G) = P (Nodos Predicados) + 1 V (G) = 1 + 1 =2	V (G) = R = 2

Tabla 11. Complejidad ciclomática de la funcionalidad: AdicionarAplicacion. Elaboración propia.

Se preparan los casos de prueba que obliguen la ejecución de cada camino del conjunto básico

Camino 1. (1-2-3-5)

- ✓ Entrada: el método utilizado sea igual a POST.
- ✓ Salida: registra aplicación en el sistema.
- ✓ Precondiciones: null.

Camino 2. (1-2-4-5)

- ✓ Entrada: el método utilizado sea distinto a POST.
- ✓ Salida: mensaje de error
- ✓ Precondiciones: null.

2. Funcionalidad: ConsultarCamposDeDocumento.

La funcionalidad implementada para adicionar una aplicación se muestra en el Anexo 6.

Grafo de flujo

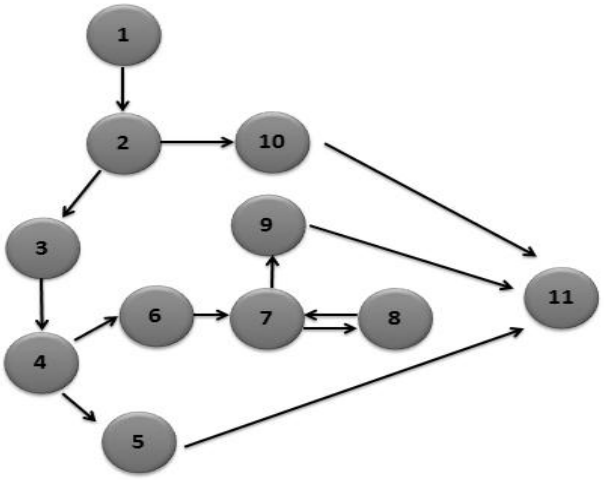

Figura 11. Grafo de Flujo: Funcionalidad ConsultarCamposDeDocumento. Elaboración propia.

Caminos independientes

Camino 1. (1-2-3-4-5-11)

Camino 2. (1-2-3-4-6-7-8-7-9-11)

Camino 3. (1-2-10-11)

Complejidad ciclomática

Fórmula 1	Fórmula 2	Fórmula 3
V (G) = (A (Aristas) –N (Nodos)) + 2 V (G) = (12 –11) + 2 = 3	V (G) = P (Nodos Predicados) + 1 V (G) = 2 + 1 =3	V (G) = R = 3

Tabla 12. Complejidad ciclomática de la funcionalidad: ConsultarCamposDeDocumento. Elaboración propia.

Se preparan los casos de prueba que obliguen la ejecución de cada camino del conjunto básico

Camino 1. (1-2-3-4-5-11)

- ✓ Entrada: el método utilizado sea igual a GET y no existan documentos.
- ✓ Salida: mensaje de error.
- ✓ Precondiciones: null.

Camino 2. (1-2-3-4-6-7-8-7-9-11)

- ✓ Entrada: el método utilizado sea igual a GET y existan documentos.
- ✓ Salida: Lista de nombres de los atributos del documento.
- ✓ Precondiciones: exista el documento.

Camino 3. (1-2-10-11)

- ✓ Entrada: el método utilizado sea distinto a GET.
- ✓ Salida: mensaje de error.
- ✓ Precondiciones: null.

3.3.1.2 Pruebas de validación

En las pruebas de validación desaparece la distinción entre *software* convencional y orientado a objetos, estas se concentran en las acciones visibles para el usuario y en la salida del sistema que este puede reconocer. La validación del software se logra mediante una serie de pruebas que demuestran que se cumplen los requisitos funcionales. Se realiza un plan de pruebas para asegurar que se satisfagan todos los requisitos y que alcance todas las características de comportamiento (50).

Para efectuar las pruebas de validación se realizaron **pruebas de caja negra**, estas se centran principalmente en los requisitos funcionales del *software* y se llevan a cabo sobre la interfaz del *software*, por lo que los casos de prueba pretenden demostrar que las funciones del *software* son operativas. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

A continuación se muestra el caso de prueba de aceptación correspondiente a la historia de usuario **Autenticar usuario** y **Recibir orden de impresión**. Los casos de prueba de aceptación definidos por el cliente para el resto de las historias de usuario se especifican en el Anexo 7.

	Caso de prueba
Código CP: CP_HU5	**Nombre HU:** Autenticar usuario.
Nombre de la persona que realiza la prueba: Claudia Peralta González.	

Descripción de la prueba: Se autentica un usuario en el sistema introduciendo su nombre de usuario y contraseña.
Condiciones de ejecución: El usuario debe existir en el sistema.
Entrada/Pasos de ejecución: 1. Entrada del nombre de usuario y contraseña. 2. Validar si el usuario existe en el sistema. **3.** Validar que la contraseña proporcionada coincida con la contraseña almacenada en el sistema correspondiente al usuario.
Resultado Esperado: Usuario autenticado en el sistema.
Evaluación: Satisfactoria.
Prototipo: 

Tabla 13. Caso de prueba de la funcionalidad: AutenticarUsuario. Elaboración propia.

Caso de prueba	
Código CP: CP_HU9	**Nombre HU:** Recibir orden de impresión.
Nombre de la persona que realiza la prueba: Dennis Durán Arzuaga.	
Descripción de la prueba: Recibe la orden de impresión del sistema externo enviada a través de un servicio.	
Condiciones de ejecución: El sistema externo debe estar autenticado en el SPDI.	
Entrada/Pasos de ejecución: 1. Comprueba que existan plantillas de documentos en el sistema. 2. Comprueba que existan aplicaciones permitidas para enviar órdenes de impresión. 3. Comprueba que existan estaciones de trabajo en el sistema. 4. Validar datos de entrada de la orden de impresión. 5. Insertar orden en el sistema.	
Resultado Esperado: Mensaje de notificación para el sistema externo que indica el registro satisfactorio de la orden de impresión.	
Evaluación: Satisfactoria.	
Prototipo: Respuesta SPDI "Orden recepcionada satisfactoriamente,el id de su orden es: 2, la cantidad de solicitudes es: 2"	

Tabla 14. Caso de prueba de la funcionalidad: RecibirOrdenDeImpresion. Elaboración propia.

3.3.1.3 Pruebas del sistema

Las pruebas del sistema abarcan una serie de pruebas diferentes cuyo propósito principal es ejercitar profundamente el sistema de cómputo. Aunque cada prueba tiene un objetivo distinto, todas trabajan para verificar que se hayan integrado adecuadamente todos los elementos del sistema y que realizan las funciones apropiadas (51). La prueba del sistema realizada a la solución fue la prueba de rendimiento.

Las **pruebas de rendimiento** tienen que diseñarse para asegurar que el sistema pueda procesar su carga esperada. Esto normalmente implica planificar una serie de pruebas en las que la carga se va incrementando regularmente hasta que el rendimiento del sistema se hace inaceptable. Las pruebas de rendimiento se ocupan tanto de demostrar que el sistema satisface sus requerimientos como de describir problemas y defectos en el sistema.

Estas pruebas implican estresar el sistema realizando demandas que están fuera de los límites del diseño del *software*, acercándose a la máxima carga del diseño del sistema hasta que el sistema falla.

Los módulos desarrollados correspondientes al SPDI fueron sometidos a pruebas de carga y estrés para medir su funcionamiento. Estas pruebas estuvieron enfocadas al análisis del comportamiento de los tiempos de respuesta del servicio de recepción de las órdenes en dependencia de condiciones variables como el crecimiento de las conexiones concurrentes de los sistemas externos al SPDI. Esta prueba se realizó primeramente para un total de quinientos sistemas externos y después para un total de mil sistemas externos cada uno enviando cien órdenes de impresión al SPDI por segundo, en una computadora con un procesador Intel *core* i3 y 4 gigabyte de RAM. Para realizar las pruebas de rendimiento se utilizó la herramienta Apache JMeter en su versión 2.13.

3.3.2 Resultados de las pruebas

Después de haber realizado las pruebas unitarias a los módulos desarrollados se detectaron varias no conformidades, por lo que se ejecutaron dos iteraciones de revisión. En una primera iteración se encontraron siete funcionalidades con errores, las cuales fueron resueltas en la iteración siguiente y en la segunda no fueron encontradas no conformidades (Ver figura 12).

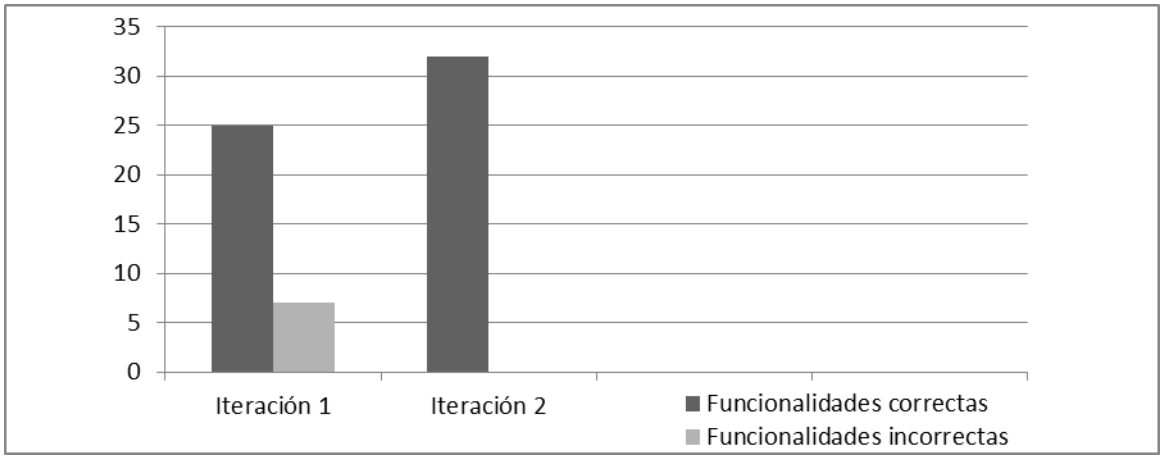

Figura 12. Resultados de las pruebas unitarias. Elaboración propia.

La realización de las pruebas de validación se ejecutó en tres iteraciones obteniéndose en cada una 9, 3 y 0 no conformidades respectivamente. Estas dificultades fueron solucionadas en un corto plazo, antes de pasar a la iteración posterior. (Ver figura 13).

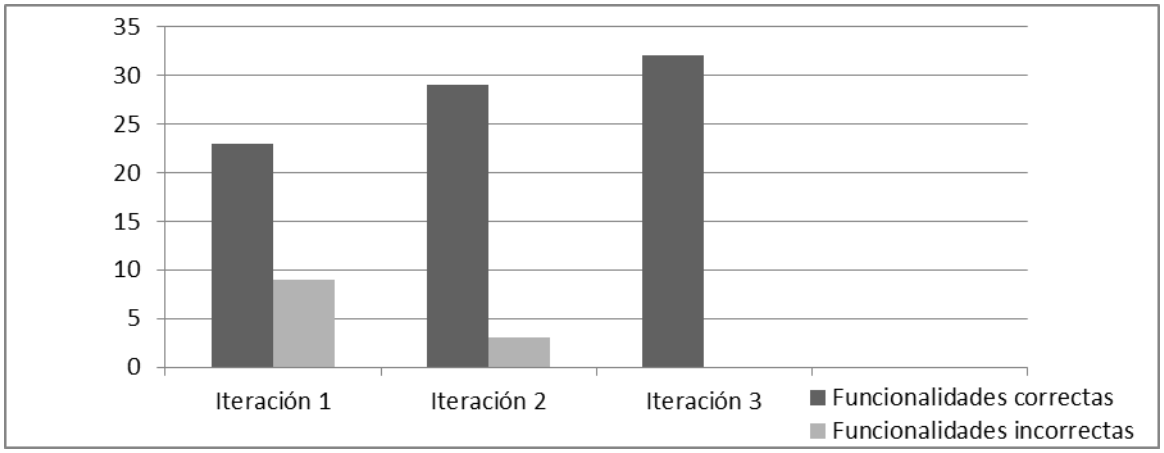

Figura 13. Resultados de las pruebas de validación. Elaboración propia.

La realización de las pruebas de rendimiento demostró que el sistema soporta el procesamiento concurrente de varias órdenes de impresión. A continuación se muestra el resultado de la prueba realizada para mil sistemas externos: el tiempo alcanzado en la ejecución de las peticiones fue satisfactorio, analizando una solicitud en un tiempo mínimo de 479 milisegundos y un tiempo máximo de

2169 milisegundos para un rendimiento final de 50,4 peticiones procesadas exitosamente por el servidor en un segundo.

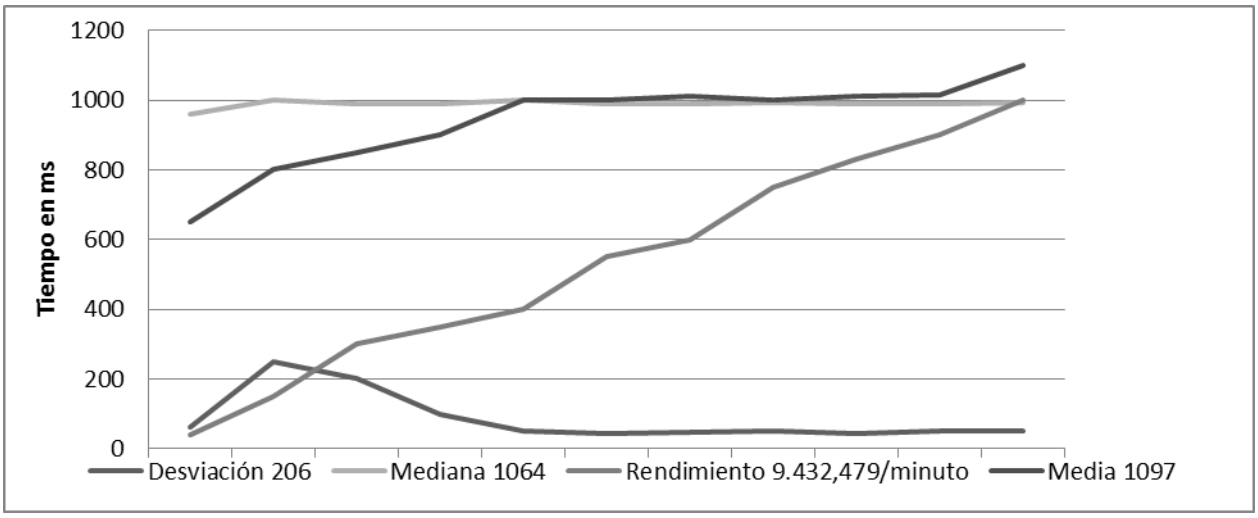

Figura 14. Resultados de las pruebas del sistema. Elaboración propia.

3.4 Conclusiones

Durante la etapa de implementación el uso de los estándares de codificación definidos permitió desarrollar un código reutilizable, comprendido por todo el equipo de desarrollo haciendo más simple la puesta a punto de la solución propuesta. El desglose de las historias de usuario en tareas de ingeniería contribuyó a facilitar el trabajo de programación al indicar específicamente las funcionalidades a desarrollar. Además la representación del diagrama de componentes y el diagrama de despliegue, visualizaron la estructura y funcionamiento de los módulos. Se constató que el desarrollo guiado por pruebas asegura la ejecución correcta de la solución en todo el período de implementación. La realización de las mismas permitió mitigar las no conformidades encontradas, obteniendo un correcto funcionamiento de los módulos desarrollados, quedando de esta forma el cliente satisfecho con el producto obtenido.

Conclusiones

La investigación desarrollada y los resultados obtenidos permiten a los autores plantear las siguientes conclusiones:

- ✓ La búsqueda y análisis de sistemas semejantes demostró que las herramientas analizadas a nivel internacional utilizan tecnologías similares para la personalización de documentos en cuanto al llenado de la página de datos, además no son de código abierto. Los resultados a nivel nacional muestran que algunas de las soluciones encontradas no poseen un editor de plantillas avanzado y otras solo imprimen un tipo de documento. Lo anterior evidenció que no existe un sistema que permita mejorar la calidad del Sistema de Personalización de Documentos de Identidad para la recepción de solicitudes de impresión y edición de plantillas basado en tecnologías libres.
- ✓ El estudio de los sistemas y los conceptos teóricos permitió sentar las bases teóricas para la personalización de los documentos de identificación y evidenció la necesidad de tener en cuenta varios elementos que permiten mejorar la calidad de los documentos, a través de diseños más adaptables y seguros que incorporan medidas de seguridad.
- ✓ El enfoque ágil propuesto por la metodología XP, el uso de tecnologías y herramientas libres, permitieron realizar el diseño de los módulos de edición de plantillas, recepción de órdenes y administración del nuevo SPDI, en concordancia con las especificaciones del cliente.
- ✓ La selección del estándar de codificación permitió la implementación de los módulos de recepción de órdenes, edición de plantillas y administración, además la utilización del editor SVG para la gestión de las plantillas de los documentos de identificación facilitó la realización del módulo edición de plantillas.
- ✓ La realización de las pruebas unitarias, de validación y de rendimiento permitió comprobar el correcto funcionamiento de los módulos desarrollados, además demostró su validez con resultados satisfactorios, obteniendo un rendimiento de 60,4 órdenes de impresión procesadas por segundo.

Recomendaciones

Los autores proponen las siguientes recomendaciones:

- ✓ Implementar los módulos de Control de la calidad, Inventario, Impresión y Reportes para obtener el SPDI basado en tecnologías libres.
- ✓ Incorporar funcionalidades que permitan la personalización de documentos electrónicos, pues estos tienen un mayor potencial en cuanto a la seguridad y permiten un mayor almacenamiento de datos.

Bibliografía referenciada

1. Safran Morpho. [En línea] 2010. [Citado el: 10 de febrero de 2015.] http://www.morpho.com/identification/identity-management/id-documents-personalization-solution/.

2. **Jurídicos., Dirección de Servicios.** Resolución No 179/14. La Habana : Universidad de las Ciencias Informáticas, 2014.

3. **Stallman, Richard M.** *Software libre para una sociedad libre.* s.l. : Free Software Foundation, 1995-1999.

4. **Malcolm, Bain.** *Aspectos legales y de explotación del software libre.* 2000. Vol. Vol 1.

5. Alegsa.com.ar. [En línea] [Citado el: 15 de enero de 2015.] http://www.alegsa.com.ar/Dic/multiplataforma.php.

6. **Expansion, Secured Boundaries.** SuperCom. [En línea] 2013. [Citado el: 12 de febrero de 2015.] http://www.supercom.com/index.asp?module=category&item_id=149.

7. DATYS.Tecnologías & Sistemas. [En línea] [Citado el: 2 de noviembre de 2014.] http://www.datys.cu/wpinfproducto.aspx?72.

8. **Morpho, Safran.** Identificación.Morpho Perso Sistema de Personalización de Documentos. Colombia : s.n.

9. 3M. [En línea] [Citado el: 4 de octubre de 2014.] http://solutions.3m.com.mx/wps/portal/3M/es_MX/SSD_LA/Security_Systems/Product/One/Two/Three/.

10. HID.La fuente fiable para una solución de identidad segura. [En línea] [Citado el: 11 de octubre de 2014.] http://www.hidglobal.mx/products/software/asure-id/exchange.

11. *Sistema de manejo de llaves y certificados digitales para soluciones de emisión de documentos de identificación electrónicos. .* **Correa Rivera, Adrian y González Rodríguez, Guillermo.** La Habana : UCI, 2011.

12. **Escribano, G.F.** *Extreme Programming.* 2002.

13. **Canós, José H., Letelier, Patricio y Penadés, María Carmen.** *Metodologías Agiles en el Desarrollo de Software.* 2013.

14. **Schwaber, Ken y Beedle, Mike.** *The Agile software development with Scrum.* s.l. : Prentice Hall, 2008.

15. **Schwaber, Ken.** *Agile Project Management with Scrum.* s.l. : Microsoft Press, 2008.

16. **Flores, Ervin.** Universidad Unión Bolivariana.Ingeniería de Software. [En línea] Universidad Bolivariana. [Citado el: 24 de noviembre de 2014.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

17. **Larmn, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2004.

18. **McClure, Carma.** *The CASE Experience.* Abril 1989.

19. *Componente de clasificación de huellas dactilares .* **Hernández, Alicia Delgado.** La Habana : UCI, 2013.

20. Visual Paradigm. [En línea] http://www.visual-paradigm.com/.

21. **Baeza, Pablo Nicolás.** Visual Paradigm DB Visual ARCHITECT SQL. [En línea] http://www.docstoc.com/docs/96492173/Visual.

22. **Duque, Raúl González.** *Python para todos.* pág. 115.

23. **Martelli, Alex.** *Python.* 2008. 978-84-415-2317-3 84-415-2317-7.

24. **Dayley, Brad.** *Python Phrasebook.* 2007.

25. **Herminio Santos Heredia, ¿Qué es el PHP?** Maestros del Web. [En línea] 2001. [Citado el: 15 de enero de 2015.] http://www.maestrosdelweb.com/phpintro.

26. *Herramienta para informatizar la gestión de los subprocesos pruebas y revisiones del aseguramiento de localidad en el Centro GEySED.* **Yeilen Morales Aliaga, Lisbet Daniel Rodríguez.** La Habana : UCI, 2013.

27. **Gutérrez, Javier J.** *¿Qué es un framework web?* Sevilla : Universidad de Sevilla, 2011.

28. **Montero, Sergio Infante.** *Django para perfeccionistas con deadlines.* s.l. : Eugenia Tobar. pág. 103.

29. **Holovaty, Adrian y Kaplan-Moss.** *The Definitive Guide to Django: Web Development Done Right.* 2009. 978-1-4302-1937-8.

30. Web2py. [En línea] [Citado el: 24 de marzo de 2015.] http://www.web2py.com.ar/examples/default/what.

31. **Álvarez, Miguel Angel.** Manual de jQuery. [En línea] [Citado el: 26 de abril de 2015.] http://www.desarrolloweb.com/manuales/manual-jquery.html.

32. **Sánzhez, Jorge.** *Sistemas Gestores de Base de Datos.* 2009.

33. **Neil, Matthew y Stones, Richard.** *Beginning Databases with PostgreSQL.* 2005.

34. **POSTGRESQL.** Sobre PostgreSQL. [En línea] [Citado el: 20 de octubre de 2014.] http://www.postgresql.org.es/sobre_postgresql.

35. **Cobo, Ángel, Gómez, Patricia, Pérez, Daniel and Rocha, Rocío.** *PHP y MySQL Tecnologías para el desarrollo de aplicaciones web.* 1ra. España. 84-7978-706-6.

36. **Barahona, Jesús M. González.** *Intoducción al software libre.* Madrid : s.n.

37. **Silva, Vladimir.** *Practical Eclipse Rich Client Platform Projects.* New York : Apress : s.n., 2009. 978-1-4302- 18128-9.

38. PR Newswire. [En línea] [Citado el: 5 de noviembre de 2014.] http://www.prnewswire.co.uk/news-releases/con-pycharm-los-desarrolladores-de-python-obtienen-finalmente-una-ide-potente-155001805.html.

39. PyCharm. [En línea] [Citado el: 10 de noviembre de 2014.] http://www.jetbrains.com/pycharm/ .

40. **Kabir, Mohammed J.** *La biblia del servidor Apache.*

41. **Sommerville, Ian.** *Ingeniería de Software. Séptima edición.* Madrid : PEARSON EDUCATION, 2005. pág. 712. Vol. Parte II. 84-7829-074-5.

42. **Letelier, Patricio y Penadés, Mª Carmen.** Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP). España : Universidad Politécnica de Valencia.

43. **GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J.** *Patrones de diseño.* 2000.

44. **Cooper, James W.** *The Design Patterns.Java Companion.* octubre 2, 1998.

45. **Alvarado, José Manuel.** *Bases de datos. Unidad 2. Modelo de datos.* s.l. : Universidad Nacional de Ingeniería.

46. **Pressman, Roger S.** *Ingeniería del Diseño.Ingeniería de Software.Sexta Edición.* s.l. : McGraw Hill. Vol. Parte II.

47. **Pressman, Roger .S.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 382.

48. **Microsoft.** [En línea] [Citado el: 31 de marzo de 2015.] http://msdn.microsoft.com.

49. **Universidad de las Ciencias Informáticas.** Entorno Virtual de Aprendizaje . [En línea] 2013. [Citado el: 24 de marzo de 2014.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.

50. **Pressman, Roger .S.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 404.

51. **Pressman, Roger.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 406.

Bibliografía consultada

1. **Hogl, Hubert.** *Ten Years of Python Programming.* december 3, 2004.

2. **Infante, Sergio Montero.** *Django para perfeccionistas con deadlines.* Eugenia Tobar.

3. **Rossum, Guido.** *El tutorial de Python.* [ed] Fred Drake.

4. **Ambler, Scott.** *Agile Modeling: Effective Practices for eXtreme Programming and the UnifiedProcess*. s.l. : John Wiley & Sons, 2002.

5. **Lee, Kent D.** *Python Programming Fundamentals.* s.l. : Springer, 2010.

6. **Sundara, Oscar.** *Visual Paradigm for UML.* s.l. : International Book Market Service, 2013. ISBN-6139166535.

7. **Bennett, James.** *Practical Django Projects.* s.l. : Apress, 2008. ISBN-13: 978-1-59059-996-9.

8. Safran Morpho. [En línea] 2010. [Citado el: 10 de febrero de 2015.] http://www.morpho.com/identification/identity-management/id-documents-personalization-solution/.

9. **Jurídicos., Dirección de Servicios.** Resolución No 179/14. La Habana : Universidad de las Ciencias Informáticas, 2014.

10. **Stallman, Richard M.** *Software libre para una sociedad libre.* s.l. : Free Software Foundation, 1995-1999.

11. **Malcolm, Bain.** *Aspectos legales y de explotación del software libre.* 2000. Vol. Vol 1.

12. Alegsa.com.ar. [En línea] [Citado el: 15 de enero de 2015.] http://www.alegsa.com.ar/Dic/multiplataforma.php.

13. **Expansion, Secured Boundaries.** SuperCom. [En línea] 2013. [Citado el: 12 de febrero de 2015.] http://www.supercom.com/index.asp?module=category&item_id=149.

14. DATYS.Tecnologías & Sistemas. [En línea] [Citado el: 2 de noviembre de 2014.] http://www.datys.cu/wpinfproducto.aspx?72.

15. **Morpho, Safran.** Identificación.Morpho Perso Sistema de Personalización de Documentos. Colombia : s.n.

16. 3M. [En línea] [Citado el: 4 de octubre de 2014.] http://solutions.3m.com.mx/wps/portal/3M/es_MX/SSD_LA/Security_Systems/Product/One/Two/Three/.

17. HID.La fuente fiable para una solución de identidad segura. [En línea] [Citado el: 11 de octubre de 2014.] http://www.hidglobal.mx/products/software/asure-id/exchange.

18. *Sistema de manejo de llaves y certificados digitales para soluciones de emisión de documentos de identificación electrónicos. .* **Correa Rivera, Adrian y González Rodríguez, Guillermo.** La Habana : UCI, 2011.

19. **Escribano, G.F.** *Extreme Programming.* 2002.

20. **Canós, José H., Letelier, Patricio y Penadés, María Carmen.** *Metodologías Agiles en el Desarrollo de Software.* 2013.

21. **Schwaber, Ken y Beedle, Mike.** *The Agile software development with Scrum.* s.l. : Prentice Hall, 2008.

22. **Schwaber, Ken.** *Agile Project Management with Scrum.* s.l. : Microsoft Press, 2008.

23. **Flores, Ervin.** Universidad Unión Bolivariana.Ingeniería de Software. [En línea] Universidad Bolivariana. [Citado el: 24 de noviembre de 2014.] http://ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.

24. **Larmn, Craig.** *UML y Patrones. Introducción al análisis y diseño orientado a objetos.* 2004.

25. **McClure, Carma.** *The CASE Experience.* Abril 1989.

26. *Componente de clasificación de huellas dactilares .* **Hernández, Alicia Delgado.** La Habana : UCI, 2013.

27. Visual Paradigm. [En línea] http://www.visual-paradigm.com/.

28. **Baeza, Pablo Nicolás.** Visual Paradigm DB Visual ARCHITECT SQL. [En línea] http://www.docstoc.com/docs/96492173/Visual.

29. **Duque, Raúl González.** *Python para todos.* pág. 115.

30. **Martelli, Alex.** *Python.* 2008. 978-84-415-2317-3 84-415-2317-7.

31. **Dayley, Brad.** *Python Phrasebook.* 2007.

32. **Herminio Santos Heredia, ¿Qué es el PHP?** Maestros del Web. [En línea] 2001. [Citado el: 15 de enero de 2015.] http://www.maestrosdelweb.com/phpintro.

33. *Herramienta para informatizar la gestión de los subprocesos pruebas y revisiones del aseguramiento de localidad en el Centro GEySED.* **Yeilen Morales Aliaga, Lisbet Daniel Rodríguez.** La Habana : UCI, 2013.

34. **Gutérrez, Javier J.** *¿Qué es un framework web?* Sevilla : Universidad de Sevilla, 2011.

35. **Cobo, Ángel, Gómez, Patricia, Pérez, Daniel and Rocha, Rocío.** *PHP y MySQL Tecnologías para el desarrollo de aplicaciones web.* 1ra. España. 84-7978-706-6.

36. **Barahona, Jesús M. González.** *Intoducción al software libre.* Madrid : s.n.

37. **Silva, Vladimir.** *Practical Eclipse Rich Client Platform Projects.* New York : Apress : s.n., 2009. 978-1-4302- 18128-9.

38. PR Newswire. [En línea] [Citado el: 5 de noviembre de 2014.] http://www.prnewswire.co.uk/news-releases/con-pycharm-los-desarrolladores-de-python-obtienen-finalmente-una-ide-potente-155001805.html.

39. PyCharm. [En línea] [Citado el: 10 de noviembre de 2014.] http://www.jetbrains.com/pycharm/ .

40. **Kabir, Mohammed J.** *La biblia del servidor Apache.*

41. **Sommerville, Ian.** *Ingeniería de Software. Séptima edición.* Madrid : PEARSON EDUCATION, 2005. pág. 712. Vol. Parte II. 84-7829-074-5.

42. **Letelier, Patricio y Penadés, Mª Carmen.** Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP). España : Universidad Politécnica de Valencia.

43. **GAMMA, E., HELM, R. y JOHNSON, R. y VLISSIDES, J.** *Patrones de diseño.* 2000.

44. **Cooper, James W.** *The Design Patterns.Java Companion.* octubre 2, 1998.

45. **Alvarado, José Manuel.** *Bases de datos. Unidad 2. Modelo de datos.* s.l. : Universidad Nacional de Ingeniería.

46. **Pressman, Roger S.** *Ingeniería del Diseño.Ingeniería de Software.Sexta Edición.* s.l. : McGraw Hill. Vol. Parte II.

47. **Pressman, Roger .S.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 382.

48. **Microsoft.** [En línea] [Citado el: 31 de marzo de 2015.] http://msdn.microsoft.com.

49. **Universidad de las Ciencias Informáticas.** Entorno Virtual de Aprendizaje . [En línea] 2013. [Citado el: 24 de marzo de 2014.] http://eva.uci.cu/file.php/158/Documentos/Recursos_bibliograficos/Libros_y_articulos_UD_2/Comun/Material_de_caja_b_y_caja_n.pdf.

50. **Pressman, Roger .S.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 404.

51. **Pressman, Roger.** *Ingeniería de Sofware.Un enfoque práctico.Capítulo 13.* Sexta edición. pág. 406.

52. **Montero, Sergio Infante.** *Django para perfeccionistas con deadlines.* s.l. : Eugenia Tobar. pág. 103.

53. **Holovaty, Adrian y Kaplan-Moss.** *The Definitive Guide to Django: Web Development Done Right.* 2009. 978-1-4302-1937-8.

54. Web2py. [En línea] [Citado el: 24 de marzo de 2015.] http://www.web2py.com.ar/examples/default/what.

55. **Álvarez, Miguel Angel.** Manual de jQuery. [En línea] [Citado el: 26 de abril de 2015.] http://www.desarrolloweb.com/manuales/manual-jquery.html.

56. **Sánzhez, Jorge.** *Sistemas Gestores de Base de Datos.* 2009.

57. **Neil, Matthew y Stones, Richard.** *Beginning Databases with PostgreSQL.* 2005.

58. **POSTGRESQL.** Sobre PostgreSQL. [En línea] [Citado el: 20 de octubre de 2014.] http://www.postgresql.org.es/sobre_postgresql.

Glosario de términos

Personalización: proceso que describe el flujo de trabajo para recibir, procesar, imprimir y asociar los datos que identifican a una persona con un documento emitido por una entidad.

Multiplataforma: se refiere a los programas informáticos que pueden funcionar en diversas plataformas.

Marco de trabajo: estructura de artefactos o módulos concretos con base en la que otro proyecto de *software* puede ser desarrollado.

Apache: servidor web de distribución libre y de código abierto.

Módulo: es una porción de un programa de computadora. De las varias tareas que debe realizar un programa para cumplir con su función u objetivos, un módulo realizará, comúnmente, una de dichas tareas.

Metodología: se refiere a una forma de trabajo o un marco de trabajo que es usado para estructurar, planear y controlar el proceso de desarrollo en sistemas informáticos.

Orden: encapsula los datos generales de un conjunto de solicitudes, así como el tipo de documento y la oficina de procedencia.

Solicitud: es el par atributo-valor, donde se almacenan los datos referentes a un documento de identificación, según su tipo.

Acrónimos

C	CASE: Ingeniería de Software Asistida por Ordenador. CISED: Centro de Identificación y Seguridad Digital. CRC: Tarjeta Clase-Responsabilidad-Colaboración.
D	DATYS: Empresa de Desarrollo de Aplicaciones, Tecnologías y Sistemas.
E	EMIMAR: Sistema Cubano de Emisión de la Credencial del Marino.
F	FDD: Acrónimo de *Feature Driven Development.*
G	GOF: Banda de los Cuatro. GRASP: Patrones Generales de Asignación de Responsabilidades.
H	HU: Historias de Usuario.
I	IDE: Acrónimo de *Integrated Development Environment*, también conocido como entorno de diseño integrado o entorno de depuración integrada.
M	MVC: Modelo Vista Controlador. MVP: Modelo Vista Plantilla. MVCC: Acrónimo de *Multi-Version Concurrency Control*, permite el acceso a una tabla sin necesidad de bloqueos, aunque otros procesos estén escribiendo simultáneamente en ella.

O	OACI: Organización de Aviación Civil Internacional.
P	PDF417: Código de longitud variable que puede codificar virtualmente cualquier letra, número o carácter. PET-F: Tarjetas plásticas de politereftalato de etileno. PVC: Tarjetas plásticas de policloruro de vinilo.
Q	QR: Código de respuesta rápida.
S	SGBD: Sistema Gestor de Base de Datos. SPDI: Sistema de Personalización de Documentos de Identificación. SVG: Gráficos Vectoriales Redimensionables (del inglés *Scalable Vector Graphics*) o SVG son una especificación para describir gráficos vectoriales bidimensionales, tanto estáticos como animados, en formato XML.
U	UCI: Universidad de las Ciencias Informáticas. UML: Lenguaje Unificado de Modelado.
X	XP: Programación Extrema. XML: Acrónimo de *eXtensible Markup Language*, es un lenguaje de marcas que permite estructurar documentos para el intercambio de información entre diferentes plataformas.